Preliminary

PDP-11/34 User's Guide

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual. (Revised March, 1976)

Printed in U.S.A.

Copyright © 1976 by Digital Equipment Corporation
Written by PDP-11 Engineering
OPERATOR'S GUIDE

1.1 INTRODUCTION
1.2 BOOTSTRAP PROGRAMS
1.3 OPERATORS CONSOLE
1.4 THE CONSOLE EMULATOR
1.5 OPERATING ERRORS & HOW TO CORRECT THEM

PROGRAMMER'S GUIDE

2.1 GENERAL
2.2 GENERAL REGISTERS & ADDRESSING MODES
2.3 INSTRUCTION OPERANDS
2.4 ADDRESSING MODES
2.5 INSTRUCTION SET
2.6 INSTRUCTION EXECUTION TIME
2.7 EXTENDED INSTRUCTION SET
2.8 PROGRAMMING THE DL11-W
2.9 CONSTRUCTING A SOFTWARE SWITCH REGISTER

HARDWARE GUIDE

3.1 PDP-11/34 SYSTEM DESCRIPTION
3.2 OPERATORS CONSOLE
3.3 M9301-YA & YB TERMINATOR/ROM
3.4 PDP-11/34 PROCESSOR
3.5 MEMORY
3.6 MEMORY PARITY
3.7 DL11W SERIAL LINE INTERFACE/REAL TIME CLOCK
3.8 POWER SUPPLIES
3.9 UNIBUS
3.10 TERMINATORS
3.11 DD11-PKBACKPLANE

M9301 SOFTWARE GUIDE

4.1 OEM VERSION (M9301-YA)
4.2 END USER VERSION (M9301-YB)
4.3 GENERAL
4.4 BOOTSTRAPS
4.5 DEVICES SUPPORTED BY M9301
4.6 PROGRAM MEMORY MAP

CONFIGURATION GUIDE

5.1 AN INTRODUCTION TO THE ELEMENTS OF CONFIGURATION
5.2 FACTORY CONFIGURATIONS - THE CHART SYSTEM
5.3 NEW CONFIGURATIONS
5.4 CONFIGURATION LOG

INSTALLATION GUIDE

6.1 SCOPE
6.2 ENVIRONMENTAL REQUIREMENTS
6.3 ELECTRICAL REQUIREMENTS
6.4 UNPACKING
6.5 INITIAL INSPECTION
6.6 SWITCH SETTINGS
6.7 MODULE PLACEMENT
6.8 FIRST TIME START-UP
6.9 CHECK STANDBY OPERATION
6.10 CABINET MOUNTING

TROUBLESHOOTING GUIDE

7.1 INTRODUCTION
7.2 HOW TO REPORT A FAILURE
7.3 TROUBLESHOOTING THE PDP-11/34
7.4 TROUBLESHOOTING FLOW CHARTS

APPENDICES

A. EXTENDED ADDRESSING
B. A REVIEW OF THE OCTAL NUMBER SYSTEM
C. STANDARD ABBREVIATIONS
D. GLOSSARY
The **PDP-11/34 User Manual** is designed to be a comprehensive document which supplies users with the basic information required to operate the basic **PDP-11/34** computer systems.

This manual assumes that the reader has a knowledge of fundamental computer theory. A knowledge of the Octal Number System is essential to the operation of this machine. A review of the Octal Number System is provided in Appendix B.
HOW TO USE THIS MANUAL

This manual is divided into seven guides for easy reference.

****************************First Time Users******************************

Read the manual in the following order:

Hardware Guide
M9301 Overview Section
Programmer's Guide
Operator's Guide

before attempting to operate the machine.

****************************Do Not Install Machine************************

Unless you Read:

All First Time User’s Material stated above,
Configuration Guide
Installation Guide

****************************Quick Reference Chart*************************

Once the reader is familiar with the manual, the quick reference chart on the next page can quickly direct you to your area of interest.
OPERATORS

GUIDE
CONTENTS

CHAPTER 1  THE PDP-11/34 OPERATOR'S GUIDE

1.1  INTRODUCTION
1.1.1  A New Breed Of Machine

1.2  BOOTSTRAP PROGRAMS
1.2.1  Definition Of Bootstrap Programs
1.2.2  Location Of Bootstrap Programs
1.2.3  Indestructible Bootstraps
1.2.4  Boot = Bootstrap = Bootstrap Programs

1.3  OPERATOR'S CONSOLE
1.3.1  Overview
1.3.1.1  Illustration Of Operator's Console
1.3.2  Description Of Switches
1.3.2.1  POWER
1.3.2.2  HALT/CONT
1.3.2.3  BOOT/INIT
1.3.2.3.1  What Can Boot Do?
1.3.2.3.1.1  Selection Of Features
1.3.2.3.1.2  Standard Setting Of M9301 Switches
1.3.2.3.1.3  Non-standard Setting Of M9301 Switches
1.3.2.3.2  Hardware Close Up Of BOOT
1.3.3  Description Of Indicator Lights
1.3.3.1  BATT
1.3.3.2  DC ON
1.3.3.3  RUN

1.4  THE CONSOLE EMULATOR
1.4.1  Overview
1.4.2  A Summary Of The Console Emulator Functions
1.4.3  Console Emulator Operation
1.4.4  Discussion Of The Simplified Operations Flow Chart
1.4.4.1  Introduction
1.4.4.2  Symbols
1.4.4.2.1  Rectangle
1.4.4.2.2  Diamond
1.4.4.2.3  Circle
1.4.4.2.4  (CR)
1.4.4.3  A Simplified Operator's Flow Chart Of The Console Emulator Routine
1.4.5  Entry Into The Console Emulator Routine
1.4.5.1  POWER Switch Entry
1, 4, 5, 2
1, 4, 5, 3
1, 4, 5, 4
1, 4, 5, 5
1, 4, 5, 6
1, 4, 5, 7
1, 4, 5, 8
1, 4, 5, 9
1, 4, 5, 10
1, 4, 5, 11
1, 4, 5, 12
1, 4, 5, 13
1, 4, 5, 2, 1
1, 4, 5, 2, 2
1, 4, 5, 2, 3
1, 5
1, 5, 1
1, 5, 1, 1
1, 5, 1, 2
1, 5, 1, 3
1, 5, 1, 4
1, 5, 2
1, 5, 2, 1
1, 5, 2, 2
1, 5, 2, 3
1, 5, 2, 3, 1
1, 5, 2, 3, 2
1, 5, 2, 3, 3
1, 4, 6
1, 4, 6, 1
1, 4, 6, 2
1, 4, 6, 3
1, 4, 6, 4
1, 4, 6, 5
1, 4, 6, 6
1, 4, 6, 7
1, 4, 6, 8
1, 4, 6, 9
1, 4, 6, 10
1, 4, 6, 11
1, 4, 6, 11, 1
1, 4, 6, 11, 2
1, 4, 6, 11, 3
1, 4, 6, 11, 4
1, 4, 6, 11, 5

BOOT/INIT Switch Entry
Power Return Entry
Using The Console Emulator
After The &
Keyboard Input Symbols
Keyboard Input Format: Load - Examine - Deposit - Start
Order Of Significance Of Key Entries
Number Of Characters
Leading Zeros
Example Using The: Load - Examine - Deposit - Start Functions
Even Addresses Only
Successive Operations
Examine
Example Of Successive Examine Operations
Deposit
Example Of Successive Deposit Operations
Alternate Deposit - Examine
Example Of Successive Deposit - Examine Operations
Alternate Examine - Deposit
Limits Of Operation
Boot From The Keyboard
Console Emulator Boot Procedure
Table Of Bootstrap Routine Codes
Before Booting ...
Example: Booting The High Speed Reader Using The Console Emulator
Example: Booting A Disc Using The Console Emulator

OPERATING ERRORS AND HOW TO CORRECT THEM
Operator's Console
POWER Switch
HALT/CONT Switch
BOOT/INIT Switch
Console Emulator
Symbols
Escape Route
Tables Of Errors
Table Of Load Errors
Table Of Examine And Start Errors
Table Of Deposit Errors
CHAPTER 1
OPERATOR'S GUIDE

1.1 INTRODUCTION

This guide presents the mechanics of operating the 11/34 computer systems.

1.1.1 A New Breed Of Machine

Traditionally, the PDP-11 family of computers has utilized a front panel which contained a switch register and light display that allowed the user to communicate with the machine.

The PDP-11/34 computers introduce the next generation of DEC Man-machine interfaces which communicate with the user via the system terminal. This new generation of machine is symbolized by the simple, distinctive Operator's Console and the graphical record of the Operator-Machine dialogue.

1.2 BOOTSTRAP PROGRAMS

In the following discussion, many references will be made to bootstrap programs.

1.2.1 Definition Of Bootstrap Program

A Bootstrap program is a program which allows the loading of another (usually larger) program.

1.2.2 Location of Bootstrap Program

The basic Bootstrap Programs for the PDP-11/34 Computer Systems are located on the M9301 Bootstrap/Terminator Module. Read Only Memory (ROM) on that module is the storage medium.
1.2.3 Indestructible Bootstraps

All Bootstrap Programs located in the M9301 are indestructable. These programs are not affected by the removal of power, or user attempts to alter them by writing into their assigned locations.

1.2.4 Boot - Bootstrap - Bootstrap Program

Boot is a verb which means to initiate execution of a Bootstrap Program.

Bootstrap and Bootstrap Program are used interchangeably.

1.3 OPERATOR'S CONSOLE

1.3.1 Overview

The Operator's Console is the first link between the user and the machine.

Switches on the Console allow the user to:

- Control DC power to the system
- Place the machine in the halt state
- Continue running from the halt state
- Boot
- Initialize the system

Indicator lights notify the user of:

- The condition of the battery (if battery backup option is present)
- The presence of DC power
- The status of machine - run or halt.

1.3.2 Description of Switches

1.3.2.1 POWER - This three position rotary switch allows the following modes of operation:
DCON - The system is fully powered.

STANDBY - DC power is only applied to the MOS memory to avoid loss of information, fans remain on (Note: This applies to systems housed in the 5 1/4" high, BA11L mounting box only, not the 10 1/2" high BA11K mounting box).

DCOFF* - DC power is removed from the system. The contents of MOS memory are lost, fans are off.

1.3.2.2 HALT/CONT - When placed in the HALT position, this switch will cause the processor to halt. The system (including keyboard functions) can not run in this position. Moving the switch to the CONT position causes the machine to continue.

If the program causes the system to halt, this switch must first be placed in the HALT position and then moved to the CONT position to resume operation.

If a trap causes a halt, it is advisable to reboot the program unless the operator is certain that the program is structured to continue from traps.

1.3.2.3 BOOT/INIT - This switch is a spring action momentary switch which is normally in the BOOT position.

It only initiates action while being moved to or from the INIT position.

To INITIALIZE and BOOT: Move this switch from the BOOT position to the INIT position. Holding the switch in this position will cause a continuous INITIALIZE. When the switch is released, the system will BOOT. The above sequence assumes that the HALT/CONT switch is in the CONT position.

To BOOT without INITIALIZING:

1. Place the HALT/CONT switch in the HALT position.

2. Move the switch to the INIT position and return to the BOOT position.

3. Place the HALT/CONT switch in the CONT position.

BOOTING without INITIALIZING allows examination of

----------
* Caution should be exercised when servicing the machine, to remove AC power, disconnect the AC line cord.
device registers which would be cleared by INIT.

*********************************************************************************************

CAUTION

Depressing the BOOT switch while running a program, will abort that program. Once aborted, it is not possible to continue execution of that program. See section 1.5.1.3 for additional information (it is possible to restart some programs).

*********************************************************************************************

1.3.2.3.1 What Can BOOT Do? - Depressing BOOT can:

1. For both the M9301-YA and M9301-YB versions:
   - print R0, R4 SP and PC see Sec. 1.4.5.
   - run a GO = NO GO diagnostic.
   - Display a $ on the terminal display to signify passing the diagnostic and the availability of the Console Emulator Routine (see Sec. 1.4.6).

OR:

2. For the M9301YA version only:
   - run a Go-No Go diagnostic
   - run a memory diagnostic
   - boot a preselected peripheral directly.

1.3.2.3.1.1 Selection of M9301 Features - The M9301 Bootstrap/Terminator module contains a bank of switches, whose settings determine which of the forementioned features will be implemented. See the Configuration Guide, Section for a detailed description of setting these switches.

1.3.2.3.1.2 Standard Switch Settings - The standard switch settings will be to enable all M9301 diagnostic tests and the console emulator routine for all PDP-11/34 systems except where the user does not specify a preference.
1.3.2.3, Non-Standard Switch Settings on the M9301 - The *OEM* (Original Equipment Manufacturer) version M9301YA can only a peripheral instead of entering the console emulator routine, when power comes up or the boot switch is depressed.

Which device is booted will depend on the switch settings of the M9301. See the Configuration Guide, Section 4.1 for a discussion of M9301 switch settings.

Utilizing a peripheral bootstrap in place of the console emulator demands that the user provide a program-controlled means of accessing the console emulator routine, otherwise this feature will be lost.

It is not possible to boot a peripheral directly with the end user M9301YB version, regardless of switch settings.

1.3.2.3.2 Hardware Close up of BOOT Function - The BOOT function is implemented by generating an AC LO signal to the M9301 module. This causes the M9301 to go to its power up routine which is: to do preliminary processor diagnostics and then enter the console emulator routine for M9301 modules with the standard switch settings.

1.3.3 Description of Indicator Lights

1.3.3.1 "BATT" - This indicator continuously monitors the condition of the battery in MOS machines which have the battery backup option. This light has four distinct states:

- Off = Indicates the total discharge or absence of the battery.
- Slow Flash (1 flash/2 seconds) = Indicates that power is within allowable limits and the battery is charging. If a power failure should occur while the battery is in this condition, the battery will maintain memory for less time than if it were fully charged.

How much less time will depend on the degree of discharge. The flash rate is fixed and does not vary with the charge rate of the battery.
- Fast Flash = (10 flashes/second) = Indicates the loss of
power and that battery is discharging while maintaining MOS memory contents. The flash rate is fixed and does not vary with the charge level remaining on the battery.

On (continuous) - indicates the battery is present and fully charged.

1.3.3.2 "DC ON" - Whenever the power switch is "on" this light should be lit, indicating the presence of DC power to the logic.

1.3.3.3 "RUN" - When this indicator is on, the machine is in one of the following states:

1. Running the user's program.
2. Failed the internal Go-No Go diagnostic and is executing a programmed loop.
3. Performing an unanticipated loop as a result of a programming error.
4. Attempting to run but disabled as the result of a system hardware failure.

1.4 CONSOLE EMULATOR, LOAD, EXAMINE, DEPOSIT, START AND BOOT FUNCTIONS

1.4.1 Overview

The M9301 bootstrap/terminator module contains a console emulator routine. When this routine is used in conjunction with the user's terminal, functions quite similar to those found on the programmer's console of traditional PDP-11 family computers are generated, as shown in the summary below. The description of operation that follows assumes that the user's M9301 bootstrap/terminator module is configured* to enter the console emulator routine on power up or whenever the boot switch is depressed.

1.4.2 A Summary of the Console Emulator Functions

LOAD - This function loads the address to be

* See configuration guide
manipulated into the system.

EXAMINE = Allows the operator to examine the contents of the address that was loaded.

DEPOSIT = Allows the operator to write into the address that was loaded and/or examined.

START = Initializes the system and starts execution of the program at the address loaded.

BOOT = Allows the booting of a specified device by typing in a two character code.

1.4.3 Console Emulator Operation

The console emulator allows the user to perform LOAD, EXAMINE, DEPOSIT, START, and BOOT functions by typing in the appropriate code on the keyboard. The combination of the console emulator routine and the keyboard will be referred to as "The Console Emulator".

1.4.4 Discussion of the Simplified Operator's Flow Chart

1.4.4.1 Introduction - The Simplified Operator's Flow Chart, Fig. 1-2 of the Console Emulator Routine (which will be referred to as the Operator Flow Chart) is presented at this point in the text to give the reader a unified picture of the Console Emulator Routine.

Detailed discussions of each aspect of the chart are presented in sections 1.4.5 to 1.4.6.
**Figure 1-2**

Simplified Operator’s Flowchart of the Console Emulator Routine

*(For Standard M9301 Configurations)*

1. The routine is entered on power up or when the boot switch is depressed (in standard M9301 configurations) or upon the return of AC power (after an interruption of AC power).

2. A diagnostic located on the M9301 tests CPU operation.

3. If the diagnostic is passed, the system terminal will print out four, six digit numbers.

4. The system is now waiting for the operator to enter information from the keyboard.

5. The operator enters a code (described in this section) which will load an address or examine a previously loaded address or deposit into a previously loaded address, or start at a previously loaded address or boot a selected peripheral.

6. The first two input characters are verified as being a known combination.

7. If the key combination indicated a start, the start will be implemented now, at the previously loaded address. Note that start will exit from this routine to enter again, the operator must boot.

8. If the key combination was an examine, the previously loaded address and its contents will be printed on the system terminal.

9. If the code was not an examine or start, the console emulator jumps to load or deposit or boot.

---

**Load Sequence**

10. Sequence is entered
11. Address is entered and significant digit is first
12. A check is made to insure that the number which was entered was an octal number 0-7. If it isn’t, a 5 is printed signifying an “error.”
13. Entering CR signifies the end of the load sequence.
14. Address is loaded into an internal register and 6 is printed signifying the system is ready for next keyboard input.
15. CR indicates that the operator wishes to deposit now.
16. Data is deposited followed by a 5 printout, signifying the system is ready for next keyboard input.

**Deposit Sequence**

17. Address to be loaded
18. Data to be deposited
19. Octal number
20. Enter CR
21. Successive deposit
22. Increment address
23. Enter CR
24. Memory diagnostic passed, device is now booting.
25. Device is booted

**Boot Sequence**

26. Load
27. Deposit
28. Boot
29. Address to be loaded
30. Data to be deposited
31. Octal number
32. Enter CR
33. Successive deposit
34. Increment address
35. Enter CR
36. Memory diagnostic failed.
37. Device is rebooted

*CR = Carriage Return Key*
1.4.4.2 Symbols

1.4.4.2.1 Rectangle - Rectangles indicate automatic operations which are performed by the machine. There is only one entrance and one exit on a rectangle.

1.4.4.2.2 Diamond - A diamond indicates an automatic operation which can take either of two paths depending on how the question stated within the diamond is answered.

1.4.4.2.3 Circle - A circle indicates operator action, the moving of a switch or the typing of keys.

1.4.4.2.4 CR - CR is the symbol for the Carriage Return key.

1.4.4.3 Commentary - The commentary on the left side of the Operation Flow Chart provides information which supplements information in the flow chart itself.

1.4.4.4 Limitations of Flow Chart - The Flow Chart is intended to present an overview of the Console Emulator only. Detailed discussions of the Console Emulator (with numerous examples) are contained in section 1.4.5 through 1.4.6.

1.4.5 Entry Into the Console Emulator

There are three ways of entering the Console Emulator:

- Move the Power Switch to the On position.
- Depress the BOOT Switch.
- Automatic entry on return from a power failure.

1.4.5.1 Power Switch - When the operator moves the power switch from Off to On, a series of numbers representing the contents of R0, R4, SP and OLD PC respectively, will be printed by the terminal. This sequence will be followed by a $ on the
Example - a typical printout on power up:

```
XXXXXX*   XXXXX   XXXXX   XXXXX
```

R0        R4        R6        OLD PC**
PROMPT    STACK     PROGRAM
CHARACTER POINTER  COUNTER
(SP)

1.4.5.2  BOOT Switch Entry - Whenever the BOOT Switch
is depressed, the system will printout R0, R4, SP and
OLD PC followed by a prompt character, as described above,
in addition to booting the system.

This feature is especially valuable when the system has
halted unexpectedly.

The operator can determine where the system has halted by
examining the OLD PC and subtracting two.

*****************************************************************************

CAUTION

If this procedure is followed, it will alter the contents of
General Registers making it impossible to continue with
the program.

*****************************************************************************

1.4.5.3  Power UP Entry - If AC power should fail and
return again, the system will go through a power up
boot sequence.

1.4.6  Using the Console Emulator

---------

* Note: X signifies an octal number (0-7).

--------

** Whenever there is a power up routine, or the BOOT switch is
released from the INIT position, the PC at this time will be stored
in R5. The contents of R5 are printed out as shown above (noted as
the old PC).
1.4.6.1 After the $ - Once the system has been powered up or booted, and R0, R4, SP, PC and $ have been printed, the Console Emulator routine can be used.

1.4.6.2 Keyboard Input Symbols - The discussion of keyboard input format uses the following symbols:

- Space Bar: (SB)
- Carriage Return Key: (CR)
- Any number 0-7 (Octal Number Key: (X))

1.4.6.3 Keyboard INPUT Format - Load, examine, deposit, start. All character keys shown in the following discussion represent themselves with the exception of those in parenthesis.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>KEYBOARD STROKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load address</td>
<td>L (SB) (X) (X) (X) (X) (X) (CR)</td>
</tr>
<tr>
<td>Examine</td>
<td>E (SB)</td>
</tr>
<tr>
<td>Deposit</td>
<td>D (SB) (X) (X) (X) (X) (X) (CR)</td>
</tr>
<tr>
<td>Start</td>
<td>S (CR)</td>
</tr>
</tbody>
</table>

1.4.6.4 Order of Significance of Input Keys - The first character that is typed will be the most significant character. Conversely, the last character that is typed is the least significant character.

1.4.6.5 Number of Characters - The console emulator routine can accept up to six octal numbers in the range of 0-32K. If all six numbers are inputted, the most significant number should be a one or a zero.

1.4.6.6 Leading Zeros - When an address or data word contains leading zeros, these zeros can be omitted when loading the address or depositing the data.

1.4.6.7 Example Using the Load, Examine, Deposit, and Start Function - Assume that a user wishes to:

1. Turn on power
2. Load address 700
3. Examine location 700
4. Deposit 777 into location 700
5. Examine location 700
6. Start at location 700

**USER**
1. turns on power --> XXXXXX XXXXXXX XXXXXXX XXXXXXX

**TERMINAL DISPLAY**

2. L(SB) 700 (CR) \( \rightarrow \) \$L700
3. E(SB) \( \rightarrow \) \$E 000700 XXXXX
4. D(SB) 777 (CR) \( \rightarrow \) \$D 777

5. E (SB) \( \rightarrow \) \$E 000700 000777
6. S (CR): \( \rightarrow \) \$S

1.4.6.8 Even Addresses Only - The console emulator routine will not work with odd addresses. Even numbered addresses must always be used.

1.4.6.9 Successive Operations

1.4.6.9.1 Examine - Successive examine operations are permitted. The address is loaded for the first examine only. Successive examines cause the address to increment and will display consecutive addresses along with their contents.

1.4.6.9.1.1 Example of Successive Examine Operations - Examine Addresses 500-506

<table>
<thead>
<tr>
<th>Operator Input</th>
<th>Terminal Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>L (SB) 500 CR</td>
<td>$L 500</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000500 XXXXX</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000502 XXXXX</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000504 XXXXX</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000506 XXXXX</td>
</tr>
</tbody>
</table>

1.4.6.9.2 Deposit - Successive deposit operations are permitted. The procedure is identical to that used with examine.
1.4.6.9.2.1 Example of Successive Deposit Operations

Deposit: 60 into Location 500
      2 into Location 502
      4 into Location 504

<table>
<thead>
<tr>
<th>Operation Input</th>
<th>Terminal Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>L (SB) 500 CR</td>
<td>$L 500</td>
</tr>
<tr>
<td>D (SB) 60 CR</td>
<td>$D 60</td>
</tr>
<tr>
<td>D (SB) 2 CR</td>
<td>$D 2</td>
</tr>
<tr>
<td>D (SB) 4 CR</td>
<td>$D 4</td>
</tr>
</tbody>
</table>

1.4.6.9.3 Alternate Deposit-Examine Operations -
This mode of operation will not increment the address. The address will contain the last data which was deposited.

1.4.6.9.3.1 Example of Alternate Deposit-Examine Operations -
Load address 500, deposit the following numbers with examines after every deposit: 1000, 2000, 5420.

<table>
<thead>
<tr>
<th>Operation Input</th>
<th>Terminal Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>L (SB) 500 (CR)</td>
<td>$L 500</td>
</tr>
<tr>
<td>D (SB) 1000 (CR)</td>
<td>$D 1000</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000500 001000</td>
</tr>
<tr>
<td>D (SB) 2000 (CR)</td>
<td>$D 2000</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000500 002000</td>
</tr>
<tr>
<td>D (SB) 5420 (CR)</td>
<td>$D 5420</td>
</tr>
<tr>
<td>E (SB)</td>
<td>$E 000500 005420</td>
</tr>
</tbody>
</table>

1.4.6.9.4 Alternate Examine-Deposit Operations - If an examine is the first operation after a load sequence, and is alternately followed by deposits and examines, the address will not be incremented, and the address will contain the last data which was deposited. (The above example applies to this operation, with the exception of the order of examine and deposit, the end result is the same).

1.4.6.10 Limits of Operation - The M9301 console emulator routine can directly manipulate the lower 28K of memory and the 4K I/O page. See Appendix A for an explanation of techniques required to access addresses above the lower 28K.

1.4.6.11 Booting from the Keyboard - Once the $ symbol has
been displayed in response to system power coming up, or the boot switch being depressed, the system is ready to load a bootstrap into the device which the operator selects.

1.4.6.11.1 Console Emulator Boot Procedure

1. Find the two character boot code on table 1-1 that corresponds to the peripheral to be booted.

2. Load medium, papertape, magtape, disc, etc., into the peripheral if required.

3. Verify that the peripheral indicators signify that the peripheral is ready (if applicable).

4. Type the two character code obtained from the table.

5. If there is more than one unit of a given peripheral, type the unit number to be booted (0-7) if no number is typed the default number will be 0.

6. Type (CR), this initiates the boot.

1.4.6.11.2 Table of Bootstrap Routine Codes - Supported by both YA and YB versions of the M9301.

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>RK11</td>
<td>Disk cartridge</td>
<td>DK</td>
</tr>
<tr>
<td>RP11</td>
<td>RP02/03 disk pack</td>
<td>DP</td>
</tr>
<tr>
<td>TC11</td>
<td>DECTAPE</td>
<td>DT</td>
</tr>
<tr>
<td>TM11</td>
<td>800 BPI Magtape</td>
<td>MT</td>
</tr>
<tr>
<td>TA11</td>
<td>Magnetic cassette</td>
<td>CT</td>
</tr>
<tr>
<td>RX11</td>
<td>Diskette</td>
<td>DX</td>
</tr>
<tr>
<td>DL11</td>
<td>ASR-33 teletype</td>
<td>TT</td>
</tr>
<tr>
<td>PC11</td>
<td>Papertape</td>
<td>PR</td>
</tr>
</tbody>
</table>

Supported by the YB version only (in addition to all the above),

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJS03/04</td>
<td>Fixed Head disk</td>
<td>DS</td>
</tr>
<tr>
<td>RJP04</td>
<td>Disk pack</td>
<td>DB</td>
</tr>
<tr>
<td>TJU16</td>
<td>Magnetic tape</td>
<td>MM</td>
</tr>
</tbody>
</table>

TABLE 1-1

1.4.6.11.3 Before Booting... - Always remember:

1. The medium (papertape, disc, magtape, cassette, etc.,) must
be placed in the peripheral to be booted prior to booting.

2. The machine will not be under the control of the console emulator routine after booting.

3. The program which is booted in must: 1) be self starting or 2) allow the user to load another program by using the CONT function or 3) be restartable after the console emulator is recalled.

4. Actuating the boot switch will always abort the program being run. The contents of the general registers (R0-R7) will be destroyed. There is no way to continue with the program which was aborted. Some programs are designed to be restartable.

146114 Example: Booting the High Speed Reader Using the Console Emulator - An operator wishes to load the CPU diagnostic for an 1134 computer system. The system has a high speed reader.

Procedure:

1. Place the HALT/CONT switch in the CONT position.

2. Obtain a $ by:
   a. Booting the system, (see sec. 145).
   b. Actuating the boot (R0, R4, SP & PC will be printed prior to the $).

3. Place the absolute loader paper tape (coded leader section) in the high speed reader.

4. Type: PR (CR)
The absolute loader tape will be loaded, and the machine will halt.

5. Remove the absolute loader and place the leader of the program, in this case a CPU Diagnostic, in the reader.

6. Move the HALT/CONT switch to HALT and then return it to CONT. The Diagnostic will be loaded and the machine will halt (normal, non-diagnostic programs could be self starting).

7. Activate the BOOT/INIT switch, this will restore the console emulator routine.

8. Using the console emulator, deposit desired functions into
the software switch register (a memory address) location, see
the diagnostic for the software switch register's actual
location
and significance.

9. Using the console emulator, load the starting address, and
then the start sequence (see section 1.4.6.3).

1.4.6.11.5 Example of Booting a Disc Using the Console Emulator —
A user wishes to boot the system's RK11 disk, which
contains the CPU diagnostic which the user wants to run.

Procedure:

1. Verify that the HALT/CONT switch is in the CONT
position; and the write lock switch on the RK11 peripheral
is in the ON position.

2. The user turns on system power, the system terminal
displays R0, R4 SP, & PC which are random binary numbers
followed by a $ on the next line.

3. The user places the disk pack into drive zero and places the
RUN/LOAD switch in the run position.

4. When the RK05 RUN light appears, the system is
ready to be booted.

5. The user types in: DK (CR)
This causes the execution of the bootstrap and the
loading of the diagnostic monitor.

6. The program should identify itself and initiate a dialogue
(which won't be discussed here).

1.5 OPERATING ERRORS AND HOW TO CORRECT THEM

Human beings make mistakes. This section deals with possible
operator errors. The purpose of this section is to explain
how the system will react to these errors and suggest
remedies where possible.

1.5.1 Operators Console Operating Errors

The following discussion will examine the consequences of incorrectly
using the POWER, HALT/CONT, and BOOT/INIT switches.
1.5.1.1 Power Switch

Error: The user intended to go to the STANDBY mode and accidentally turned the switch to the OFF position.

Result: Contents of MOS memories will be lost.

Remedy: NONE

1.5.1.2 HALT/CONT Switch

Error: The user unintentionally placed this switch in the halt position.

Result: The result will vary depending on the user’s application. The processor will halt, the program will not be altered, however, MEMORY data acquisition and real time control of peripherals is lost.

Remedy: Returning this switch to the CONT position will resume execution of the program from the program count which was present at the time of the halt.

1.5.1.3 BOOT/INIT Switch

Error: The user actuated this switch unintentionally while a program was running.

Result: All devices which respond to init will be cleared and the M9301 will be activated. The contents of all general registers will be modified. The console emulator will be activated (possible on YA or YB or a peripheral will be booted (YA only). NOTE: It is imperative that the user does not attempt to boot at this time. Any attempt to boot will modify the lower 28K of memory.

Remedy: If the data in the user’s system is critical there are several methods of retrieving it.

The methods are:

1. For small amounts of Data, the console emulator routine examine function can be used if the critical addresses are known.

2. If the program is restartable, the user could load the starting address of the program, and then initiate a start command.
via the console emulator. Some restartable programs modify memory. The user must understand the specific program.

3. If the program is not restartable the user has a major problem. There are two approaches which can be used:

   a. Reboot the system via the console emulator from the starting address of the boot routine. This will bypass the memory modifying diagnostics. The boots will occupy the memory locations shown in table 1-2. This will only work with the OEM version of the M9301 (M9301-YA).

   b. If the user understands the program thoroughly he could write a routine to record his data on another medium (papertape, magtape or disk). The user would have to create a space to work by allocating a section of memory which is known to contain no useful information or record the contents of the area to be destroyed using the examine function.

CONSOLE EMULATOR

The following discussion will describe the effects of entering information incorrectly to the console emulator routine.

1.5.2.1 Symbols

Space Bar (SB)
Carriage Return Key (CR)
Any Octal Number (0-7) Key (X)
Non-Octal Number (8 or 9) Keys (O)

(Y) Represents:

1. All Keys other than numerics which are unknown.

2. Keys which are known but do not constitute a valid code in the context which they are entered.

See Section 1-4 for a discussion of the correct method of operating the console emulator routine.
1.5.2.2 Escape Route - If an entry has not been completed, and the user realizes that an incorrect or unwanted character has been entered, depress the rubout or delete key. This action will void the entire entry, and allow the user to try again.

1.5.2.3 Tables of Errors - The tables of load, examine, deposit, and start errors which follow provide a quick reference for these problems.
<table>
<thead>
<tr>
<th>ERROR</th>
<th>RESULT</th>
<th>REMEDY</th>
<th>OPERATOR TERMINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>L was followed by a key other than (SH).</td>
<td>Terminal display will immediately return a $ to signify an unknown</td>
<td>Try again</td>
<td>L(Y) $</td>
</tr>
<tr>
<td>An illegal (non octal number (8 or 9) is typed after the</td>
<td>the console will ignore the entire address and return a $.</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>correct load entrance and within an otherwise valid number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>An incorrect alpha key is typed after the correct load entrance</td>
<td>Same as illegal number.</td>
<td>Try again</td>
<td>L(SR) $L XXXX</td>
</tr>
<tr>
<td>within an otherwise valid number.</td>
<td></td>
<td></td>
<td>XXXY $</td>
</tr>
<tr>
<td>The most significant octal number in a six bit address is greater</td>
<td>An address will be loaded, however the state of the address bit will</td>
<td>Try again</td>
<td>L(SR)6 $L XXXX</td>
</tr>
<tr>
<td>than one.</td>
<td>be determined by bit 15 only:</td>
<td></td>
<td>XXXX $</td>
</tr>
<tr>
<td></td>
<td>2=0 4=1 6=2 8=1 10=1</td>
<td></td>
<td>(CR)</td>
</tr>
<tr>
<td>An unwanted but legal octal number is loaded.</td>
<td>Address will be loaded, but unchanged.</td>
<td>Try again</td>
<td>L(Sb) $L 1 XXXXX</td>
</tr>
<tr>
<td>An extra (seventh) octal number is typed.</td>
<td>The loaded number will be incorrect</td>
<td>Try again</td>
<td>L(SR) $L 1 XXXX</td>
</tr>
<tr>
<td></td>
<td>the system will accept any size word but will only remember the last</td>
<td></td>
<td>XXXX $</td>
</tr>
<tr>
<td></td>
<td>six characters typed in.</td>
<td></td>
<td>(CR) Actually loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XXXXX</td>
</tr>
<tr>
<td>A memory location higher than the highest memory location available</td>
<td>No errors will result unless a deposit, examine or start is attempted.</td>
<td>Try again</td>
<td>L(SR) $L 1 XXXX</td>
</tr>
<tr>
<td>in the machine is loaded.</td>
<td></td>
<td></td>
<td>XXXX(CR)</td>
</tr>
<tr>
<td>Load entrance and number were extended correctly (CR) wasn't</td>
<td>Machine will wait indefinitely for (CR) $ will not be returned.</td>
<td>Type (CR)</td>
<td>L(SR) $L XXXX</td>
</tr>
<tr>
<td>entered.</td>
<td></td>
<td></td>
<td>XXXX $</td>
</tr>
</tbody>
</table>

* Load entrance refers to the correct inputting of the L(SR) keys.
<table>
<thead>
<tr>
<th>ERROR</th>
<th>RESULT</th>
<th>REMEDY</th>
<th>USEFUL EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>E or $ is followed by a key other than space.</td>
<td>Terminal display will immediately return a $ to signify an unknown code.</td>
<td>Try again</td>
<td>E(X) $E</td>
</tr>
<tr>
<td>Examine or start is attempted to a memory location which is higher than the highest available memory location in the machine (I/O can be examined) or an odd memory location.</td>
<td>The system will hang up when $(SB)$ is executed.</td>
<td>Depress the BOOT switch</td>
<td>E(SE) $E</td>
</tr>
<tr>
<td>Examine is performed without loading an address prior to first examine.</td>
<td>Examine of an unknown address will be performed. It is possible that the system could try to access an address which doesn't exist.</td>
<td>Try again or BOOT if system hangs up</td>
<td></td>
</tr>
<tr>
<td>Start is performed without loading an address prior to starting.</td>
<td>Start at an unknown location will occur.</td>
<td>Reload program and try again</td>
<td></td>
</tr>
</tbody>
</table>

* Hang up = system is executing an illegal operation which will possibly cause loss of console emulator or program execution functions.
ERROR

D was followed by a key other than space.

Deposit is attempted to a memory location which is higher than the highest available memory location in the machine with the exception of the I/O page.

Deposit is performed without loading an address or knowing what address has been previously loaded.

Deposit is attempted into an odd address.

DEPOSIT ERRORS

RESULT

Terminal display will immediately return a $ to signify an unknown code.

The system will hang up when (SR) is executed.

The system will hang up when (CR) is executed.

CORRECT FORMAT D(SR) XXXXXX(CR)

REMEDY

Try again.

USEFUL EXAMPLES

OPERATOR DISPLAY

0(Y) SD

$5

Actuate the boot switch to reboot the system.

1. Immediately following the error perform an examine to determine the address which was deposited into restore original contents if known.

2. Reboot machine.

1. Reboot machine.

2. Reboot machine.

Depress the boot switch.

See 1,5,2,3,2
# TABLE OF CONTENTS

## CHAPTER 2  PROGRAMMERS GUIDE

### 2.1  GENERAL

### 2.2  GENERAL REGISTERS AND ADDRESSING MODES
- 2.2.1 Register R7 (Program Counter)
- 2.2.2 Register R6 (Stack Pointer)
- 2.2.3 Registers R5 through R0

### 2.3  INSTRUCTION OPERANDS

### 2.4  ADDRESSING MODES
- 2.4.1 Direct Addressing Modes
- 2.4.2 Indirect Addressing Modes
- 2.4.3 Program Counter (PC) Addressing

### 2.5  INSTRUCTION SET

### 2.6  INSTRUCTION EXECUTION TIME
- 2.6.1 Basic Instruction Set Timing
- 2.6.2 Latency

### 2.7  EXTENDED INSTRUCTION SET
- 2.7.1 Multiply Instruction
- 2.7.2 Divide Instruction
- 2.7.3 Arithmetic Shift Instruction
- 2.7.4 Arithmetic Shift Combined Instruction

### 2.8  PROGRAMMING THE DL11-W SERIAL LINE INTERFACE
- 2.8.1 Data Formats
- 2.8.2 Device Registers
- 2.8.2.1 Receiver Status Register
- 2.8.2.2 Receiver Data Buffer
- 2.8.2.3 Transmitter Status Register
- 2.8.3 Interrupts
- 2.8.4 Address and Vector Assignments
- 2.8.5 Timing Considerations
- 2.8.5.1 Receiver
- 2.8.5.2 Transmitter
- 2.8.5.3 Break Generation

### 2.9  CONSTRUCTING A SOFTWARE SWITCH REGISTER
CHAPTER 2

PROGRAMMERS GUIDE

2.1 GENERAL

The KD11-E Central Processor is one of a family of PDP-11 processors spanning a broad range of instruction execution speed and system performance. Instruction set differences between the KD11-E and other PDP-11 processors are listed in Table 2-1.

Several features, which are optional with the KD11-A Central Processor (employed with the PDP-11/35 and PDP-11/40 processing systems), are standard with the KD11-E. One is the Extended Instruction Set (EIS) which includes multiply, divide, single precision and double precision multiple shifts. A second feature is the KT11-D compatible Memory Relocation and Management, which includes two special memory management instructions.

In addition to the instruction set differences, this section lists the PDP-11/34 instruction set, instruction execution times, programming the DL11-W serial line interface, and constructing a software switch register.
### TABLE 2-1 PROGRAMMING DIFFERENCES

<table>
<thead>
<tr>
<th></th>
<th>11/05 &amp; 11/10</th>
<th>11/35 &amp; 11/40</th>
<th>11/04</th>
<th>11/34</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. GENERAL REGISTERS (including PC &amp; SP)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. <code>OPR&amp;R,(R)+</code> or <code>OPR&amp;R,*(R)</code></td>
<td>Initial contents of R are used as the source operand.</td>
<td>Contents of R are incremented by 2 for decrement by 2 before being used as the source operand.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>or <code>OPR&amp;,,(R)+</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or <code>OPR&amp;R,*(P)</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Using the same reg. as both source &amp; destination)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. <code>JMP(R)+</code> or <code>JSR</code></td>
<td>Contents of R are incremented by 2 then used as the new PC address.</td>
<td>Initial contents of P are used as new PC.</td>
<td>Same as 11/40</td>
<td>Same as 11/40</td>
</tr>
<tr>
<td>reg, (R)+ (jump using auto increment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. <code>MOV PC, #A</code> or <code>MOV PC, A</code></td>
<td>Location A will contain PC +2.</td>
<td>Location A will contain the PC of the move instruction +4.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>(Moving the incremented PC to a memory address referenced by the PC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Stack Pointer (SP), R6 used for referencing.</td>
<td>Using the SP for pointing to odd addresses or non-existent memory causes a halt (double bus error)</td>
<td>Odd address or non-existent memory references with SP cause a fatal trap with a new stack created at locations 6 and 2.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>E. Stack Overflow</td>
<td>Stack limit fixed at 400 (octal) overflow (going lower) checked after mode 4 &amp; 5 using R6 and JSR and traps. Overflow serviced by an overflow trap no Red zone.</td>
<td>Variable limit with stack limit option. Overflow checked after JSR, TRAPS and address modes 1, 2, 4, &amp; 6, non-altering references to stack data is always allowed. There is a 16 word yellow (warning) zone.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>E. Stack Overflow (Cont)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. TRAPS &amp; INTERRUPTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A. RTI Instruction</td>
<td>B. RTI Instruction</td>
<td>C. Processor status odd byte at location 777777</td>
<td>D. T bit of PS</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>11/05 &amp; 11/10</td>
<td>11/35 &amp; 11/40</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>A</td>
<td>First instruction after RTI instruction is always executed.</td>
<td>If RTI sets the T bit, the T bit trap is acknowledged immediately after the RTI instruction.</td>
<td>Odd byte of PS can be addressed w/out a trap.</td>
<td>T bit can be loaded by direct address of PS, or from console.</td>
</tr>
<tr>
<td>B</td>
<td>Not implemented.</td>
<td>First instruction after RTI is guaranteed to be executed.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td>E</td>
<td>1. PC contains odd address</td>
<td>PC unincremented</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td></td>
<td>2. PC contains an address in non-existent memory</td>
<td>PC incremented</td>
<td>PC unincremented</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td></td>
<td>3. Register contains odd address &amp; instruction mode 2</td>
<td>Register unincremented</td>
<td>Register incremented</td>
<td>Same as 11/05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Register contains address in non-existent memory &amp; instruction mode 2</td>
<td>Register incremented</td>
<td>Register incremented</td>
<td>Register unincremented.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Interrupt service routine.</td>
<td>The first instruction will not be executed if another interrupt occurs at a higher priority.</td>
<td>Same as 11/05</td>
<td>Same as 11/05</td>
</tr>
</tbody>
</table>
6. Priority order of traps & interrupts

<table>
<thead>
<tr>
<th>Condition</th>
<th>11/05 &amp; 11/10</th>
<th>11/35 &amp; 11/40</th>
<th>11/04</th>
<th>11/34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Halt inst</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>odd address</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Overflow</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trap Inst.</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace Trap</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Fail</td>
<td>Same as 11/05</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Overflow (red)</td>
<td>Same as 11/04</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Fail</td>
<td>Same as 11/04</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Overflow (yellow)</td>
<td>Same as 11/04</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halt from console</td>
<td>Same as 11/04</td>
<td>Same as 11/04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. MISCELLANEOUS

A. Swab and V Bit

<table>
<thead>
<tr>
<th>Condition</th>
<th>V Bit is cleared</th>
<th>Same as 11/05</th>
<th>Same as 11/05</th>
<th>Same as 11/05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Set</td>
<td>Basic set &amp; mark</td>
<td>Basic set &amp; ATT</td>
<td>Basic set &amp; mark</td>
<td>Basic set &amp; mark</td>
</tr>
<tr>
<td></td>
<td>RTI, SOR, SXT, XOR</td>
<td>RTI, SOR, SXT, XOR</td>
<td>SXT, XOR, MUL, DIV</td>
<td>SXT, XOR, MUL, DIV</td>
</tr>
<tr>
<td></td>
<td>EIS adds: MUL, DIV</td>
<td>EIS adds: MUL, DIV</td>
<td>EIS adds: MADD, FSUB, F_MUL, F_DIV</td>
<td>EIS adds: MADD, FSUB, F_MUL, F_DIV</td>
</tr>
<tr>
<td></td>
<td>ASH, ASHC</td>
<td>ASH, ASHC</td>
<td>ASH, ASHC, MADD, F_SUB, M_MUL, M_DIV</td>
<td>ASH, ASHC, MADD, F_SUB, M_MUL, M_DIV</td>
</tr>
<tr>
<td></td>
<td>EIS adds: MADD, F_SUB, M_MUL, M_DIV</td>
<td>EIS adds: MADD, F_SUB, M_MUL, M_DIV</td>
<td>EIS adds: MADD, F_SUB, M_MUL, M_DIV</td>
<td>EIS adds: MADD, F_SUB, M_MUL, M_DIV</td>
</tr>
<tr>
<td></td>
<td>RTII=2 adds: M_TP1</td>
<td>RTII=2 adds: M_TP1</td>
<td>RTII=2 adds: M_TP1</td>
<td>RTII=2 adds: M_TP1</td>
</tr>
<tr>
<td></td>
<td>M_TP1, M_TP2</td>
<td>M_TP1, M_TP2</td>
<td>M_TP1, M_TP2</td>
<td>M_TP1, M_TP2</td>
</tr>
</tbody>
</table>

B. Instruction Set

C. Memory management violation during a trap sequence

<table>
<thead>
<tr>
<th>Condition</th>
<th>Does not apply</th>
<th>Does not apply</th>
<th>Does not apply</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If a set, not, violation occurs between the first and second push down of the stack during a trap sequence the status of the CPU before the violation is placed as the PS on the kernel stack.</td>
<td>If a set, not, violation occurs between the first and second push down of the stack during a trap sequence the status of the vector +2 of the original trap is placed as the PS on the kernel stack.</td>
<td>If a set, not, violation occurs between the first and second push down of the stack during a trap sequence the status of the vector +2 of the original trap is placed as the PS on the kernel stack.</td>
</tr>
</tbody>
</table>
2.2 GENERAL REGISTERS AND ADDRESSING MODES

Data manipulated by the PDP-11 is accessed directly, indirectly, or indexed via one of eight general purpose registers contained in the processor. The registers are generally referred to as R0 through R7. Although all eight registers are treated identically by the CPU, two of these registers have been chosen for specific functions while the remaining six may be used for general purpose. The following paragraphs describe the registers.

2.2.1 Register R7 (Program Counter)

R7 is used as a pointer to the instruction sequence and can be thought of as a traditional program counter (PC). When used in conjunction with certain addressing modes specified by the instruction (see Paragraph 7.2), R7 can also be used to point to immediate data, indirect addresses, or indexed addresses in line with the instruction sequence.

2.2.2 Register R6 (Stack Pointer)

R6 is used as a pointer to a list or "pushdown stack" in main memory where the machine state (as indicated by the program counter and processor status word) is saved during processor interrupts or subroutine calls. The "stack" is also often used to store temporary data or pass arguments between routines.

2.2.3 Registers R5 Through R0

Registers R5 through R0 have no assigned use and may be assigned for general purpose applications.

2.3 INSTRUCTION OPERANDS

Data manipulated by instructions is accessed via registers in conjunction with an addressing mode. The specific addressing mode and register form a field and are contained in the instruction along with the op code. Single operand instructions require only a destination address field while double operand instruction requires both a source and destination address field. Figure 2-1 shows the formats of these two types of instructions.

2.4 ADDRESSING MODES

PDP-11 instructions allow six bits for each operand address. Three of
these bits are designated Rn (see Figure 2-1), and specify one of the general registers; the other three bits define one of the addressing modes. The addressing mode dictates how the specified register is used in obtaining the desired operand.

In the three-bit address mode field, the two most significant bits control register use while the least significant bit indicates a direct or indirect address. Table 2-2 lists the various address modes.

2.4.1 Direct Addressing Modes

The direct modes (register, autoincrement, autodecrement, and index) are employed when the least significant bit in the address mode field is zero (modes 0, 2, 4, or 6).

The autoincrement direct and autodecrement direct modes allow extremely flexible addressing of structured data, character strings, data arrays, etc.
** Single-Operand Instruction

<table>
<thead>
<tr>
<th>OP CODE</th>
<th>MODE</th>
<th>( R_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11 10 9 8</td>
</tr>
</tbody>
</table>

** Specifies direct or indirect address
*** Specifies how register will be used
**** Specifies one of 8 general purpose registers

---

** Double-Operand Instruction

<table>
<thead>
<tr>
<th>OP CODE</th>
<th>MODE</th>
<th>( R_n )</th>
<th>MODE</th>
<th>( R_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11 10 9 8</td>
<td>6</td>
<td>5 4 3 2 0</td>
</tr>
</tbody>
</table>

** Direct/Deferred bit for source and destination address
*** Specifies how selected registers are to be used
**** Specifies a general register

---

**Figure 2-1 Addressing Mode Instruction Formats**
<table>
<thead>
<tr>
<th>Binary Code</th>
<th>Name</th>
<th>Assembler Syntax</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Register</td>
<td>Rn</td>
<td>Register contains operand.</td>
</tr>
<tr>
<td>010</td>
<td>Autoincrement</td>
<td>(Rn)+</td>
<td>Register contains address of operand. Register contents incremented after reference.</td>
</tr>
<tr>
<td>100</td>
<td>Autodecrement</td>
<td>-(Rn)</td>
<td>Register contents decremented before reference. Register contains address of operand.</td>
</tr>
<tr>
<td>110</td>
<td>Index</td>
<td>X(Rn)</td>
<td>Value X (stored in a word following the instruction) is added to (Rn) to produce address of operand. Neither X nor (Rn) are modified.</td>
</tr>
</tbody>
</table>

**INDIRECT MODES**

<table>
<thead>
<tr>
<th>Binary Code</th>
<th>Name</th>
<th>Assembler Syntax</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Register Indirect</td>
<td>@Rn</td>
<td>Register contains the address of the operand. or (Rn)</td>
</tr>
<tr>
<td>011</td>
<td>Autoincrement Indirect</td>
<td>@(Rn)+</td>
<td>Register is first used as a pointer to a word containing the address of the operand, then incremented (always by two, even for byte instructions).</td>
</tr>
<tr>
<td>101</td>
<td>Autodecrement Indirect</td>
<td>@-(Rn)</td>
<td>Register is decremented (always by two; even for byte instructions) and then used as a pointer to a word containing the address of the operand.</td>
</tr>
<tr>
<td>111</td>
<td>Index Indirect</td>
<td>@X(Rn)</td>
<td>Value X (stored in a word following the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) are modified.</td>
</tr>
</tbody>
</table>
### PC ADDRESSING

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>Immediate</td>
<td>#n</td>
<td>Operand follows instruction.</td>
</tr>
<tr>
<td>011</td>
<td>Absolute</td>
<td>@#A</td>
<td>Absolute address follows instruction.</td>
</tr>
<tr>
<td>110</td>
<td>Relative</td>
<td>A</td>
<td>Address of A, relative to the instruction, follows the instruction.</td>
</tr>
<tr>
<td>111</td>
<td>Relative Deferred</td>
<td>@A</td>
<td>Address of location containing address of A, relative to the instruction, follows the instruction.</td>
</tr>
</tbody>
</table>

Rn=Register
X, n, A=next program counter (PC) word (constant)
The index direct mode is utilized where the general register contains a base address and the operand is indexed or displaced a specific distance from the base.

2.4.2 Indirect Addressing Modes

The indirect addressing modes (register indirect, autoincrement indirect, autodecrement indirect, and index indirect) are employed when the least significant bit of the address mode field is a 1 (modes 1, 3, 5, 7).

The autoincrement indirect and autodecrement indirect modes can be used to access data via lists of pointers (non-structured data or program transfer addresses).

The index indirect mode is utilized where the general register contains a base address and the address of the operand is indexed or displaced a specific distance from the base.

2.4.3 Program Counter (PC) Addressing

Several extremely useful forms of addressing are available when R7 (Program Counter) is used as the general register.

The autoincrement mode causes the operand (or its address in indirect mode) to follow the instruction. This is commonly referred to as immediate (direct mode) or absolute (indirect mode).

The index mode causes the operand (or its address in indirect mode) to be located a specific distance from the PC. Code written using relocatable addresses can be loaded and run in any part of memory.

2.5 INSTRUCTION SET

The instruction set in the PDP-11/34 processing systems is divided into the following groups:

- Single operand (see Table 2-3)
- Double operand (see Table 2-4)
- Program Control (see Table 2-5)
- Miscellaneous Group (see Table 2-6)
- Condition Code Operator (see Table 2-7)

The tables referenced above list the instructions within each group and specify:

- Mnemonic of the instruction
1. Binary opcode
2. Operation (uses symbolic notation)
3. Effect on the condition codes, and
4. Description with comments and examples, as applicable.

Figure 2-2 lists the instruction formats for the opcodes within each group.
1. Single Operand Group (CLR, CLRB, COM, COMB, INC, INCB, DEC, DECB, NEG, NEG, ADC, ADCB, SBC, SBCB, TST, TSTB, ROL, RORB, ROL, ROLB, ASR, ASRB, ASL, ASLB, SXT, SWAB)

2. Double Operand Group (BIT, BITB, BIC, BICB, UIS, UISB, ADD, SUB)

3. Program Control Group
   a. Branch (all branch instructions)

b. (ASH, ASHC)
d. Subroutine Return (RTS)

\[ \phi \quad \phi \quad \phi \quad 2 \quad \phi \quad 3 \quad 2 \quad \phi \]

e. MARK

\[ \text{OP Code} \quad \text{Offset} \]

15 \quad 6 \quad 5

g. MOV

\[ \text{OP Code} \]

15

h. Trans (break BPT, IOT, EMT, TRAP, RTI, RTT)

\[ \text{OP Code} \]

15

4. Condition Code Operators (all condition code instructions)

\[ \phi \quad \phi \quad \phi \quad 2 \quad 1 \quad N \quad Z \quad V \quad C \quad 4 \quad 3 \quad 2 \quad 1 \quad \phi \]

5. Miscellaneous Group (HALT, WAIT, RESET)

\[ \text{OP Code} \]

15

(MFPI, MFPD, MTPI, MTPD)
2.6 INSTRUCTION EXECUTION TIME

The execution time for an instruction depends on the instruction itself, the modes of addressing used, and the type of memory being referenced. In the most general case, the Instruction Execution Time is the sum of a Source Address Time, a Destination Address Time, and an Execute, Fetch Time.

$$\text{Instr Time} = \text{SRC Time} + \text{DST Time} + \text{EF Time}$$

Some of the instructions require only some of these times, and are so noted. All Timing Information is in microseconds, unless otherwise noted. Times are typical; processor timing can vary +/- 10%.

2.6.1 Basic Instruction Set Timing

Table 2-8 lists the PDP-11/34 instruction set together with the timing characteristics and memory cycles required.

Double Operand all instructions

$$\text{Instr Time} = \text{SRC Time} + \text{DST Time} + \text{EF Time}$$

Single Operand all instructions

$$\text{Instr Time} = \text{DST Time} + \text{EF Time}$$

Branch, Jump, Control, Trap, and Misc. All Instructions:

$$\text{Instr Time} = \text{EF Time}$$

Notes:

1. The times specified generally apply to Word instructions. Even Byte and Odd Byte instructions have the same times in all cases.
2. Timing is given without regard for NPR or BR servicing.
3. If the Memory Management is enabled, instruction execution times increase by 0.12 usec for each memory cycle used.
4. All timing is based on memory with the following performance characteristics.

**CORE**
- Access time: 510 nsec
- Cycle time: 1.0 usec

**MOS**
- Access time: 635 nsec
- Cycle time: 920 nsec
2.6.2 Latency

Interrupts (BR requests) are acknowledged at the end of the current instruction. For a typical instruction, with an instruction execution time of 4 usec, the average time to request acknowledgement would be 2 usec.

Interrupt service time, which is the time from BR acknowledgement to the first subroutine instruction, is 7.32 usec, max. for core, 7.7 usec for MAS.

NPR (DMA) latency, which is the time from request to bus mastership for the first NPR device, is 2.5 usec, max.
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>0050DD*</td>
<td>(dst) → 0</td>
<td>N: cleared</td>
<td>Contents of specified destination are replaced with zeroes.</td>
</tr>
<tr>
<td>CLRB</td>
<td>1050DD</td>
<td></td>
<td>Z: set</td>
<td></td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td></td>
<td>V: cleared</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: cleared</td>
<td></td>
</tr>
<tr>
<td>COM</td>
<td>0051DD</td>
<td>(dst) ← n (dst)</td>
<td>N: set if most significant bit of result is 0</td>
<td>Replaces the contents of the destination address by their logical complement (each bit equal to 0 set and each bit equal to 1 cleared).</td>
</tr>
<tr>
<td>COMB</td>
<td>1051DD</td>
<td></td>
<td>Z: set if result is 0</td>
<td></td>
</tr>
<tr>
<td>Complement</td>
<td></td>
<td></td>
<td>V: cleared</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: set</td>
<td></td>
</tr>
<tr>
<td>INC</td>
<td>0052DD</td>
<td>(dst) ← (dst) + 1</td>
<td>N: set if result is less than 0</td>
<td>Add 1 to the contents of the destination.</td>
</tr>
<tr>
<td>INCB</td>
<td>1052DD</td>
<td></td>
<td>Z: set if result is 0</td>
<td></td>
</tr>
<tr>
<td>Increment</td>
<td></td>
<td></td>
<td>V: set if (dst) was 077777</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: not effected</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>0053DD</td>
<td>(dst) ← (dst) - 1</td>
<td>N: set if result is less than 0</td>
<td>Subtract 1 from the contents of the destination.</td>
</tr>
<tr>
<td>DECB</td>
<td>1053DD</td>
<td></td>
<td>Z: set if result is 0</td>
<td></td>
</tr>
<tr>
<td>Decrement</td>
<td></td>
<td></td>
<td>V: set if (dst) was 100000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: not effected</td>
<td></td>
</tr>
<tr>
<td>NEG</td>
<td>0054DD</td>
<td>(dst) ← -(dst)</td>
<td>N: set if result is less than 0</td>
<td>Replaces the contents of the destination address by its 2's complement. Note that 100000 is replaced by itself.</td>
</tr>
<tr>
<td>NEGB</td>
<td>1054DD</td>
<td></td>
<td>Z: set if result is 0</td>
<td></td>
</tr>
<tr>
<td>Negate</td>
<td></td>
<td></td>
<td>V: set if result is 100000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: cleared if result is 0</td>
<td></td>
</tr>
<tr>
<td>ADC</td>
<td>0055DD</td>
<td>(dst) ← (dst) + C</td>
<td>N: set if result is less than 0</td>
<td>Adds the contents of the C-bit into the destination. This permits the carry from the addition of the low order words/bytes to be carried into the high order result.</td>
</tr>
<tr>
<td>ADCB</td>
<td>1055DD</td>
<td></td>
<td>Z: set if result is 0</td>
<td></td>
</tr>
<tr>
<td>Add Carry</td>
<td></td>
<td></td>
<td>V: set if (dst) is 077777 and C is 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: set if (dst) is 177777 and C is 1</td>
<td></td>
</tr>
<tr>
<td>Mnemonic/ Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------</td>
<td>--------------------</td>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| SBC                        | 0056DD    | (dst) ← (dst) - C  | N: set if result is less than 0  
Z: set if result is 0  
V: set if (dst) was 100000  
C: cleared if (dst) is 0 and C is 1 | Subtracts the contents of the C-bit from the destination. This permits the carry from the subtraction of the low order words/bytes to be subtracted from the high order part of the result. |
| SBCB                       | 1056DD    |                    |                                                      |                                                                             |
| Subtract Carry             |           |                    |                                                      |                                                                             |
| TST                        | 0057DD    | (dst) ← (dst)      | N: set if result is less than 0  
Z: set if result is 0  
V: cleared  
C: cleared | Sets the condition codes N and Z according to the contents of the destination address. |
| TSTB                       | 1057DD    |                    |                                                      |                                                                             |
| Test                       |           |                    |                                                      |                                                                             |
| ROR                        | 0060DD    | (dst) ← (dst)      | N: set if high order bit of the result is set  
Z: set if all bits of result are 0  
V: loaded with the exclusive-OR of the N-bit and the C-bit as set by ROR | Rotates all bits of the destination right one place. The low order bit is loaded into the C-bit and the previous contents of the C-bit are loaded into the high order bit of the destination. |
| RORB                       |           | rotate right one place. |                                                      |                                                                             |
| Rotate Right               |           |                    |                                                      |                                                                             |
| ROL                        | 0061DD    | (dst) ← (dst)      | N: set if the high order bit of the result word is set (result < 0); cleared otherwise  
Z: set if all bits of the result word = 0; cleared otherwise  
V: loaded with the exclusive-OR of the N-bit and C-bit (as set by the completion of the rotate operation)  
C: loaded with the high order bit of the destination | Rotate all bits of the destination left one place. The high order bit is loaded into the C-bit of the status word and the previous contents of the C-bit are loaded into the low order bit of the destination. |
<p>| ROLB                       | 1061DD    | rotate left one place. |                                                      |                                                                             |
| Rotate Left                |           |                    |                                                      |                                                                             |</p>
<table>
<thead>
<tr>
<th>Mnemonic/Operation</th>
<th>OP Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR</td>
<td>0062DD</td>
<td>Shifts all bits of the destination right one place. The high order bit is replicated. The C-bit is loaded from the low order bit of the destination. ASR performs signed division of the destination by two.</td>
</tr>
<tr>
<td>ARSB</td>
<td>1062DD</td>
<td></td>
</tr>
<tr>
<td>Arithmetic Shift Right</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASL</td>
<td>0063DD</td>
<td>Shifts all bits of the destination left one place. The low order bit is loaded with a 0. The C-bit of the status word is loaded from the high order bit of the destination. ASL performs a signed multiplication of the destination by 2 with overflow indication.</td>
</tr>
<tr>
<td>ASLB</td>
<td>1063DD</td>
<td></td>
</tr>
<tr>
<td>Arithmetic Shift Left</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mnemonic/Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>ASH</td>
<td>072RSS</td>
<td>R ← R Shifted</td>
</tr>
<tr>
<td>Arithmetic Shift</td>
<td></td>
<td>Arithmetically NN places to right or left Where NN = (src)</td>
</tr>
<tr>
<td>ASHC</td>
<td>073RSS</td>
<td>R, Rv1 ← R, Rv1</td>
</tr>
<tr>
<td>Arithmetic Shift Combined</td>
<td></td>
<td>The double word is shifted NN places to the right or left, where NN = (src)</td>
</tr>
<tr>
<td>SXT</td>
<td>0067DD</td>
<td>(dst) ← 0 if N bit is clear</td>
</tr>
<tr>
<td>Sign Extend</td>
<td></td>
<td>(dst) ← −1 N bit is set</td>
</tr>
<tr>
<td>SWAB</td>
<td>0003DD</td>
<td>Byte 1/Byte 0</td>
</tr>
<tr>
<td>Swap Byte</td>
<td></td>
<td>Byte 0/Byte 1</td>
</tr>
<tr>
<td>Mnemonic/ Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
| MOV                        | 01SSDD* | (dst) ← (src) ‡ | N: set if (src) < 0, cleared otherwise  
                             |         |            | Z: set if (src) = 0, cleared otherwise  
                             | 11SSDD  |            | V: cleared  
                             |         |            | C: not effected  | Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The source operand is not effected. Byte: Same as MOV. The MOVB to a resistor (unique among byte instructions) extends the most significant bit of the low order byte (sign extension). Otherwise MOVB operates on bytes exactly as MOV operates on words. |
| MOVB                       | 02SSDD  | (src) - (dst) | N: set if result < 0, cleared otherwise  
                             |         |            | Z: set if result = 0; cleared otherwise  
                             | 12SSDD  | [in detail.] (src) + \( \sim \) (dst) + 1 | V: set if there was arithmetic overflow, i.e., operands were of opposite signs and the sign of the destination was the same as the sign of the result; cleared otherwise  
<pre><code>                         |         |            | C: cleared if there was a carry from the most significant bit of the result; set otherwise  | Compares the source and destination operands and sets the condition codes which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Note that unlike the subtract instruction the order of operation is (src) - (dst), not (dst) - (src). |
</code></pre>
<table>
<thead>
<tr>
<th>Mnemonic/Operation</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
</table>
| BIT                | 03SSDD  | $(src) \land (dst)$ | N: set if high order bit of result set; cleared otherwise  
Z: set if result = 0; cleared otherwise
V: cleared
C: not effected | Performs logical AND comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination operands are effected. The BIT instruction may be used to test whether any of the corresponding bits that are set in the destination are clear in the source. |
| BITB               | 13SSDD  |              |                 |             |
| Bit Test           |         |            |                 |             |
| BIC                | 04SSDD  | $(dst) \rightarrow \neg (src)$ | N: set if high order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not effected | Clears each bit in the destination that corresponds to a set bit in the source. The original contents of the destination are lost. The contents of the source are uneffected. |
| BICB               | 14SSDD  | $(dst) \land (dst)$ | N: set if high order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not effected | Performs inclusive-OR operation between the source and destination operands and leaves the result at the destination address: i.e., corresponding bits set in the destination. The contents of the destination are lost. |
| BIS                | 05SSDD  | $(dst) \rightarrow (src)$ | N: set if high order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not effected | Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not effected. Two's complement addition is performed. |
| BISB               | 15SSDD  | $(dst) \land (dst)$ | N: set if high order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not effected |             |
<table>
<thead>
<tr>
<th>Mnemonic/Operation</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD (Cont)</td>
<td></td>
<td>(dst) ← (dst) - (src) in detail, (dst) + ~ (src) + 1 (dst)</td>
<td>V: set if there was arithmetic overflow as a result of the operation; that is both operands were of the same sign and the result was of the opposite sign; cleared otherwise C: set if there was a carry from the most significant bit of the result; cleared otherwise</td>
<td>Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double precision arithmetic, the C-bit, when set, indicates a borrow.</td>
</tr>
<tr>
<td>SUB Subtract</td>
<td>16SSDD</td>
<td></td>
<td>N: set if result &lt; 0; cleared otherwise Z: set if result = 0; cleared otherwise V: set if there was arithmetic overflow as a result of the operation, i.e., if operands were of opposite signs and the sign of the source was the same as the sign of the result; cleared otherwise C: cleared if there was a carry from the most significant bit of the result; set otherwise</td>
<td></td>
</tr>
</tbody>
</table>

* SS = source (address mode and register)  
† (src) = source contents
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUL Multiply</td>
<td>070RSS</td>
<td>R, Rv1 ← R x(src)</td>
<td>N: set if product is &lt;0; cleared otherwise&lt;br&gt;Z: set if product is 0; cleared otherwise&lt;br&gt;V: cleared&lt;br&gt;C: set if the result is less than $-2^{15}$ or greater than or equal to $2^{15}$ -1.</td>
<td>The contents of the destination register and source taken as two's complement integers are multiplied and stored in the destination register and the succeeding register (if R is even). If R is odd only the low order product is stored. Assembler syntax is: MUL S,R. (Note that the actual destination is R,Rv1 which reduces to just R when R is odd.) (See Paragraph 1.7 for example).</td>
</tr>
<tr>
<td>DIV Divide</td>
<td>071RSS</td>
<td>R, Rv1 ← R, Rv1 (src)</td>
<td>N: set if quotient &lt;0; cleared otherwise&lt;br&gt;Z: set if quotient = 0; cleared otherwise&lt;br&gt;V: set if source = 0 or if the absolute value of the register is larger than the absolute value of the source. (In this case the instruction is aborted because the quotient would exceed 15 bits.)&lt;br&gt;C: set if divide 0 attempted; cleared otherwise</td>
<td>The 32-bit two's complement integer in R and Rv1 is divided by the source operand. The quotient is left in R; the remainder is of the same sign as the dividend. R must be even. (See Paragraph 1.7 for example)</td>
</tr>
<tr>
<td>XOR</td>
<td>074RDD</td>
<td>(dst) ← Rv (dst)</td>
<td>N: set if the result &lt;0; cleared otherwise&lt;br&gt;Z: set if result = 0; cleared otherwise&lt;br&gt;V: cleared&lt;br&gt;C: unaffected</td>
<td>The exclusive OR of the register and destination operand is stored in the destination address. Contents of register are unaffected. Assembler format is XOR R,D</td>
</tr>
</tbody>
</table>
### Table 2-5
Program Control Instructions

<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>000400</td>
<td>PC ← PC + (2 X offset)</td>
<td>Uneffected</td>
<td>Provides a way of transferring program control within a range of -128 to +127 words with a one word instruction. It is an unconditional branch.</td>
</tr>
<tr>
<td>Branch</td>
<td>xxx†</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td>001000</td>
<td>PC ← PC + (2 X offset) if Z = 0</td>
<td>Uneffected</td>
<td>Tests the state of the Z-bit and causes a branch if the Z-bit is is clear. BNE is the complementary operation to BEQ. It is used to test inequality following a CMP, to test that some bits set in the destination were also in the source, following a BIT, and generally, to test that the result of the previous operation was not 0.</td>
</tr>
<tr>
<td>Branch if not equal</td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEQ</td>
<td>001400</td>
<td>PC ← PC + (2 X offset) if Z = 1</td>
<td>Uneffected</td>
<td>Tests the state of the Z-bit and causes a branch if Z is set. As an example, it is used to test equality following a CMP operation. to test that no bits set in the destination were also set in the source following a BIT operation. and generally, to test that the result of the previous operation was 0.</td>
</tr>
<tr>
<td>Branch if equal</td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGE</td>
<td>002000</td>
<td>PC ← PC + (2 X offset) if N ∨ V = 0</td>
<td>Uneffected</td>
<td>Causes a branch if N and V are either both clear or both set. BGE is the complementary operation to BLT. Thus, BGE always causes a branch when it follows an operation that caused addition to two positive numbers. BGE also causes a branch on a 0 result.</td>
</tr>
<tr>
<td>Branch if greater than or equal</td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mnemonic/ Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>BLT</strong> Branch if less than</td>
<td>002400</td>
<td>PC ← PC + (2 × offset) if ( N \lor V = 1 )</td>
<td>Unaffected</td>
<td>Causes a branch if the exclusive-OR of the ( N ) and ( V )-bits are 1. Thus, BLT always branches following an operation that added two negative numbers, even if overflow occurred. In particular, BLT always causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BLT never causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT does not cause a branch if the result of the previous operation was 0 (without overflow).</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BGT</strong> Branch if greater than</td>
<td>003000</td>
<td>PC ← PC + (2 × offset) if ( Z \lor (N \lor V) = 0 )</td>
<td>Unaffected</td>
<td>Operation of BGT is similar to BGE, except BGT does not cause a branch on a 0 result.</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BLE</strong> Branch if less than or equal to</td>
<td>003400</td>
<td>PC ← PC + (2 × offset) if ( Z \land (N \land V) = 1 )</td>
<td>Unaffected</td>
<td>Operation is similar to BLT but in addition will cause a branch if the result of the previous operation was 0.</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BPL</strong> Branch if plus</td>
<td>100000</td>
<td>PC ← PC + (2 × offset) if ( N = 0 )</td>
<td>Unaffected</td>
<td>Tests the state of the ( N )-bit and causes a branch if ( N ) is clear. BPL is the complementary operation of BMI.</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BMI</strong> Branch if minus</td>
<td>100400</td>
<td>PC ← PC + (7 × offset) if ( N = 1 )</td>
<td>Unaffected</td>
<td>Tests the state of the ( N )-bit and causes a branch if ( N ) is set. It is used to test the sign (most significant bit) of the result of the previous operation.</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mnemonic/Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>-----------------------------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BHI Branch if higher</td>
<td>101000</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $C = 0$</td>
<td>Unaffected</td>
<td>Causes a branch if the previous operation causes neither a carry nor a 0 result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.</td>
</tr>
<tr>
<td>BLOOS Branch if lower or same</td>
<td>101400</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $C \lor Z = 1$</td>
<td>Unaffected</td>
<td>Causes a branch if the previous operation caused either a carry or a 0 result. BLOOS is the complementary operation to BHI. The branch occurs in comparison operations as long as the source is equal to or has a lower unsigned value than the destination. Comparison of unsigned values with the CMP instruction to be tested for &quot;higher or same&quot; and &quot;higher&quot; by a simple test of the C-bit.</td>
</tr>
<tr>
<td>BVC Branch if V-bit clear</td>
<td>102000</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $V = 0$</td>
<td>Unaffected</td>
<td>Tests the state of the V-bit and causes a branch if the V-bit is clear. BVC is a complementary operation to BVS.</td>
</tr>
<tr>
<td>BVS Branch if V-bit set</td>
<td>102400</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $V = 1$</td>
<td>Unaffected</td>
<td>Tests the state of V-bit (overflow) and causes a branch if the V-bit is set. BVS is used to detect arithmetic overflow in the previous operation.</td>
</tr>
<tr>
<td>BCC BHIS Branch if carry clear</td>
<td>103000</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $C = 0$</td>
<td>Unaffected</td>
<td>Tests the state of the C-bit and causes a branch if C is clear. BCC is the complementary operation to BCS.</td>
</tr>
<tr>
<td>BCS Branch if carry set</td>
<td>103400</td>
<td>$PC \leftarrow PC + (2 \times \text{offset})$ if $C = 1$</td>
<td>Unaffected</td>
<td>Tests the state of the C-bit and causes a branch if C is set. It is used to test for a carry in the result of a previous operation.</td>
</tr>
<tr>
<td>Mnemonic/ Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>JMP Jump</td>
<td>0001DD</td>
<td>PC ← (dst)</td>
<td>Not affected</td>
<td><strong>JMP</strong> provides more flexible program branching than provided with the branch instruction. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes, with the exception of register mode 0. Execution of a jump with mode 0 will cause an illegal instruction condition. (Program control cannot be transferred to a register.) Register deferred mode is legal and will cause program control to be transferred to the address held in the specified register. Note that instructions are word data and must therefore be fetched from an even numbered address. A boundary error trap condition will result when the processor attempts to fetch an instruction from an odd address.</td>
</tr>
</tbody>
</table>

* DD = destination (address mode and register)
† (dst) = destination contents
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSR</td>
<td>004RBD</td>
<td>(mp) → (dst) (mp is an interrupt processor register)</td>
<td>Unaffected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 (SP) → reg (push reg contents onto processor stack)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>push PC → reg (push PC)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>JSR: this location following JSR; this address PC; (mp) now put in (reg)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>JSR; the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions.</td>
<td></td>
</tr>
</tbody>
</table>

**JSR PC**: JSR PC is a special case of the JSR instruction, where the return address is the address of the next instruction. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions.

**RTS**

Returns from subroutine calls that transmit parameters. The contents of register into PC and pops the top element of the processor stack into the specified register. Parameters are popped from the processor stack and then passed on to the PC. The condition codes remain unchanged.

**Return from subroutine**

Parameters are popped from the processor stack and then passed on to the PC. The condition codes remain unchanged.

---

In execution of the JSR, the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. JSR PC is a special case of the JSR instruction, where the return address is the address of the next instruction. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. Parameters are popped from the processor stack and then passed on to the PC. The condition codes remain unchanged.

In execution of the JSR, the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. JSR PC is a special case of the JSR instruction, where the return address is the address of the next instruction. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. Parameters are popped from the processor stack and then passed on to the PC. The condition codes remain unchanged.

In execution of the JSR, the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. JSR PC is a special case of the JSR instruction, where the return address is the address of the next instruction. The JSR instruction is executed in the normal manner, except that the linkage register contains the address of the next instruction, not the return address, as in other JSR instructions. Parameters are popped from the processor stack and then passed on to the PC. The condition codes remain unchanged.
<table>
<thead>
<tr>
<th>Mnemonic/Operation Time</th>
<th>Op Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARK</td>
<td>0BE4NN</td>
<td>SP ← SP + 2xnn, PC ← R5, R5 ← (SP)</td>
<td>Unaffected</td>
<td>Used as part of the standard PDP-11 subroutine return convention. MARK facilitates the stack clean up procedures involved in subroutine exit. Assembler format is: MARK N.</td>
</tr>
<tr>
<td>Example: MOV R5,-(SP)</td>
<td></td>
<td>place old R5 on stack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV P1,-(SP)</td>
<td></td>
<td>place N parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV P2,-(SP)</td>
<td></td>
<td>on the stack to be used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV PN,-(SP)</td>
<td></td>
<td>there by the subroutine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV #MARKN,-(SP)</td>
<td></td>
<td>MARK N on the stack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV SP,R5</td>
<td></td>
<td>set up address at Mark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR PC,SUB</td>
<td></td>
<td>N instruction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At this point the stack is as follows:

```
OLD R5
P1
PN
MARK N
OLD PC
```

And the program is at the address SUB which is the beginning of the subroutine. SUB: execution of the subroutine itself

RTS R5: the return begins; this causes the contents of R5 to be placed in the PC which then results in the execution of the instruction MARK N. The contents of old PC are placed in R5.

MARK N causes: (1) the stack pointer to be adjusted to point to the old R5 value; (2) the value now in R5 (the old PC) to be placed in the PC; and (3) contents of the old R5 to be popped into R5 thus completing the return from subroutine.
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>Op Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOB</td>
<td>077R00</td>
<td>R ← R - 1 if this result ≠ 0 then PC ← PC -(2 x offset)</td>
<td>Unaffected</td>
<td>The register is decremented. If it is not equal to 0, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a sixbit positive number. This instruction provides a fast, efficient method of loop control. Assembler syntax is: SOB R,A Where A is the address to which transfer is to be made if the decremented R is not equal to 0. Note that the SOB instruction can not be used to transfer control in the forward direction.</td>
</tr>
<tr>
<td>Mnemonic/ Instruction Time</td>
<td>OP Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Break-point Trap</td>
<td>000003</td>
<td>↓ (SP) ← PS ↓ (SP) ← PC PC ← (14) PS ← (16)</td>
<td>N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector</td>
<td>Performs a trap sequence with a trap vector address of 14. Used to call debugging aids. The user is cautioned against employing code 000003 in programs run under these debugging aids.</td>
</tr>
<tr>
<td>I/O Trap</td>
<td>000004</td>
<td>↓ (SP) ← PS ↓ (SP) ← PC PC ← (20) PS ← (22)</td>
<td>N: loaded from trap vector Z: loaded from trap vector C: loaded from trap vector</td>
<td>Performs a trap sequence with a trap vector address of 20. Used to call the I/O executive routine IOX in the paper-tape software system and for error reporting in the disk operating system.</td>
</tr>
<tr>
<td>Emulator Trap</td>
<td>104000</td>
<td>↓ (SP) ← PS ↓ (SP) ← PC PC ← (30) PS ← (32)</td>
<td>N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector</td>
<td>All operation codes from 104000 to 104377 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30: the new central processor status (PS) is taken from the word at address 32. CAUTION EMT is used frequently by DEC system software and is therefore not recommended for general use.</td>
</tr>
<tr>
<td>TRAP</td>
<td>104400 to 104777</td>
<td>↓ (SP) ← PS ↓ (SP) ← PC PC ← (34) PS ← (36)</td>
<td>N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector</td>
<td>Operation codes from 104400 to 104777 are TRAP instructions TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34. NOTE Since DEC software makes frequent use of EMT, the TRAP instruction is recommended for general use.</td>
</tr>
</tbody>
</table>

NOTE: Condition Codes are uneffected by these instructions

†xxxx = offset, 8 bits (0-7) of instruction format
R = register (linkage pointer)
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>Op Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTI</td>
<td>000002</td>
<td>PC ← (SP)†</td>
<td>N: loaded from processor stack</td>
<td>Used to exit from an interrupt or trap service routine. The PC and PSW are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSW ←(SP)†</td>
<td>Z: loaded from processor stack</td>
<td>restored (popped) from the processor stack. If the RTI sets the T bit in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>V: loaded from processor stack</td>
<td>the PSW, a trace trap will occur prior to executing the next instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: loaded from processor stack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000006</td>
<td>PC ←(SP)†</td>
<td>N: loaded from processor stack</td>
<td>This is the same as the RTI instruction except that it inhibits a trace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS ←(SP)†</td>
<td>Z: loaded from processor stack</td>
<td>trap, while RTI permits a trace trap. If a trace trap is pending, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>V: loaded from processor stack</td>
<td>first instruction after the RTT will be executed prior to the next “T”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: loaded from processor stack</td>
<td>trap. In the case of the RTI instruction the “T” trap will occur</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>immediately after the RTI.</td>
</tr>
<tr>
<td>Mnemonic/Instruction Time</td>
<td>Op Code</td>
<td>Operation</td>
<td>Condition Codes</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MFPI</td>
<td>0065SS</td>
<td>(temp) ← (src)</td>
<td>N: set if the source &lt; 0; otherwise cleared</td>
<td>This instruction pushes a word onto the current stack from an address in previous space. Processor Status (bits 13, 12). The source address is computed using the current registers and memory map.</td>
</tr>
<tr>
<td>MFPD</td>
<td>1065SS</td>
<td>↓ (SP) ← (temp)</td>
<td>Z: set if the source = 0; otherwise cleared</td>
<td></td>
</tr>
<tr>
<td>MTPI</td>
<td>0066SS</td>
<td>(temp) ← (SP)†</td>
<td>V: cleared</td>
<td></td>
</tr>
<tr>
<td>MTPD</td>
<td>1066SS</td>
<td>(dst) ← (temp)</td>
<td>C: unaffected</td>
<td></td>
</tr>
<tr>
<td>MFPS</td>
<td>1067DD</td>
<td>(DST) ← PSW</td>
<td>N: set if the source &lt; 0; otherwise cleared</td>
<td>The 8-bit content of the PSW are moved to the effective destination if destination is mode 0. PSW bit 7 is sign extended through upper byte of the register. The destination operand is treated as a byte address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DST Lower, 8 Bits</td>
<td>Z: set if PSW &lt; 0:7 = 0; otherwise cleared</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>V: cleared</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C: not affected</td>
<td></td>
</tr>
<tr>
<td>MTPS</td>
<td>1064SS</td>
<td>PSW ← (SRC)</td>
<td>Set according to effective SRC operand Ø-3,</td>
<td></td>
</tr>
</tbody>
</table>

* Since there is no hardware to prevent the execution of these instructions in user mode, it is necessary for the system software to prevent a reference to the PSW address by a user.
<table>
<thead>
<tr>
<th>Mnemonic/ Instruction Time</th>
<th>OP Code</th>
<th>Operation</th>
<th>Condition Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALT</td>
<td>000000</td>
<td>Not effected</td>
<td></td>
<td>Causes the processor operation to cease. The console is given control of the processor. The console data lights display the address of the HALT instruction plus two. Transfers on the Unibus are terminated immediately. The PC points to the next instruction to be executed. Pressing the CON key on the console causes processor operation to resume. No INIT signal is given.</td>
</tr>
<tr>
<td>WAIT</td>
<td>000001</td>
<td>Not effected</td>
<td></td>
<td>Provides a way for the processor to relinquish use of the bus while it waits for an external interrupt. Having been given a WAIT command, the processor will not compete for bus by fetching instructions or operands from memory. This permits higher transfer rates between device and memory, since no processor induced latches will be encountered by bus requests from the device. In WAIT, as in all instructions, the PC points to the next instruction following the WAIT operation. Thus, when an interrupt causes the PC and PS to be pushed onto the stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction following the WAIT.</td>
</tr>
<tr>
<td>RESET</td>
<td>000005</td>
<td>PC (SP) PSW (SP)</td>
<td>Not effected</td>
<td>Sends INIT on the Unibus for 100 ms. All devices on the Unit are reset to their state at power-up.</td>
</tr>
</tbody>
</table>
Table 2-7
Condition Code Operators

<table>
<thead>
<tr>
<th>Mnemonic/ Instruction time</th>
<th>Op Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC</td>
<td>000241</td>
<td>Set and clear condition code bits. Selectable combination of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator, i.e., set the bit specified by bit 0,1,2, or 3 if bit 4 is a 1, clear corresponding bits if bit 4=0.</td>
</tr>
<tr>
<td>CLZ</td>
<td>000242</td>
<td></td>
</tr>
<tr>
<td>CLN</td>
<td>000244</td>
<td></td>
</tr>
<tr>
<td>CLV</td>
<td>000250</td>
<td></td>
</tr>
<tr>
<td>Set all CCs</td>
<td>000261</td>
<td></td>
</tr>
<tr>
<td>Clear all CCs</td>
<td>000262</td>
<td></td>
</tr>
<tr>
<td>Clear V and C</td>
<td>000264</td>
<td></td>
</tr>
<tr>
<td>No operation</td>
<td>000270</td>
<td></td>
</tr>
<tr>
<td>No operation</td>
<td>000277</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000257</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000243</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000240</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000260</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2-8
Instruction Set Timing

#### I. SOURCE ADDRESS TIME

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Source Mode</th>
<th>Memory Cycles</th>
<th>Core (MM11-DP)</th>
<th>MOS (MS11-JP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Operand</td>
<td>0</td>
<td>0</td>
<td>0.00 µsec</td>
<td>0.00 µsec</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1.13</td>
<td>1.26</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1.33</td>
<td>1.46</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2.37</td>
<td>2.62</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>1.28</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>2.57</td>
<td>2.82</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>2.57</td>
<td>2.82</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>3.80</td>
<td>4.18</td>
</tr>
</tbody>
</table>

#### II. DESTINATION TIME

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Destination Mode</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifying Single Operand and</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1.62</td>
<td>1.74</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1.77</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>2.90</td>
<td>3.15</td>
</tr>
<tr>
<td>Modifying Double Operand (Except MOV, SWAB, ROR, ROL, ASR, ASL)</td>
<td>4</td>
<td>2</td>
<td>1.77</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td>3.00</td>
<td>3.25</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3</td>
<td>3.10</td>
<td>3.35</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>4.29</td>
<td>4.66</td>
</tr>
</tbody>
</table>

| MOV | 0 | 0 | 0.00 | 0.00 |
| | 1 | 1 | 0.93 | 0.93 |
| | 2 | 1 | 0.93 | 0.93 |
| | 3 | 2 | 2.17 | 2.29 |
| | 4 | 1 | 1.13 | 1.13 |
| | 5 | 2 | 2.22 | 2.34 |
| | 6 | 2 | 2.37 | 2.49 |
| | 7 | 3 | 3.50 | 3.75 |

| MTPS | 0 | 0 | 0.00 | 0.00 |
| | 1 | 1 | 0.95 | 0.95 |
| | 2 | 1 | 1.13 | 1.26 |
| | 3 | 2 | 2.26 | 2.51 |
| | 4 | 1 | 1.13 | 1.26 |
| | 5 | 2 | 2.26 | 2.51 |
| | 6 | 2 | 2.44 | 2.69 |
| | 7 | 3 | 3.57 | 4.20 |
### Table 2-8 (Cont)
Instruction Set Timing

<table>
<thead>
<tr>
<th>Destination Mode</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.64</td>
<td>0.64</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.64</td>
<td>0.64</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.95</td>
<td>2.08</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1.95</td>
<td>2.08</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2.13</td>
<td>2.26</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3.26</td>
<td>3.51</td>
</tr>
</tbody>
</table>

**III. EXECUTE, FETCH TIME**

#### DOUBLE OPERAND

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD, SUB, CMP, BIT, BIC, BIS, XOR MOV</td>
<td>1</td>
<td>2.03</td>
<td>2.16</td>
</tr>
</tbody>
</table>

#### SINGLE OPERAND

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR, COM, INC, DEC, ADC, SBC, TST</td>
<td>1</td>
<td>1.83</td>
<td>1.96</td>
</tr>
<tr>
<td>SWAB, NEG</td>
<td>1</td>
<td>2.03</td>
<td>2.16</td>
</tr>
<tr>
<td>ROR, ROL, ASR, ASL</td>
<td>1</td>
<td>2.18</td>
<td>2.31</td>
</tr>
<tr>
<td>MTPS</td>
<td>2</td>
<td>2.99</td>
<td>3.12</td>
</tr>
<tr>
<td>MFPS</td>
<td>2</td>
<td>1.99</td>
<td>2.12</td>
</tr>
</tbody>
</table>

**EIS INSTRUCTIONS (use with DST times)**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUL</td>
<td>1</td>
<td><strong>8.82</strong></td>
<td><strong>8.95</strong></td>
</tr>
<tr>
<td>DIV (overflow)</td>
<td>1</td>
<td>2.78</td>
<td>2.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.48</td>
<td>12.61</td>
</tr>
<tr>
<td>ASH</td>
<td>1</td>
<td><strong>4.18</strong></td>
<td><strong>4.31</strong></td>
</tr>
<tr>
<td>ASHC</td>
<td>1</td>
<td><strong>4.18</strong></td>
<td><strong>4.31</strong></td>
</tr>
</tbody>
</table>

**MEMORY MANAGEMENT INSTRUCTIONS**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFPI (D)</td>
<td>2</td>
<td>3.07</td>
<td>3.14</td>
</tr>
<tr>
<td>MTPI (D)</td>
<td>2</td>
<td>3.37</td>
<td>3.34</td>
</tr>
</tbody>
</table>

* Add 200ns for each bit transition in serial data from LSB to MSB
** Add 200ns per shift
Table 2-8 (Cont)
Instruction Set Timing

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Destination Mode</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAB, ROR, ROL, ASR, ASL</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1.42</td>
<td>1.54</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1.57</td>
<td>1.69</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>2.70</td>
<td>2.95</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>1.62</td>
<td>1.74</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td>2.80</td>
<td>3.05</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3</td>
<td>2.90</td>
<td>3.15</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>4.09</td>
<td>4.46</td>
</tr>
<tr>
<td>Non-Modifying</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1.13</td>
<td>1.26</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1.28</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2.42</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>1.33</td>
<td>1.46</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>2.52</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>2.62</td>
<td>2.87</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>3.80</td>
<td>4.18</td>
</tr>
<tr>
<td>Single Operand and Double Operand</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0.98</td>
<td>1.24</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1.32</td>
<td>1.44</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2.20</td>
<td>2.45</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>1.18</td>
<td>1.44</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>2.20</td>
<td>2.45</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>2.40</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>3.59</td>
<td>3.96</td>
</tr>
</tbody>
</table>

**BRANCH INSTRUCTIONS**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR, BNE, BEQ, (Branch)</td>
<td>1</td>
<td>2.18</td>
<td>2.31</td>
</tr>
<tr>
<td>BPL, BMI, BVC, BVS, BCC, BCS, BGE, BLT, BGT, BLE, BHI, BLOS, BHIS, BLO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(No Branch)</td>
<td>1</td>
<td>1.63</td>
<td>1.76</td>
</tr>
<tr>
<td>SOB (Branch)</td>
<td>1</td>
<td>2.38</td>
<td>2.51</td>
</tr>
<tr>
<td>(No Branch)</td>
<td>1</td>
<td>1.98</td>
<td>2.11</td>
</tr>
</tbody>
</table>
### Table 2-8 (Cont)
**Instruction Set Timing**

#### JUMP INSTRUCTIONS

<table>
<thead>
<tr>
<th>Destination Mode</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.83</td>
<td>1.96</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.18</td>
<td>2.31</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3.12</td>
<td>3.37</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2.03</td>
<td>2.16</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3.07</td>
<td>3.32</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3.07</td>
<td>3.32</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4.25</td>
<td>4.78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Cycles</th>
<th>Core</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>2</td>
<td>3.32</td>
<td>3.57</td>
</tr>
<tr>
<td>MARK</td>
<td>2</td>
<td>4.27</td>
<td>4.52</td>
</tr>
<tr>
<td>RTI, RTT</td>
<td>3</td>
<td>4.60</td>
<td>4.98</td>
</tr>
<tr>
<td>Set or Clear C,V,N,Z</td>
<td>1</td>
<td>2.03</td>
<td>2.16</td>
</tr>
<tr>
<td>HALT</td>
<td>1</td>
<td>1.68</td>
<td>1.81</td>
</tr>
<tr>
<td>WAIT</td>
<td>1</td>
<td>1.68</td>
<td>1.81</td>
</tr>
<tr>
<td>RESET</td>
<td>1</td>
<td>100 msec</td>
<td>100 msec</td>
</tr>
<tr>
<td>IOT, EMT, TRAP, BPT</td>
<td>5</td>
<td>7.32</td>
<td>7.7</td>
</tr>
</tbody>
</table>

### LATENCY

Interrupts (BR requests) are acknowledged at the end of the current instruction. For a typical instruction, with an instruction execution time of 4 μsec, the average time to request acknowledgement would be 2 μsec.

Interrupt service time, which is the time from BR acknowledgement to the first subroutine instruction, is 7.32 μsec, max. for core, and 7.7 μsec for MOS.

NPR (DMA) latency, which is the time from request to bus mastership for the first NPR device, is 2.5 μsec, max.
2.7 EXTENDED INSTRUCTION SET

The Extended Instruction Set (EIS) provides the user with the capability of extended manipulation of fixed point numbers. Use of the EIS instructions does not degrade processor timing or affect NPR latency.

The EIS instructions are:

- Multiply (MUL)
- Divide (DIV)
- Arithmetic Shift (ASH)
- Arithmetic Shift Combined (ASHC)

An example of the operation of each instruction follows.

2.7.1 Multiply Instruction

**MUL 070RSS**

**Example:** 16-bit product (R is odd)

```
000241, CLC ; Clear carry condition code
012701,400, MOV #400,R1
070127,10, MUL #10,R1
1034xx, BCS ERROR ; Carry will be set if product is less than 2(15) or greater than or equal to 2(15)

Before
(R1)=000400
```

```
After
(R1)=004000
```

2.7.2 Divide Instruction

**DIV 071RSS**

**Example:**

```
005000, CLR R0
012701,20001, MOV #20001,R1
071027,2, DIV #2,R0
```

Before

After
ASH 072RSS

Arithmetic Shift

2.7.3 Arithmetic Shift Instruction

ASH 072RSS

Example: ASH R0, R3

Before

\[(R0) = 010000\]
\[(R1) = 000001\]

After

\[(R0) = 000003\]
\[(R1) = 001234\]

2.7.4 Arithmetic Shift Combined Instruction

ASHC = 073RSS

Example: Similar to the example for the ASH instruction except that two registers are used.

2.8 PROGRAMMING THE DL11-W SERIAL LINE INTERFACE

The DL11-W is a serial-line interface and a real-time clock. It translates parallel information to serial information (required by a communication device) and translates serial information to parallel information (required by the processor).

In the following description, "transmitter" refers to the registers.
and bits associated with accepting a parallel character from the Unibus for transmission to a communication device via the DL11-W. "Receiver" refers to the registers and bits associated with accepting serial information which is converted to a parallel character and sent to the Unibus.

2.8.1 Device Registers (See Figure 2-3)

The DL11-W contains five Programmable device registers -- the following paragraphs describe these registers. Unused bits are always read as zeros and loading on unused bit has no effect.
Figure 2-3 DL11-M Device Registers
8.1.1 Receiver Status Register (RCSR—Figure 2-4)

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
| RCVR ACT |  |  | RCVR DONE | RCVR INT ENB |  |  | RDR ENB |
```

Receiver Active (Read)
Receiver Done (Read)
Receiver Interrupt Enable (Read/Write)
Reader Enable (Write)

**Figure 2-4 RECEIVER STATUS REGISTER BIT FORMAT**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning and Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Unused.</td>
</tr>
<tr>
<td>7</td>
<td>RCVR DONE == Read Only. Set when an entire character has been received and is ready for transfer to the UNIBUS. Cleared by setting RDR ENB, addressing (READ or WRITE) RBUF or INIT. Starts an interrupt sequence when RECEIVER INTERRUPT ENABLE (bit 6) is also set.</td>
</tr>
<tr>
<td>6</td>
<td>RECEIVER INTERRUPT ENABLE == Read/Write. Cleared by INIT. Starts an interrupt sequence when receiver DONE is set.</td>
</tr>
<tr>
<td>5</td>
<td>Unused.</td>
</tr>
<tr>
<td>0</td>
<td>READER ENABLE == Write Only. Cleared by INIT or at middle of a START bit. Advances paper tape reader of ASR teletypes. Clears RCVR DONE. 20 mA current loop circuit output associated with this bit.</td>
</tr>
</tbody>
</table>

2.8.1.2 Receiver Data Buffer (RBUF — Figure 2-5)
Figure 2-5 Receiver Data Buffer Bit Format

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning and Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td><strong>ERROR</strong> (READ) Logical &quot;OR&quot; of OR, FER, and PER. Cleared by removing the error conditions. <strong>ERROR</strong> is not tied to the interrupt logic, but <strong>RCVR DONE</strong> is.</td>
</tr>
<tr>
<td>14</td>
<td><strong>OVERRUN</strong> (READ) Set if previously received character is not read (<strong>RCVR DONE</strong> not reset) before the present character is read.</td>
</tr>
<tr>
<td>13</td>
<td><strong>FRAMING ERROR</strong> (READ) Set if the character read has no valid stop bit. Also used to detect break.</td>
</tr>
<tr>
<td>12</td>
<td><strong>RECEIVE PARITY ERROR</strong> (READ) Set if received parity does not agree with the expected parity. Always = 0 if no parity is selected.</td>
</tr>
</tbody>
</table>

**NOTE:** Error conditions remain present until the next character is received, at which time, the error bits are updated. **INIT** does not necessarily clear the error bits. Error bits may be disabled via a switch.

| 11, 10, 9, 8 | Unused. |
| 7 = 0        | **RECEIVED DATA BITS** (READ) These bits contain the character just read. If less than 8 bits are selected, the buffer will be right justified into the least significant bits with the higher
unused bit or bits, reading as 0's. Not cleared by INIT.

281.3 Transmitter Status Register (XCSR = Figure 2-6)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XMT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RDY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MAINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BREAK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2-6 Transmitter Status Register Bit Format

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning and Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unused.</td>
</tr>
<tr>
<td>7</td>
<td>TRANSMITTER READY -- Read Only. Set by INIT. Cleared when XBUF is loaded; set when XBUF can accept another character. When set it will start an interrupt sequence if XMIT INT ENH is also set.</td>
</tr>
<tr>
<td>6</td>
<td>Unused.</td>
</tr>
<tr>
<td>6</td>
<td>TRANSMITTER INTERRUPTENABLE -- Read/Write. Cleared by INIT. When set it will start an interrupt sequence if XMIT READY is also set.</td>
</tr>
<tr>
<td>5, 4, 3</td>
<td>Unused.</td>
</tr>
<tr>
<td>2</td>
<td>MAINTENANCE -- Read/Write. Cleared by INIT. When set it disables the serial line input to the RECEIVER and sends the serial output of the TRANSMITTER into the serial input of the RECEIVER. Forces receiver to run at transmitter speed.</td>
</tr>
<tr>
<td>1</td>
<td>Unused.</td>
</tr>
<tr>
<td>0</td>
<td>BREAK -- Read/Write. Cleared by INIT. When set, it transmits a continuous space. May be disabled via a switch.</td>
</tr>
</tbody>
</table>
2.8.1.4 Transmitter Data Buffer (XBUF - Figure 2-7).

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
<td>TDB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07</td>
<td>06</td>
<td>05</td>
<td>04</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transmitted Data Buffer Bits (Write)

Figure 2-7 Transmitter Data Buffer Bit Format

Bit

15 - 8

Unused.

7 - 0

TRANSMITTED DATA BUFFER = Write Only. If less than 8 bits are selected then the character must be right justified into the least significant bits.

2.8.1.5 Clock Status Register (LKS - Figure 2-8)

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MONITOR</td>
<td>INT</td>
<td>ENG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LINE CLOCK MONITOR (READ/CLEAR)
LINE CLOCK INTERRUPT ENABLE (READ/WRITE)

Figure 2-8 Clock Status Register Bit Format

Bit

15 - 8

Unused.
LINE CLOCK MONITOR = Read/Clear. Set only by the line frequency clock signal and cleared only by the program. Set by INIT.

LINE CLOCK INTERRUPT ENABLE = Read/Write. Cleared by INIT. when set, starts an interrupt sequence if Line Clock monitor is also set.

Unused.

NOTE: Line Clock circuit must be disabled via a switch when serial line portion is used as other than console interface (Address 777560).

Priorities are hardwired and are not selectable.

Floating vectors for serial line interface portion are switch selectable.

2.8.2 Interrupts

The DL11-W has three channels of interrupts: one for the receiver section (vector = 0X0), one for the transmitter section (vector = 0X4) and one for the clock section (vector = 100). These circuits operate independently.

2.8.3 Address and Vector Assignments

The DL11-W follows the same address and vector assignments as the KL11, DL11-A, B, C, D which are:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>VECTOR</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE CLOCK</td>
<td>777546</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>777560</td>
<td></td>
</tr>
<tr>
<td>CONSOLE</td>
<td>777652</td>
<td>60/64</td>
</tr>
<tr>
<td></td>
<td>777564</td>
<td></td>
</tr>
<tr>
<td></td>
<td>777566</td>
<td></td>
</tr>
</tbody>
</table>
2.8.4 Timing Considerations

2.8.4.1 Receiver — The RCVR DONE flag sets when the UART has assembled a full character, which occurs at the middle of the first stop bit.

NOTE

The UART (Universal Asynchronous Receiver/Transmitter) is an asynchronous subsystem. The transmitter accepts parallel characters and converts them to a serial asynchronous output. The receiver accepts asynchronous serial characters and converts them to a parallel output.

Since the UART is double buffered, data remains valid until the next character is received and assembled. This allows one full character time for servicing the RCVR DONE flag.

2.8.4.2 Transmitter — The UART’s transmitter section is also double buffered. After initialization, the XMIT RDY flag is set. When the buffer is loaded with the first character, the flag clears but sets
again within a fraction of a bit time. A second character can then be loaded, clearing the flag again. The flag then remains clear for nearly a full character time.

2,8,4,3 Break Generation -- Setting the break bit causes the transmission of a continuous space. Since the XMIT RDY flag continues to function as normal, the duration of break can be timed by the "pseudo-transmission" a number of characters. However, since the transmitter is double buffered, a null character (all zeros) should precede transmission of break to insure the previous character clears the line. Likewise, the last "pseudo-transmitted" character under break should be null.

2.9 CONSTRUCTING A SOFTWARE SWITCH REGISTER

The primary reason for creating a software switch register is to simplify the modification of an existing program which references a hardware switch register. To an executing program, a hardware switch register is essentially a sixteen bit word which may be set externally by an operator. A software switch register is a word in memory which assigned the function of a switch register. A means must be provided for the operator to set the value of the software switch register. In some cases, the means is provided by the M9301 console functions; Load Address, Examine, Deposit, and Start. These cases are those in which the switch register may be set after the program has been loaded but before it has been started and those which are self initializing and may be restarted. The M9301 is not suited for those cases in which the switch register must be set dynamically without aborting the execution of the program. For these cases, most solutions will probably require an interrupt routine driven by the console terminal.

One approach to solving the dynamic switch register problem is to program the keyboard interrupt routine to recognize a typed command to set a new value for the switch register. The value would be entered following the command. The disadvantage to this technique is that several keystrokes are required to set a new value for the switch register when in some cases only a single bit needs to be changed. An alternate approach is to assign a character to clear the switch register, another to set all bits of the register, and sixteen others which would each be assigned to a specific bit of the register which would be complemented when the character was struck. The advantage of this approach is that allows a single bit to be changed without changing the others.
CHAPTER 3 HARDWARE GUIDE

3.1 PDP-11/34 SYSTEM DESCRIPTION
3.1.1 OEM Machine Components
3.1.2 End User Machine Components
3.1.3 Processor Mounting Boxes

3.2 OPERATORS CONSOLE
3.2.1 Information to Computer
3.2.2 Information to Operator

3.3 M9301-YA AND YB TERMINATOR/ROM
3.3.1 M9301-YA
3.3.2 M9301-YB

3.4 PDP-11/34 PROCESSOR
3.4.1 Processor Functions
3.4.2 Instruction Set
3.4.3 Extended Instruction Set
3.4.4 Memory Management
3.5.1.1 Volatility
3.5.2 Core Memory
3.5.2.1 Volatility
3.5.3 Memory Parity
3.5.4 Memory Organization
3.5.4.1 Addressing
3.5.4.2 Addressing Capability

3.6 MEMORY PARITY
3.6.1 Odd Parity
3.6.2 Detecting Odd Parity

3.7 DL11W SERIAL LINE INTERFACE/REAL TIME CLOCK
3.7.1 Physical Description
3.7.1.1 Switch Packs
3.7.2 Serial Line Interface
3.7.3 Real Time Clock

3.8 POWER SUPPLIES
3.8.1 H777 Power Supply
3.8.1.1 Power Supply Conversion
3.8.1.2 Auxiliary Power Supply Signals
3.8.1.3 Power Control
3.8.2 H765 Power Supply
3.8.2.1 Power Supply Conversion
3.8.2.2 Auxiliary Power Supply Signals

3.9 UNIBUS
3.9.1 Non-Processor Communication
3.9.2 Handshaking Sequence
3.9.3 Asynchronous Operation
3.9.4 Non-Processor Request (NPR)
3.9.5 Standard or Modified Unibus

3.10 TERMINATORS
3.10.1 Beginning of Bus Terminator
3.10.2 End of Bus Terminator

3.11 DD11-P BACKPLANE
3.11.1 Physical Description
3.11.2 Unibus Connections
3.11.3 Processor Connections
3.11.4 Small Peripheral Controller Connections
The PDP-11/34 Computer System consists of a set of modular building blocks combined to suit both original equipment manufacturer (OEM) and end user (see Figure 3-1) needs. The PDP-11/34 can interface with the PDP-11 family of options.

### 3.1.1 OEM Machine Components

The basic components of this PDP-11/34 System are:

- KD11-E Central Processor
- KY11-LA Operators Console
- M9301-YA Terminator/ROM
- Unibus
- M9302 Unibus Terminator
- Core or MOS Memory

Devices such as the DL11-W Serial Line Interface, KY11-LB Programmers Console, and the M7850 Parity Controller are optional.

### 3.1.2 End User Machine Components

The basic components of this PDP-11/34 System are:

- KD11-E Central Processor
- KY11-LA Operators Console
- M9301-YB Terminator/Boot/Diagnostic
- Unibus
- M9302 Unibus Terminator
- DL11-W Serial Line Interface
- M7850 Parity Controller
- Core or MOS Memory

Note that the DL11-W Serial Line Interface and M7850 Parity Controller are standard on the end user system.
The KY11-LB programmers Console is available as an option.

3.1.3 Processor Mounting Boxes

Two processor mounting boxes are available for the PDP-11/34 or PDP-11/39 Systems - one box is the compact RA11-L measuring 5-1/4 inches high; the second is the larger RA11-K measuring 10-1/2 inches high. The RA11-L contains one nine-slot backplane (MD11-P). Details of the mounting boxes are included in the configuration section.
Figure 1: System Block Diagram

Standard for the DEMM Processor

*Options for the DEMM processor

Central Processor

Memory

Options

M9302 UHBISS Terminator

Operator's Console

M9301 Term/Boot/Dial

DL16 Serial Line Interface

M7450 Parity Controller

Output Device
(TTY, DECSWITCH, CRT, LINE PRINTER)

Power Supply

→ TO ALL MODULES

Ball L 5 1/4" Box

Ball K 10 1/2" Box
3.2 OPERATORS CONSOLE

The operators console (front panel of the computer) is a communication link between the operator and the computer (see Figure 3-2).

3.2.1 Information to Computer

The user can provide the system with control information via the following switches:

- **OFF/ON/STBY rotary switch** - The three settings allow the user to respectively:
  1. Remove power
  2. Remain in standby to power refresh circuits on MOS memory
  3. Apply DC power to logic

- **HALT/CONT Switch** - This switch allows the user to manually halt the processor or to resume operation from the halt state without re-initializing the processor.

- **BOOT/INIT Switch** - This switch allows manual entry of the boot sequence or initialization of the processor.

3.2.2 Information to Operator

The three indicator lamps on the front panel provide a continuous display of machine status. The three indicators are:

- **BATT** Indicates the condition of the battery by its flash rate.
- **DC ON** Indicates power has been applied to the logic.
- **RUN** Indicates processor is halted when this light is extinguished.

A detailed description of the operators console is provided in the Systems Maintenance Guide.
Figure 3-2 Operators Console Front Panel
3.3 M9301-YA AND -YB TERMINATOR/ROM

The M9301 Terminator/ROM is a double-height module which terminates the modified Unibus and which provides 512 words of read-only memory. The read-only memory contains (1) the software to generate the START, LOAD, EXAMINE, and DEPOSIT functions from the terminal keyboard, (2) bootstrap routines to cause data transfer from peripheral devices to memory, and (3) diagnostics for verifying proper machine operation.

3.3.1 Original Equipment Manufacturer Version

The M9301 comes in an original equipment manufacturer (OEM) version designated M9301-YA. In this version, a bootstrap program can be executed automatically without using the console. The OEM version, however, does not support Massbus mass storage devices. The M9301-YA supports the bootstrap programs of the following devices:

- RK11 Disk Cartridge
- RP11 Disk Pack (RP02/RP03)
- TC11 DECTape
- TM11 800 BPI Magtape
- RX11 Diskette
- DL11 Teletype (ASR 33)
- PC11 Paper Tape
  Console Emulator

3.3.2 End User Version

The end user version of the M9301 (designated M9301-YB) must execute all bootstrap programs via the operators console. In addition, the M9301-YB supports the following Massbus devices: RS04, RP04, disks and the TU16 Magtape Unit. The M9301-YB supports the bootstrap programs of the following devices:

- RK11 Disk Cartridge
- RP11 Disk Pack (RP02/RP03)
- TC11 DECTape
- TM11 800 BPI Magtape
- TA11 Magnetic Cassette
- RX11 Diskette
- DL11 Teletype (ASR 33)
- PC11 Paper Tape
  RJS03/RJS04 Fixed Head Disk
  RJP04 Moving Head Disk
  TJU16 Magnetic Tape
  Console Emulator

3.4 PDP-11/34 PROCESSORS
The PDP-11/34 Processing System contains a KD11E Central Processor, which consist of two high-density, hex-height modules. The modules are designated M7265 and M7266.

3.4.1 Processor Functions

The KD11E connects to the system via the Unibus, controls the time allocation of the Unibus for peripherals, and performs arithmetic operations, logic operations, and instruction decoding. It contains multiple, high-speed, general purpose registers which can be used as accumulators, address pointers, index registers and other specialized functions. The processor can perform data transfers directly between I/O devices and memory without disturbing the processor registers; does both single and double-operand addressing and handles both 16-bit word and 8-bit byte data.

3.4.2 Instruction Set

The KD11-E executes the PDP-11/35 instruction set.

3.4.3 Extended Instruction Set

The Extended Instruction Set (EIS) is a standard feature which provides the capability of performing hardware fixed-point arithmetic and allows direct implementation of multiply, divide, and multiple shifting. This feature can process double-precision 32-bit words.

3.4.4 Memory Management

The KD11-E contains memory management logic which provides memory extension, relocation and protection. This enables the user to:

1. Extend memory space from 28K to 124K words.
3. Provide effective protection of memory segments in multi-user environments.

3.4.4.1 Modes of Operation - The memory management modes of operation are kernel and user. When the machine is in kernel mode, a program has complete control of the machine. When in User mode, the processor cannot execute certain instructions and can be denied direct access to the peripherals in the system. This hardware feature can be used to
provide complete executive protection in a multi-programming environment. A software operating system can insure that no user (operating in User Mode) can gain access to non-user program areas (such as the operating system monitor).

3.4.4.2 Processor Status Word - Bits 12 through 15 of the processor status word are used with memory management and contain the operating mode of the CPU. Mode information includes the present mode, either kernel or user (bits 15,14) and the mode the machine was in prior to the last interrupt or trap (bits 13,12).

3.5 MEMORY

The PDP-11/34 Processing system is designed to operate with core memory and MOS (metal oxide semiconductor) memory.

3.5.1 MOS Memory

MOS memory is available in 16K increments. Each 16K increment is contained in a single hex-height board. The MOS module contains a Unibus interface, timing and control, refresh circuitry, and MOS storage array.

3.5.1.1 Volatility - MOS memory is volatile which means that data is lost when power is removed. In order to prevent loss of data during loss of power, an optional battery system is available to ensure the preservation of the data. When the system is operating in this battery back-up mode, MOS power consumption is less than one-half of the normal power requirement, which ensures maximum data retention time for a given battery capacity.

3.5.2 Core Memory

16K of core memory is provided on a double hex-height module. The module consists of a stack board containing the core mat and diode arrays which plug into a control board containing the drive circuitry, timing chain and Unibus interface logic.

3.5.2.1 Volatility - Core memory is not volatile; that is, data is not lost when power is removed from the system.
3.5.3 Memory Parity

Either MOS or core memory is designed to operate as a parity or non-parity system. The parity system provides 18 bits per word and requires one M7850 parity controller per backplane, with a non-parity system, 16 bit words are provided to the Unibus.

3.5.4 Memory Organization

The memory is organized in 16-bit words consisting of two 8-bit bytes (high byte and low byte). The memory contains 16,384 words (32,768 bytes); therefore, 32,768 locations are assigned. The address locations are specified as six-digit octal numbers. The 32,768 locations for the 16K memory are designated 000000 through 077776.

3.5.4.1 Addressing - Each byte is addressable and has its own unique address; low bytes are even-numbered and high bytes are odd-numbered. Words are addressed at even locations only with both low and high bytes automatically included. Consecutive words are therefore found in even numbered addresses.

3.5.4.2 Addressing Capability - The PDP-11 performs data transfers using the address and data lines of the Unibus. Unibus addresses are 18 bits (A17-A0) providing the capability of addressing 262,144 (256K) bytes or 13,1072 (128K) words. The basic processor provides 16 bits of address which gives a maximum addressable range of 32K to the memory management unit (standard in the PDP-11/34 Processor). With memory management disabled, if bits A(15:13) are all 1's, bits A(17:16) are forced to 1's which relocates the last 8K bytes (4K words) to the highest 4K words of address space (124-128K). These top 4096 words are reserved for peripherals and register addresses and the user therefore has a maximum of 32K-4K or 28K (28,672) words of memory to program. With the memory management unit enabled, addresses may be relocated anywhere in the 128K address space. The high 4K of address space is still present for peripherals and registers; therefore the user may utilize a maximum of 124K of memory.

3.6 MEMORY PARITY (M7850)

The memory parity option consist of a double-height module designated the M7850 Parity Controller. The board also contains a Control and Status register which logs the higher order address bits of a memory location containing a parity error.
3.6.1 Odd Parity

The memory parity option contains two parity bits - one per each 8-bit byte. If the number of 1's in a byte is even, the parity bit for that byte is asserted, resulting in odd (correct) parity. If the number of 1's in a byte is odd, the parity bit for that byte is unasserted (0), also resulting in odd parity.

3.6.2 Detecting Odd Parity

If the Error Enable bit (bit 0 in Control and Status register) is set, parity is checked and if a parity error occurs, the LED will be lit and a signal sent back to the master device informing the device about the error. *(Unibus signal BUS PBL)*

3.7 DL11-W SERIAL LINE INTERFACE

The DL11-W Serial Line Interface is a serial line interface and a line frequency clock. All software control of this device is accomplished via five device registers on the module itself. These registers are assigned Unibus addresses and can be loaded and/or read via PDP-11 instructions. The Programmers Guide provides a detailed description of these registers.

3.7.1 Physical Description

The DL11-W is a quad-height board and can be plugged into one of seven small peripheral controller slots in the backplane.

3.7.1.1 Switch Packs - The DL11-W module contains five switch packs which allow the user to change:

1. Baud rate
2. Address assignment of internal registers
3. Vector address assignments
4. UART (Universal Asynchronous Receiver/Transmitter) Options
   a. Number of data bits in the Serial Transmission
   b. Enable or disable parity in the Serial Transmission
   c. Selection of odd or even parity in the Serial Transmission
5. Active or passive transmitter mode
6. Active or passive receiver mode
7. TTY break enable
8. Error bits in receiver buffer
9. Enable of line clock circuitry

The Configuration Guide (Section 5) provides a detailed explanation of the switch pack settings.

3.7.2 Serial Line Interface

The serial line interface translates parallel data to serial data required by a communications device, such as a Teletype or display terminal. It also translates serial data to parallel data required by the PDP-11/34 Processor. This two-way communication is provided by an LSI (large-scale integration) integrated circuit known as a UART (Universal Asynchronous Receiver Transmitter).

The DL11-W may be used with all DEC terminals and teletypes and other switch-selectable baud rates of 110, 150, 300, 600, 1200, 2400, 4800, and 9600.

3.7.3 Real Time Clock

The real time clock portion of the DL11-W provides the user with 50 Hz or 60 Hz line frequency intervals and is directly compatible with programs that utilized the KW-11L Line Time Clock. The KW11-L register address is 777546, vector address is 100 (ctcal), and priority level is BR6.

3.8 POWER SUPPLIES

The PDP-11/34 can utilize the compact BA11-L mounting box (5-1/4-inch chassis), the BA11-K mounting box (10-1/2-inch chassis), or the BA11-F mounting box (21-inch chassis). The BA11-L contains the H777 Power Supply, the BA11-K contains the H765 Power Supply, and the BA11-F contains the H7420 Power Supply.

3.8.1 H777 power Supply

The H777 Power Supply consists of two d-c regulator modules, dc distribution board, ac input section, transformer, and fan. A third dc regulator is included in processors which have core memory.
3.8.1.1 Power Supply Conversion - The H777 Power Supply converts 115 VAC or 230 VAC, 47-63 Hz, line voltage to six regulated d-c voltages that are used by all system elements. The dc voltages are:

<table>
<thead>
<tr>
<th>REGULATOR (MAX)</th>
<th>VOLTAGE</th>
<th>OUTPUT CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5411597</td>
<td>+5 V</td>
<td>25 A</td>
</tr>
<tr>
<td>5411601</td>
<td>+15 V</td>
<td>1 A</td>
</tr>
<tr>
<td></td>
<td>-15 V</td>
<td>1 A</td>
</tr>
<tr>
<td></td>
<td>+5 V</td>
<td>4 A</td>
</tr>
<tr>
<td>Core Option</td>
<td>+20 V</td>
<td>6 A</td>
</tr>
<tr>
<td></td>
<td>-5 V</td>
<td>4 A</td>
</tr>
</tbody>
</table>

3.8.1.2 Auxiliary Power Supply Signals - In addition to the dc voltages defined above, LTC, LBUS AC LO L, and BUS DC LO L signals are available. LTC L is the Line Time Clock signal provided to drive the Line Time Clock. BUS AC LO L and BUS DC LO L activate the processor power-fail and auto-restart circuitry.

3.8.1.3 Power Control - The power control provides the ac power to the power supply and cooling fans. The power control is installed in the front of the HA11-L mounting box, and consists of a line cord, circuit breaker, delay and transformer. The power control is configured to run on 115 Vac or 230 Vac by jumper straps on a readily accessible terminal strip.

3.8.2 H765 Power Supply

The H765 Power Supply, contained in the HA11-K mounting box, consists of five regulators, two fans, ac input box, transformer assembly, and power distribution board.

3.8.2.1 Power Supply Conversion - The H765 Power Supply converts 115 Vac or 230 Vac, 47-63 Hz line voltage to six regulated dc voltages. The dc voltages are:

<table>
<thead>
<tr>
<th>REGULATOR (MAX)</th>
<th>VOLTAGE</th>
<th>OUTPUT CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>H745</td>
<td>-15 Vdc</td>
<td>10 A</td>
</tr>
<tr>
<td>H744</td>
<td>+5 Vdc</td>
<td>25 A</td>
</tr>
<tr>
<td>H744</td>
<td>+5 Vdc</td>
<td>25 A</td>
</tr>
<tr>
<td>H754</td>
<td>+20 Vdc</td>
<td>8 A</td>
</tr>
<tr>
<td></td>
<td>-5 Vdc</td>
<td>1 to 8 A</td>
</tr>
</tbody>
</table>

Only available on core memory machines.
Maximum +5 V current is dependent on +20 V current. It is equal to 1A plus the current of the +20 V Supply, up to a total of 8A.

3.8.2.2 Auxiliary Power Supply Signals - In addition to the dc voltages defined above, LTC L, BUS AC LO L, and BUS DC LO L signals are available. LTC L is the line time clock signal provided to drive the line Time Clock. BUS AC LO L and BUS DC LO L activate the processor power-fail and auto-restart circuitry.

3.9 UNIBUS

All computer system devices and peripherals connect to and communicate with each other on a single high-speed bus known as the Unibus (see Figure 3-1). Since all devices (including the central processor) communicate with each other in the same manner, the processor has the same easy access to peripherals as it has to memory.

3.9.1 Non-Processor Communication

Both bidirectional and asynchronous communication occur on the Unibus. Devices can send and receive data independently without processor intervention. For example, a cathode ray tube display can refresh itself from memory while the central processor attends to other tasks.

3.9.2 Handshaking Sequence

Device communications on the Unibus are interlocked. For each command issued by a master device, a response signal is received from a ‘slave’ completing the data transfer. Device-to-device communication is completely independent of physical bus length and the response times of master and slave devices within Unibus specifications.

3.9.3 Asynchronous Operation

Because the Unibus is asynchronous, it is compatible with devices operating over a wide range of speeds. The Maximum Transfer rate on the Unibus is one 16-bit word every 400 nanoseconds or 2,500,000 words per second.
3.9.4 Non-Processor Request (NPR)

Input/output devices transferring directly to or from memory are given highest priority and may request bus mastership to steal bus cycles during instruction operations. Multiple devices can operate simultaneously at maximum direct memory access (DMA) rates by "stealing" bus cycles.

3.9.5 Standard or Modified Unibus

The PDP-11/34 processing system has both standard and modified Unibus connections. The modified Unibus is similar to the standard Unibus except that extra grounds and all bus grant signals have both been removed and replaced by voltage rails and parity signals to provide the proper signals and voltages for modified Unibus devices (MOS memory, for example).
Figure 3-2 PDP-11 System Simplified Block Diagram
3.10 TERMINATORS

Every PDP-11 is shipped with two Unibus terminators— a beginning-of-bus Terminator and an end-of-bus terminator.

3.10.1 Beginning-of-Bus Terminator

The terminator at the beginning of the Unibus (near the Central Processor) is a M9301 terminator which is compatible with the modified Unibus (DD11-P backplane— slots 3 through 8). The M9301 contains the Unibus termination circuits and logic to implement the ROOT features and 512 words of ROM memory.

3.10.2 End-of-Bus Terminator

The terminator at the end of the Unibus is an M9302 which contains terminating resistors and an auxiliary circuit which generates a BUS SACK signal if a GRANT signal ever reaches the end of the bus. As a result of this circuitry, the SACK timeout feature found on other processors is not required, and thus system latency is reduced. This terminator must be installed at the end of the Unibus in all PDP-11/34 systems.

3.11 DD11-P BACKPLANE

The DD11-PK backplane provides the electrical connections between all modules in the system (see Figure 3-3).

3.11.1 Physical Description

The DD11-PK backplane is an assembly consisting of 54 slots arranged in a 6X9 matrix. The rows of the matrix are designated by numbers and the columns by letters. The modules are inserted along the rows—for example, a hex-height module might be placed in the hex-height slot from A1 to F1 in row 1 and a quad-height module in C6 through F6 as shown in Figure 3-3.

3.11.2 Unibus Connections

A1, B1, A9, and B9 are standard Unibus connector slots. A2, B2 through A8, B8 are modified Unibus slots which are utilized by modified Unibus devices (e.g., MOS memory).
3.11.3 Processor Connections

The two hex-height processor boards are plugged into slots one and two. These slots contain interconnections to allow inter-communication between the two processor boards.

3.11.4 Small Peripheral Controller (SPC) Connections

The remaining connectors on the backplane are pinned to accommodate small peripheral controllers or modified Unibus devices such as MOS memory.
M9301
SOFTWARE
GUIDE
CONTENTS

4.1 M9301 Terminator/ROM
  4.1.1 OEM Version (M9301-YA)
  4.1.2 End-User Version (M9301-YB)
  4.1.3 General
  4.1.4 Bootstraps
  4.1.5 Devices Supported by M9301
  4.1.6 M9301 = Program Memory Map
4.1 M9301 TERMINATOR/ROM

The M9301 Terminator/ROM comes in an OEM (original equipment manufacturer) version and an end-user version. The OEM version is designated M9301-YA and the end-user version is designated M9301-YB. Both versions contain basic CPU and memory GO-NO GO diagnostics.

4.1.1 OEM Version (M9301-YA)

The OEM version can be set up such that a bootstrap can be executed automatically without using the console terminal. However, the OEM version does not support Massbus devices.

4.1.2 End-User Version (M9301-YB)

The end-user version requires bootstraps to be started via the console and does support Massbus devices.

4.1.3 General

The normal setting for both the YA and YB produce the same program flow (see Figure 1-2 Operators Guide). The basic CPU diagnostics, TEST1 through TEST5, are completely executed if no CPU error is detected. The contents of register R0, R4, SP, and PC at the time of power up or BOOT request are displayed on the console terminal. The prompting character string (CR) (LF) ($) indicates that some diagnostics have been run and the processor is operating. The string is displayed on the console terminal. The keyboard dispatch routine is a program in read-only memory which (1) allows LOAD, DEPOSIT, EXAMINE, and START commands to manipulate data, and start a program, or (2) which allows a bootstrap program to be executed. This routine waits for a console function or a bootstrap request to be entered at the terminal. For load, examine, and deposit functions, the function is executed and the keyboard dispatch routine is reentered. For all bootstrap requests, an optional unit number may be typed following the device code. Zero is the assumed unit number if one is not specified. Before the bootstrap is executed, additional memory modifying CPU diagnostics (TEST6 and TEST7) are executed, followed by memory diagnostics. If the memory diagnostic fails, the machine will halt, the data read, the data written, the address of the location at fault, and the error indicator (PC) are printed out when the BOOT switch is depressed.

If all diagnostics are executed successfully, the common bootstrap routine loads the address of the device's main control status register into R1 (R0 will contain the unit number typed or zero if none) and transfer to the specified device bootstrap.
With battery back-up, a good battery, and TPl on the M930l connected, the system does not go into the bootstrap routine on power restart. If there is no battery back-up (i.e. system uses core memory) or if the battery is low and TPl is connected the system will go into the bootstrap routine on a power restart. Therefore, in a system without battery back-up, TPl on the M930l must be disconnected if the user does not want to go into the bootstrap routine on all power restarts.
4.1.4 Bootstraps

Generally bootstrap routines read a secondary bootstrap from the specified device into memory and transfer execution to the secondary bootstrap. Typically the bootstrap routine will read the first physical sector, block, or 512 words, depending on the device, starting at memory location zero, and transfer to location zero. The paper tape bootstraps (TT or PR) are different in that they read a bootstrap formatted paper tape, such as the absolute loader, into the uppermost region of memory. If an error occurs (paper tape bootstraps cannot detect any errors), the bootstrap procedure will be retried up to ten times. If the bootstrap routine is unsuccessful with all attempts the console routine will be entered unless a non-standard switch setting has been set on the M9301 module in which case the programs will halt. The console routine displays R0, R4, SP and R5, and enters the keyboard dispatch routine.

4.1.5 Devices supported by M9301

Table 4-1 shows the devices supported by the M9301-YA and M9301-YB modules, the device code used by the keyboard dispatch routine, the standard Unibus address of the device, and any special notes about the device.

4.1.6 M9301 Program Memory Map

Table 4-2 is a program memory map of the M9301 module and briefly lists the nature of each test in the M9301 ROM.
## Devices Supported by M9301

### Table 4-1

<table>
<thead>
<tr>
<th>DEVICE CODE</th>
<th>UNIBUS ADDRESS</th>
<th>DEVICE &amp; SPECIAL NOTES IF ANY FOR M9301-YA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>777560</td>
<td>TERMINAL PAPER TAPE READER</td>
</tr>
<tr>
<td>DK</td>
<td>777404</td>
<td>RK11 MOVING HEAD DISK CARTRIDGE</td>
</tr>
<tr>
<td>DT</td>
<td>777342</td>
<td>TC11 DECTAPE</td>
</tr>
<tr>
<td>MT</td>
<td>772522</td>
<td>TM11 MAGNETIC TAPE DRIVE. TAPE MUST BE 7 OR 9 TRACK, 800 BPI, ODD PARITY, AND DUMP MODE</td>
</tr>
<tr>
<td>DP</td>
<td>776714</td>
<td>RP11 MOVING HEAD DISK PACK FOR RP02/03</td>
</tr>
<tr>
<td>CT</td>
<td>777500</td>
<td>TA11 CASSETTE</td>
</tr>
<tr>
<td>PR</td>
<td>777550</td>
<td>PC11 HIGH SPEED PAPER TAPE READER, TAPE MUST BE IN A SPECIAL BOOTSTRAP FORMAT (SUCH AS ABSLDR)</td>
</tr>
<tr>
<td>DX</td>
<td>777170</td>
<td>RX11 DISKETTE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEVICE CODE</th>
<th>UNIBUS ADDRESS</th>
<th>DEVICE &amp; SPECIAL NOTES IF ANY FOR M9301-YB</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>777560</td>
<td>TERMINAL PAPER TAPE READER</td>
</tr>
<tr>
<td>DS</td>
<td>772040</td>
<td>RJS03/04 MASSBUS FIXED HEAD DISK</td>
</tr>
<tr>
<td>MM</td>
<td>772440</td>
<td>TJU16 MASSBUS TAPE DRIVE. TAPE MUST BE 9-TRACK, 800 BPI, AND ODD PARITY.</td>
</tr>
<tr>
<td>MC</td>
<td>776300</td>
<td>MIXED COMBINATION OF MASSBUS DEVICES. THE ACTUAL DEVICE IS DETERMINED BY THE SPECIFIED UNIT NUMBER. THE DEVICE CAN BE A TJU16, RJP04, OR RJS03/04.</td>
</tr>
<tr>
<td>DB</td>
<td>776700</td>
<td>RJP04 MASSBUS DISK PACK. FORMAT 22, ECC INHIBIT</td>
</tr>
<tr>
<td>DK</td>
<td>777404</td>
<td>RK11 MOVING HEAD DISK CARTRIDGE</td>
</tr>
<tr>
<td>DT</td>
<td>777342</td>
<td>TC11 DECTAPE</td>
</tr>
<tr>
<td>MT</td>
<td>772522</td>
<td>TM11 MAGNETIC TAPE DRIVE. TAPE MUST BE 7 OR 9 TRACK, 800 BPI, ODD PARITY, AND DUMP MODE</td>
</tr>
<tr>
<td>DP</td>
<td>776714</td>
<td>RP11 MOVING HEAD DISK PACK FOR RP02/03</td>
</tr>
<tr>
<td>CT</td>
<td>777500</td>
<td>TA11 CASSETTE</td>
</tr>
<tr>
<td>PR</td>
<td>777550</td>
<td>PC11 HIGH SPEED PAPER TAPE READER, TAPE MUST BE IN A SPECIAL BOOTSTRAP FORMAT (SUCH AS THE ABSLDR)</td>
</tr>
</tbody>
</table>
DX 777170 RX11 DISKETTE
TEST1 - SINGLE OPERAND TEST

This test executes all single operand instructions using destination mode 0. The basic objective is to verify that all single operand instructions operate. It also provides a cursory check on the operation of each instruction, while ensuring that the CPU decodes each instruction in the correct manner.

TEST1 brings the test destination register through its three possible states: zero, negative, and positive. Each instruction operates on the register contents in one of four ways:

Data will be changed via a direct operation, i.e., increment, clear, decrement, etc.

Data will be changed via an indirect operation, i.e., arithmetic shifts, add carry, and subtract carry.

Data will be unchanged, but operated upon, via a direct operation, i.e., clear a register already containing zeroes.

Data will be unchanged via a non-modifying instruction (TST).

Note that when operating upon data in an indirect manner, the data is modified by the state of the appropriate condition code. Arithmetic shift will move the "C" bit into or out of the destination. This operation, when performed correctly, implies that the "C" bit was set correctly by the previous instruction.

There are no checks on the data integrity prior to the end of the test. However, a check is made on the result. A correct result implies that all instructions manipulated (or, did not manipulate) the data in the correct way. If the data is incorrect, the program will loop.

TEST2 - DOUBLE OPERAND, ALL SOURCE MODES, DESTINATION MODE 0

This test verifies all double operand general and logical instructions - each in one of the seven modes (excludes mode 0). Thus, two operations are checked; the correct decoding of each double operand instruction, and the correct operation of each addressing mode for the source operand.

Each instruction in the test must operate correctly in order for the next instruction to operate. This inter-dependence is carried through to the last instruction (bit test) where, only through correct execution of all previous instructions, a data field is examined for a specific bit configuration. Thus, each instruction prior to the last, serves to set up the pointer to the test data.

Two checks on instruction operation are made in TEST2. One check, a branch on condition, is made following the compare instruction, while the second is made as the last instruction in the test sequence.
Since the GO-NO GO test resides in a ROM memory, all data manipulation (modification) must be performed in destination mode 0 (register contains data). The data and addressing constants used by TEST2 are contained in a literal pool within the ROM.

It is important to note that two different types of operations must execute correctly in order for this test to operate:

1. Those instructions that participate in computing the final address of the data mask for the final bit test instruction.

2. Those instructions that manipulate the test data within the register to generate the expected bit pattern.

Detection of an error within this test results in a hard loop.

TEST3 - JUMP TEST MODES 1, 2, AND 3

The purpose of this test is to ensure correct operation of the Jump instruction. This test is constructed such that only a Jump to the expected instruction will provide the correct pointer for the next instruction.

There are two possible failure modes that can occur in this test:

1. The Jump addressing circuitry will malfunction causing a transfer of execution to an illogical instruction sequence or non-existent memory.

2. The Jump addressing circuitry will malfunction in such a way as to cause the CPU to loop.

The latter case is a logical error indicator. The former, however, may manifest itself as an after-the-fact error. For example, if the Jump causes control to be given to other routines within the M9381, the inter-dependent instruction sequences would probably cause a failure to eventually occur. In any case, the failing of the Jump instruction will eventually cause an out of sequence or "illogical event" or occur. This is a meaningful indicator of a malfunctioning CPU.

TEST4 - SINGLE OPERAND, NON-MODIFYING, BYTE TEST

This test focuses on the one unique single operand instruction, the TST, TST is a special case in the CPU execution flow since it is a non-modifying operation. TEST4 also tests the byte operation of this instruction. The TSTB instruction will be executed in Mode 1 (register deferred) and Mode 2 (register deferred, auto increment).

The TSTB is programmed to operate on data which has a negative value most significant byte and a zero (not negative) least significant
byte.

In order for this test to operate properly, the TSTB on the LS8 must, first, be able to access the even addressed MSB, then set the proper condition codes. The TSTB is then re-executed with the auto-increment facility. After the auto-increment, the addressing register should be pointing to the MSB of the test data. Another TSTB is executed on what should be the MSB, the "N" bit of the condition codes should be set by this operation.

Correct execution of the last TSTB implies that the auto-increment recognized that a byte operation was requested, thereby only incrementing the addressing by one, rather than two. If the correct condition code was not set by the associated TSTB instruction, the program will loop.

TEST5 - DOUBLE-OPERAND, NON-MODIFYING TEST

There are two non-modifying double-operand instructions - the compare (CMP) and bit test (BIT). These two instructions operate on test data in source modes 1 and 4, and destination modes 2 and 4.

The BIT and CMP instructions will operate on data consisting of all ones (177777). Two separate fields of ones are used in order to utilize the compare instructions, and to provide a field large enough to handle the auto-incrementing of the addressing register. Since the compare instruction is executed on two fields containing the same data, the expected result is a true "Z" bit, indicating equality.

The BIT instruction will use a mask argument of all ones against another field of all ones. The expected result is a non-zero condition (Z).

All failures will result in a one instruction loop.

TEST6 - DOUBLE-OPERAND, MODIFYING, BYTE TEST

The objective of this test is to verify that the double-operand, modifying instructions will operate in the byte mode. TEST5 contains three sub-tests:

1. Test source mode 2, destination mode 1, odd and even bytes.
2. Test source mode 3, destination mode 2.
3. Test source mode 0, destination mode 3, even byte.

The move byte (MOVB), bit clear byte (BICB), and bit set byte (BISB) are used within TEST6 to verify the operation of the modifying, double-operand functions.
Since modifying instructions are under test, memory must be used as a destination for the test data. TEST6 uses location 500 as a destination address. Later, in TEST7 and the Memory Test, location 500 is used as the first available storage for the stack.

Note that, since TEST6 is a byte test, location 500 implies that both 500 and 501 are used for the byte test (even and odd, respectively). Thus, in the word of data at 500, both odd and even bytes are caused to be all zeroes and all ones throughout the test. Each byte is modified independently of the other.

TEST7 = JSR TEST

The JSR is the first test in the GO-NOGO sequence that utilizes the stack. The Jump Subroutine command (JSR) is executed in modes 1 and 6. After the JSR is executed, the subroutine which was given control, will examine the stack to ensure that the correct data was placed in the correct stack location (500). The routine will also ensure that the link back register points to the correct address. Any errors detected in this test will result in a Halt.

TEST8 = MEMORY TEST

Although this test is intended to test both core and MOS memories, the data patterns used are designed to exhibit the most taxing operation for MOS. Before the details of the test are described, it would be appropriate to discuss the assumptions placed upon the failure modes of the MOS technology.

This test is intended to check for two types of problems that may arise in the memory:

1. Solid Element or Sense Amp failures
2. Addressing Malfunctions external to the chip.

The simplest failure to detect is a solid read or write problem. If a cell fails to hold the appropriate data, it is expected that the Memory Test will easily detect this problem. In addition, the program attempts to saturate a chip in such a way as to cause marginal sense amp operation to manifest itself as a loss or pick-up of unexpected data. The 4K X 1 chip used in the memory consists of a 64 X 64 matrix of MOS elements. Each 64 bit section is tied to a common sense amplifier. The objective of the program is to saturate the section with, at first, all zeroes and one "1" bit. This "1" bit is then floated down through the section. At the end, the data is complemented, and the test repeated.

For external addressing failures it is assumed that if two or more locations are selected at the same time, and a write occurs, it is likely that both locations will assume the correct state. Thus, prior to writing any test data, the background data is checked to ensure
that there was no crosstalk between any two locations. All failures will result in a program halt as do failures in tests 6 and 7. After the halt, it is expected that the operator will depress the BOOT switch causing R0 (Expected Data), R4 (Received Data), SP (Failing Address), and PC (PC indicating memory failure) to be displayed.

NOTE

If the expected and received data are the same, it is highly probable that an intermittent failure has been detected (i.e., timing or margin problem). The reason the expected and received data can be identical is that the test program re-reads the failing address after the initial non-compare is detected. Thus, a failure at CPU speed is detected, and indicated by the reading of the failing address on a single reference (not at speed) operation.
<table>
<thead>
<tr>
<th>TEST 1</th>
<th>ALL SINGLE OPERAND INSTRUCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST 2</td>
<td>ALL DOUBLE OPERAND INSTRUCTIONS</td>
</tr>
<tr>
<td>TEST 3</td>
<td>JUMP TEST MODES 1, 2, 3</td>
</tr>
<tr>
<td>TEST 4</td>
<td>SINGLE OPERAND, NON-MOD, BYTE TEST</td>
</tr>
<tr>
<td>TEST 5</td>
<td>DOUBLE OPERAND, NON-MOD S1, S4, D2, D4</td>
</tr>
<tr>
<td></td>
<td>REGISTER DISPLAY ROUTINE</td>
</tr>
<tr>
<td>TEST 6</td>
<td>DOUBLE OPERAND, MODIFYING, BYTE TEST</td>
</tr>
<tr>
<td>TEST 7</td>
<td>JSR TEST</td>
</tr>
<tr>
<td>TEST 8</td>
<td>MEMORY TEST</td>
</tr>
<tr>
<td></td>
<td>BOOTSTRAP PROGRAMS</td>
</tr>
<tr>
<td>CHAPTER 5.0</td>
<td>CONFIGURATION GUIDE</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>5.1</td>
<td>AN INTRODUCTION TO THE ELEMENTS OF CONFIGURATION</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Definition of Configuration</td>
</tr>
<tr>
<td>5.1.2</td>
<td>The Approach = Building An 11/34 or a BAIL</td>
</tr>
<tr>
<td>5.1.2.1</td>
<td>Defining Space</td>
</tr>
<tr>
<td>5.1.2.2</td>
<td>Transforming Power</td>
</tr>
<tr>
<td>5.1.2.3</td>
<td>DC Power</td>
</tr>
<tr>
<td>5.1.2.4</td>
<td>The Backplane Elements</td>
</tr>
<tr>
<td>5.1.2.4.1</td>
<td>Examination of a Single Edge Card Connector</td>
</tr>
<tr>
<td>5.1.2.4.1.1</td>
<td>Connector Pin Identification</td>
</tr>
<tr>
<td>5.1.2.4.2</td>
<td>Formation of a Slot</td>
</tr>
<tr>
<td>5.1.2.4.2.1</td>
<td>SRO Letter Identification</td>
</tr>
<tr>
<td>5.1.2.5</td>
<td>Formation of a Backplane</td>
</tr>
<tr>
<td>5.1.2.5.1</td>
<td>Slot Identification</td>
</tr>
<tr>
<td>5.1.2.5.2</td>
<td>SRO Identification Within A Backplane</td>
</tr>
<tr>
<td>5.1.2.6</td>
<td>The Backplane = Cable = Connector Assembly</td>
</tr>
<tr>
<td>5.1.2.7</td>
<td>The Basic System Prior to Module Placement Modules</td>
</tr>
<tr>
<td>5.1.2.8</td>
<td>Metallic Fingers</td>
</tr>
<tr>
<td>5.1.2.8.1</td>
<td>Module Placement In The DD11P</td>
</tr>
<tr>
<td>5.1.2.9</td>
<td>Function Allocation Of the DD11P</td>
</tr>
<tr>
<td>5.1.2.9.1</td>
<td>Four Type of Connector Sets</td>
</tr>
<tr>
<td>5.1.2.9.2</td>
<td>Processor Connectors</td>
</tr>
<tr>
<td>5.1.2.9.3</td>
<td>Modified UNIBUS Connectors</td>
</tr>
<tr>
<td>5.1.2.9.4</td>
<td>Placement of Dual Height, Modified Unibus Compatible Modules</td>
</tr>
<tr>
<td>5.1.2.9.4.1</td>
<td>Placement of Hex Height, Modified Unibus Compatible Modules</td>
</tr>
<tr>
<td>5.1.2.9.5</td>
<td>Standard Unibus Connectors</td>
</tr>
<tr>
<td>5.1.2.9.6</td>
<td>Small Peripheral Controller (SPC) Connectors</td>
</tr>
<tr>
<td>5.1.2.9.6.1</td>
<td>Bus Grant Continuity Cards</td>
</tr>
<tr>
<td>5.1.2.9.6.2</td>
<td>NPG Continuity Jumpers</td>
</tr>
<tr>
<td>5.1.2.9.7</td>
<td>Hex Modules</td>
</tr>
<tr>
<td>5.1.2.10</td>
<td>The Standard 11/34 Configuration</td>
</tr>
<tr>
<td>5.1.2.11</td>
<td>11/34 System Nearly Assembled</td>
</tr>
<tr>
<td>5.1.2.12</td>
<td>Module Switches</td>
</tr>
<tr>
<td>5.1.2.12.1</td>
<td>Identification of Switch Banks</td>
</tr>
<tr>
<td>5.1.2.12.2</td>
<td>Module Switch Designations</td>
</tr>
<tr>
<td>5.1.2.13</td>
<td>Module Jumpers</td>
</tr>
<tr>
<td>5.1.2.13.1</td>
<td>Identification of Jumpers</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Battery Back-Up Option</td>
</tr>
<tr>
<td>5.1.3.1</td>
<td>Three Sections</td>
</tr>
<tr>
<td>5.1.3.1.1</td>
<td>Battery Charging Circuit</td>
</tr>
<tr>
<td>5.1.3.1.2</td>
<td>Discharge Circuit</td>
</tr>
<tr>
<td>5.1.3.1.3</td>
<td>The Battery</td>
</tr>
<tr>
<td>5.1.3.2</td>
<td>Operation</td>
</tr>
<tr>
<td>5.1.3.3</td>
<td>Charge Status Indicator</td>
</tr>
<tr>
<td>5.1.3.4</td>
<td>External Battery Provision</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Elements of the BAIL</td>
</tr>
<tr>
<td>5.1.4.1</td>
<td>BAIL=BAIL Overview</td>
</tr>
</tbody>
</table>
5.1, 4.2
5.1, 4.2.1
5.1, 4.2.2
5.1, 4.2.3
5.1, 4.2.4
5.1, 4.2.5
5.1, 4.3
5.1, 4.4
5.1, 4.4.1
5.1, 4.4.2
5.1, 4.4.3
5.1, 4.4.4
5.1, 4.4.5
5.1, 4.4.6
5.1, 4.4.7
5.1, 4.4.8
5.2
5.2.1
5.2.2
5.2.2.1
5.2.2.2
5.2.2.3
5.2.2.4
5.2.2.5
5.2.2.6
5.2.2.7
5.2.2.8
5.2.2.9
5.2.2.10
5.2.2.11
5.2.2.12
5.3
5.3.1
5.3.2
5.3.2.1
5.3.2.1.1
5.3.2.2
5.3.2.2.1
5.3.2.2.2
5.3.2.2.2.1
5.3.2.2.2.2

Major Differences
Slot Capacity
Power Supply
Construction
Orientation of Modules
Battery Back Up
Similarities To The BA11L
System Units
BA11K System Unit Capacity
Dead Slots
Elements of the BA11F
BA11F Overview
Major Differences
Slot Capacity
Power Supply
Construction
Orientation & Modules
Battery Back Up Option
Similarities to the BA11L
System Units
BA11F System Unit Capacity
Slot Numbers

FACTORY CONFIGURATIONS - THE CHART SYSTEM
Introduction
The Chart System
Overview
Location of Module Placement Chart
BA11L Mounting Box
BA11K Mounting Box
The Module Placement Chart
Heading
Slot
Connector
Option
Module
CS Rev.
Etch Rev.
Bus Address
Vector
Multifunction Modules
Sample Module Placement Chart
Sample Expanded Configuration Module Placement Chart

NEW CONFIGURATIONS
What Constitutes a New Configuration?
Switches and Jumpers
Introduction
Outline of Topics
KD11E Processor
Switches
Jumpers
M7265 Data Path Module
M7266 Control Module
M9301 Terminator/ROM
Low Rom Enable Switch
Power Up Reboot Enable Switch
Address Offset Switches
Example Using Address Offset Switches
DL11W (M7856) Serial Line Interface/Real Time Clock
Switches
Overview
Address Selection of Serial Line Interface
Device Registers
Device Register Address Format
Address Switch Designations
Standard Setting
Range of Operation
Baud Rate Selection
Vector Bit Selection
Vector Bits
Standard Settings of Vector Bit Switches
Address Settings
Non Standard Settings
Vector Bit Enable
Parity Bit Enable
Parity Select
Active/Passive Modes of 2mA Current Loop Transmitter
Paper Tape Reader Enable
Receiver
Break Bit Enable
Error Bit Enable
Real Time Clock Enable
Jumpers
MS11JP (M7847-YJ) 16K MOS Memory with Parity Switches
Memory Starting Address Selection
Range Of Starting Address Selection
Capacity of 16K Memory
Standard Configuration
Expanded Configuration
Non-Standard Starting Addresses
I/O Page Protection Switches
Switch Settings
Jumpers
Jumper Designations
Jumper Placement
MM11DP (G653/H222) Core Memory with Parity Switches
Jumpers
Starting Address Selection
Interleaving Control Jumper
Definition of Interleaving
Interleave Jumpers
I/O Page Protection Jumpers
Strobe Range Control Jumpers
M9302 Bus End Terminator
M7850 Parity Controller
Switches
Jumpers
CSR Address
Standard Settings
Expanded Systems
Backplanes
Introduction
The Processor Backplane = DD11PK
Anatomy of A DD11PK Backplane
Processors Slots
Standard Unibus Connectors
Modified Unibus Connectors
Modified Vs Standard Unibus
SPC Connectors
Bus Grants
NPG
Expanding a System Within A DD11PK Backplane
General Purpose Expander Backplanes
Definition
Overview
DD11DK
DD11CK
Dedicated Backplane
Definition
Typical Options Which Require A Dedicated Backplane
Dedicated Backplane Documentation
Expanding The Basic BA11K PDP 11/34-11/39 System
Introduction
Expanded System = Factory Shipped
User Expanded System
Interconnecting Backplanes
Additional Connectors
Introduction
Power Connectors
Placement Of Power Connectors
Operators Console Connector
Remote Power Control Connector
Power Supply Loading Considerations
Introduction
BA11L Power Considerations
BA11L = H777 Power Supply Capabilities
Calculating System Power Requirements
Battery Back Up Loading
BA11K Power Considerations
BA11K = H765 Power Supply Capabilities
Calculating BA11K System Power Requirements
<table>
<thead>
<tr>
<th>FIG.</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-1</td>
<td>The Wire Frame</td>
</tr>
<tr>
<td>5-2</td>
<td>H777 Power Supply In The Wire Frame</td>
</tr>
<tr>
<td>5-3</td>
<td>DC Distribution Board</td>
</tr>
<tr>
<td>5-4</td>
<td>The Backplane</td>
</tr>
<tr>
<td>5-5</td>
<td>One Edge Card Connector</td>
</tr>
<tr>
<td>5-6</td>
<td>Connectors Formation Of A Slot</td>
</tr>
<tr>
<td>5-7</td>
<td>Connector Slot Designators</td>
</tr>
<tr>
<td>5-8</td>
<td>A Stack Of Slots</td>
</tr>
<tr>
<td>5-9</td>
<td>Slot Numbers</td>
</tr>
<tr>
<td>5-10</td>
<td>Connector Designations Within A Backplane</td>
</tr>
<tr>
<td>5-11</td>
<td>DD11PK Backplane-Cable-Connector Assembly</td>
</tr>
<tr>
<td>5-12</td>
<td>The Basic System Prior To Module Placement</td>
</tr>
<tr>
<td>5-13</td>
<td>Modules Of Various Heights</td>
</tr>
<tr>
<td>5-14</td>
<td>DD11PK Backplane Connector Allocation</td>
</tr>
<tr>
<td>5-15</td>
<td>The Standard 11/34 Configuration</td>
</tr>
<tr>
<td>5-16</td>
<td>The Basic 11/34 System After Module Placement</td>
</tr>
<tr>
<td>5-17</td>
<td>Pictorial Of The Standard 11/34 System</td>
</tr>
<tr>
<td>5-18</td>
<td>Typical Switch Banks</td>
</tr>
<tr>
<td>5-19</td>
<td>Location Of Switch Banks On Various Modules</td>
</tr>
<tr>
<td>5-20</td>
<td>Typical Jumpers</td>
</tr>
<tr>
<td>5-21</td>
<td>Jumper Location and Designation on Various Modules</td>
</tr>
<tr>
<td>5-22</td>
<td>The Battery Back Up Option</td>
</tr>
<tr>
<td>5-23</td>
<td>The BA11K Mounting Box</td>
</tr>
<tr>
<td>5-24</td>
<td>DD11PK Orientation In The BA11K Mounting Box</td>
</tr>
<tr>
<td>5-25</td>
<td>BA11F Mounting Box</td>
</tr>
<tr>
<td>5-26</td>
<td>DD11-PK Orientation In The BA11F Mounting Box</td>
</tr>
<tr>
<td>5-27</td>
<td>Module Placement Chart</td>
</tr>
<tr>
<td>5-28</td>
<td>Module Placement Chart Sample</td>
</tr>
<tr>
<td>5-29</td>
<td>Expanded Configuration Module Placement Chart Sample</td>
</tr>
<tr>
<td>5-30</td>
<td>M9301 Switch Settings</td>
</tr>
<tr>
<td>5-31</td>
<td>DL11W Device Register Format</td>
</tr>
<tr>
<td>5-32</td>
<td>DL11W Vector Bit Format</td>
</tr>
<tr>
<td>5-33</td>
<td>Battery Back Up Time Vs MOS Memory Size</td>
</tr>
<tr>
<td>5-34</td>
<td>DD11-DK Backplane</td>
</tr>
<tr>
<td>5-35</td>
<td>DD11-CK Backplane</td>
</tr>
<tr>
<td>5-36</td>
<td>Interconnecting Backplanes With Cables</td>
</tr>
<tr>
<td>5-37</td>
<td>DD11-PK Connectors</td>
</tr>
<tr>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5-1</td>
<td>M9301 Bit #=Switch # Correlation</td>
</tr>
<tr>
<td>5-2</td>
<td>DLI1W Address Switch Designations</td>
</tr>
<tr>
<td>5-3</td>
<td>Standard Setting of Address Switches</td>
</tr>
<tr>
<td>5-4</td>
<td>Baud Rate Selection</td>
</tr>
<tr>
<td>5-5</td>
<td>Correlation Of Vector Bits and Switches</td>
</tr>
<tr>
<td>5-6</td>
<td>Standard Vector Bit Switch Settings</td>
</tr>
<tr>
<td>5-7</td>
<td>Number of Data Bit Select Switches</td>
</tr>
<tr>
<td>5-8</td>
<td>Number of Stop Bit Switches</td>
</tr>
<tr>
<td>5-9</td>
<td>Parity Bit Enable Switch</td>
</tr>
<tr>
<td>5-10</td>
<td>Odd-Even Parity Select Switch</td>
</tr>
<tr>
<td>5-11</td>
<td>Transmitter Mode Switches</td>
</tr>
<tr>
<td>5-12</td>
<td>Paper Tape Reader Enable Mode</td>
</tr>
<tr>
<td>5-13</td>
<td>Receiver Mode Switches</td>
</tr>
<tr>
<td>5-14</td>
<td>Break Bit Switch</td>
</tr>
<tr>
<td>5-15</td>
<td>Error Bit Enable Switch</td>
</tr>
<tr>
<td>5-16</td>
<td>Real Time Clock Enable Switch</td>
</tr>
<tr>
<td>5-17</td>
<td>M7847 Starting Address Switch Setting Of First Memory</td>
</tr>
<tr>
<td>5-18</td>
<td>M7847 Starting Address Switch Settings</td>
</tr>
<tr>
<td>5-19</td>
<td>M7847 Normal I/O Page Protect Switch Settings</td>
</tr>
<tr>
<td>5-20</td>
<td>M7847 Top Of Memory I/O Page Protect Switch Settings</td>
</tr>
<tr>
<td>5-21</td>
<td>MM11DP Address Zero Setting</td>
</tr>
<tr>
<td>5-22</td>
<td>MM11DP Starting Address Switch Settings</td>
</tr>
<tr>
<td>5-23</td>
<td>M7850 Standard Jumper Settings - Address Zero</td>
</tr>
<tr>
<td>5-24</td>
<td>M7850 CSR Address Select Jumpers</td>
</tr>
<tr>
<td>5-25</td>
<td>DDI1PK Slot 1 Pin Designation</td>
</tr>
<tr>
<td>5-26</td>
<td>DDI1PK Slot 2 Pin Designation</td>
</tr>
<tr>
<td>5-27</td>
<td>Standard Unibus Pin Designations</td>
</tr>
<tr>
<td>5-28</td>
<td>SPC Pin Designations</td>
</tr>
<tr>
<td>5-29</td>
<td>Power Connector Pin Listings</td>
</tr>
<tr>
<td>5-30</td>
<td>Operator's Console Connector Pin Listings</td>
</tr>
<tr>
<td>5-31</td>
<td>BAII-L = H777 Output Current Power Capabilities</td>
</tr>
<tr>
<td>5-32</td>
<td>Module &amp; Option Power Requirements</td>
</tr>
<tr>
<td>5-33</td>
<td>BAII-L Power Consideration Worksheet</td>
</tr>
<tr>
<td>5-34</td>
<td>BAII-K Output Current - Power Capabilities</td>
</tr>
<tr>
<td>5-35</td>
<td>BAII-K-H765 Power Consideration Worksheet</td>
</tr>
</tbody>
</table>
5.0 CONFIGURATION GUIDE

This guide is divided into four major parts:

- 5.1 An Introduction to the Elements of Configuration
- 5.2 Factory Configuration - A Description of the Chart System
- 5.3 New Configurations - The Significance of Module Switches, Jumpers and Location, Connectors, Power Considerations
- 5.4 Configuration Log - A Place to Log the Configuration History of your Machine.

5.1 AN INTRODUCTION TO THE ELEMENTS OF CONFIGURATION

5.1.1 Definition of Configuration

Configuration is the arrangement of the electromechanical elements of the computer. These elements include:

- Mounting Boxes
- Power Supplies
- Backplane
- Cables
- Connectors
- Modules
- Switches
- Jumpers

Each of these elements will be discussed within the following text.

5.1.2 The Approach - Building an 11/34 in a BA11L

The most natural way to approach the elements of a computer is to build a computer from scratch. The discussion that follows will build a computer into a BA11L mounting box. (A 5 1/4" high x 17 5/8" wide x 25" deep chassis.)

5.1.2.1 Defining Space

A rectangular wire frame (see Figure 5-1) is the foundation of the computer. All other elements will mount directly or indirectly to this frame.
5.1.2 2. Transforming Power

The AC power from the wall outlet must be transformed into low-voltage direct current for the electronic elements. The H777 power supply fits into the wire frame (Figure 5-2) and transforms the AC line voltage into several levels of low voltage direct current.

5.1.2 3. DC Power

The DC power from the power supply connects to the DC distribution board which is located on the power supply. The DC distribution board contains a bank of connector sockets (Figure 5-3) which will be connected to other elements of the computer later in this discussion.
5.1.2 4. The Backplane Elements

The Backplane consists of a matrix of single rectangular openings, which are called Single Edge Card Connectors (SECC). See Figure 5-4.

5.1.2 4.1 Examination of A Single Connector

Each connector serves as an electro-mechanical interface, between the mounting box and the modules. This interface performs two distinct functions:

1. It is the mechanical holder that holds the module by a press fit.

2. It is an electrical interface that connects to the power supply and to other connectors via two rows of spring pins which line the sides of the connector.
5.1.2 4.1.1 Connector Pin Identification

Figure 5-5 illustrates the pin identification system. Any pin can be uniquely identified by referencing its side and pin letter. The top leftmost pin would be referred to as A1. The bottom rightmost pin is V2.

![Diagram of connector and pin side](image)

**FIGURE 5-5 EDGE CARD CONNECTOR**

5.1.2 4.2 Formation of a Slot

A slot can be formed by placing six Single Edge Card Connectors (SECC) end to end as shown in Figure 5-6.

![Diagram of SECC connectors](image)

**FIGURE 5-6 FORMATION OF SLOT**

5.1.2 4.2.1 Connector Location Identification

Once a slot is formed we can assign a connector letter designator to each connector. See Figure 5-7.
| A | B | C | D | E | F |

FIGURE 5-7 SLOT DESIGNATORS
5.1.2 5. Formation of a Backplane

If we stack a number of slots we will form a backplane (Figure 5-8). The two standard backplane sizes are four slot stacks and nine slot stacks.

![A Stack of Slots](image)

**FIGURE 5-8 A STACK OF SLOTS**

5.1.2 5.1 Slot Identification

The top slot will be designated as number one. All other slots will be numbered sequentially as shown in Figure 5-9.
5.1.2 5.2 Connector Identification within a Backplane

Any connector can be uniquely identified by referring to its slot number and letter designator (Figure 5-10).
5.1.2 6. The Backplane=Cable=Connector Assembly

The set of wires that provide DC power to the backplane exit from the backplane to a set of rectangular plastic cable connectors (Figure 5-11). These connectors plug directly into the DC distribution board of the power supply.
FIGURE 5-11  DD11-PK BACKPLANE CABLE CONNECTOR ASSEMBLY
5.1.27. The Basic System — Prior to Module Placement

Mounting the backplane to the wire frame and connecting it to the power supply (Figure 5-12) enable us to insert modules into the system.
FIGURE 5-13 MODULES OF VARIOUS HEIGHTS
5.1.2 8. Modules

Modules are printed circuit cards which contain the electronic circuits which comprise the computer. Figure 5-13 illustrates modules of various heights.

Note:

Single Height Modules require 1 connector
Double Height Modules require 2 Horizontally adjacent connectors
Quad Height Modules require 4 Horizontally adjacent connectors
Hex Height Modules require 6 Horizontally adjacent connectors (1 full slot)

5.1.2 8.1 Metallic Fingers

The edge of each module is lined with a row of gold plated fingers which come in contact with the spring metal pins of the connector. These metallic fingers and spring pins provide power to the module as well as signal interconnections.

5.1.2 9. Module placement in the DD11-P

Before we begin discussing module placement, it should be noted that the DD11-P, 9-slot backplane is used in both the BA11L and BA11K mounting boxes.

5.1.2 9.1 Functional Allocation of the DD11-P

Figure 5-14 illustrates the connector allocation of the DD11-P backplane when used in an 11/34 computer system.

5.1.2 9.2 Four Types of Connector Groups

The DD11-P offers four distinct types of dedicated connector groups:

- Processor
- Modified Unibus
- Standard Unibus
- SPC (Small Peripheral Controller)

5.1.2 9.3 Processor Connectors

All of the connector in slots 1 and 2 are dedicated to accepting the
two module KU11E processor. Special connections between slots one and two are present. The processor can not work in any other slots.

5.1.2 9.4 Modified Unibus Connectors

Connectors 3A-3B, 4A-4B, 5A-5B, 6A-6B, 7A-7B, and 8A-8B comprise six sets of modified unibus connectors. Two distinct types of boards fit into modified unibus slots: dual height modules, which require two adjacent connectors within one slot; and hex height modules, which utilize an entire slot. (See Table 5-27 for standard and modified Unibus pinouts.)
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>M7266 PROCESSOR MODULE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>M7265 PROCESSOR DATA PATH MODULE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MODIFIED UNIBUS MUST BE M9301</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MODIFIED UNIBUS</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MODIFIED UNIBUS</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MODIFIED UNIBUS</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MODIFIED UNIBUS</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>MODIFIED UNIBUS</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>STANDARD UNIBUS MUST BE M9302 OR UNIBUS CABLE</td>
<td></td>
<td>SPC*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Must contain Module or BG card
** Wire Jumper CA1-CB1 must be removed if an MPR device is used.

**FIGURE 5-14 NN11-PK BACKPLANE CONNECTOR ALLOCATION**
At the time of this writing there are two dual height, modified unibus compatible modules available:

- M9301 Terminator/ROM
- M7850 Memory Parity Controller

There are currently two modified unibus compatible memory modules available:

- MS11 16K MOS Memory
- MM11DP 16K CORE Memory

5.1.2 9.4.1 Placement of Dual Heights, Modified Unibus Compatible Modules

M9301 Terminator/ROM: can be placed in any modified unibus connector but should be as close to the CPU as possible, typically 3A and 3B.

M7850 Parity Controller: can be placed in any modified unibus connector.

5.1.2 9.4.2 Placement of Hex Width Modified Unibus Compatible Modules

MS11 16K MOS Memory: can be placed in any slot from 3 through 9 (4 is preferred).

MM11DP 16K CORE Memory: this board requires two slots. It can be placed in any slot from 3 through 7.

5.1.2 9.5 Standard Unibus Connector Groups

Connectors 9A and 9B are pinned for standard unibus. Two cases occur in the discussion at this time:

1. The DD11-P is the only backplane in the system, as it is if the BA11-L box is the only mounting box in the system.

2. Other backplanes in the system must be serviced by the processor's mounting box.

If case 1 applies then connectors 9A and 9B must contain an M9302 bus end terminator.

If case 2 applies, the connectors 9A and 9B must have a BC11A unibus jumper cable between backplanes. The last Unibus slots must contain an M9302 terminator.
The M9302 Bus End Terminator

All 11/34 computer systems must contain an M9302 Bus End Terminator at the end of the bus. This terminator causes BG's which reach the end of the bus to return SACK to the CPU. This terminator is placed in the last available standard unibus slot.

5.1.2 9.6 Small Peripheral Controller (SPC) Connectors

DEC offers a wide variety of peripheral options which are pinned to a SPC interface standard. These options include terminals, clocks, the high speed read/punch, floppy disk, or cassette controllers, etc. SPC slots are available in slots 3 through 9.

5.1.2 9.6.1 Bus Grant (BG) Continuity Cards

Within the SPC slots shown on Figure 5-13, connectors 3D, 4D, 5D, 6D, 7D, 8D, and 9D contain a note: "BG Card or Module". This means that each of these connectors must contain either by a SPC module or a BG continuity card. If a BG card or module is not supplied the machine will not function properly.

5.1.2 9.6.2 NPG Continuity Jumpers

NPG jumpers are wired into the DD11 P backplane, connectors 3C-9C. If a module utilizing NPG and NPG is added to the system, the wire pertaining to that particular connector must be removed. See section 5.3.3. 2.14.1 for a detailed description of these jumpers.

5.1.2 9.7 Hex Modules

Hex modules occupy an entire slot (all six connectors).

If a hex module is designed to be SPC compatible, it will pick up its signals from the SPC slots and modified unibus signals, with the exception of power and grounds.

If a hex module is designed to be modified unibus compatible, it will utilize the modified unibus signals and ignore the SPC signals with the exception of power, ground, bus grant and NPG signals.

5.1.2 10. The Standard 11/39 Configuration

Figure 5-15 illustrates the standard 11/3 configuration with MOS memory.
Figure 5-16 pictorially illustrates the configuration shown in Figure 5-14.
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>7266 Control Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>7265 Data Path Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>M9301 Terminator ROM</td>
<td>7856 Serial Line Interface/Real Time Clock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>7847-YJ 16K MOS Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>7850 Parity</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>M9302 Terminator</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5-15**  Standard 11/34 Configuration

**Figure 5-16**  Basic 11/34 System After Module Placement
5.1.2 11. 11/34 System Nearly Assembled

Figure 5-17 shows the standard 11/34 configuration partially in its protective housing with the fan and operator's console, card guide and other accessories in view.

FIGURE 5-17 PICTORIAL OF THE STANDARD 11/34 SYSTEM

5.1.2 12. Module Switches

Several of the modules which comprise the basic 11/34 computer systems contain packs of small switches. Figure 5-18 shows three types that are commonly used. Note that all the switches have the switch numbers identified adjacent to each switch. All switches shown in Figure 5-18 are illustrated in the "on" position.

FIGURE 5-18 TYPICAL SWITCH PACKS
DL11W (M7856) SERIAL LINE INTERFACE/REAL TIME CLOCK

M9301 (M9301) TERMINATOR BOOT

MS11J M7847

FIGURE 5-19 LOCATION OF SWITCH PACKS ON VARIOUS MODULES
5.1.2.1 Identification of Switch Packs

Each switch pack on a module will be assigned a number. Figure 5-19 illustrates the:

- M9301 Bootstrap/Terminator
- DL116 Asynch, Ser Line Interface/Line clock
- M7847 16K MOS Memory

This illustration designates the switch pack number of each module.

5.1.2.2 Module Switch Designations

Each module switch will be designated by its pack number followed by the switch number.

Example: Switch Bank #2, Switch #1 on the DL11 W is to be discussed. How would it appear in the discussion?

B2-S1

5.1.2.3 Module Jumpers

Many of the modules used in the 11/34 computers contain provisions for modification of module performance by the installation, removal, or relocation of small wire jumpers.

Changing these semi-permanent jumpers requires a competent technician with soldering equipment. Figure 5-20 illustrates typical jumper arrangements with the jumper wires present.

![Figure 5-20 Typical Jumpers](etch_jumper-terminal_post_jumper.png)
FIGURE 5-21 JUMPER LOCATION OF DESIGNATION ON VARIOUS MODULES to be supplied
5.1.2 Identification of Jumpers

Wire Jumpers are usually identified by the prefix "W" followed by a number. Figure 5-21 illustrates the location and number of jumpers on the following modules:

KD11E Processor (M7265), Data Path (M7266), Control
DL11 W (M7856) Serial Line Interface/Real Time Clock
M9301 (M9301) Terminator ROM
MS11JP (M7847) 16K MOS Memory
MM11D (G652-H222) 16K CORE Memory

Example: The designation of the only jumper on the DL11 W is W1.

5.1.3 Battery Back-Up Option

5.1.3.1 Three Sections

The H775 Battery Back-Up system (Fig. 5-22) consists of three sections: a charging circuit, a discharge circuit and the battery.

5.1.3.1.1 Battery Charging Circuit

The charging circuit, basically, takes an input voltage of 24-48 VDC and switches it up to 45-48 VDC. This output voltage is then used to drive a current source of approximately 0.5 amps to charge the battery. The battery voltage is sensed and, at a preset voltage of 31.5 volts, the circuit is switched to a trickle charge rate of 50 mA. This trickle charge is maintained until the battery is charged.

5.1.3.1.2 Discharge Circuit

The discharge circuit consists of an SCR and a voltage sensing trigger circuit. The input voltage to the module is normally 24-48 VDC. If the line should fall, this voltage will drop below its normal range. When this voltage drops to 20V, the sense circuit allows the trigger circuit to fire the SCR, and the battery will discharge into the load.

5.1.3.1.3 The Battery

The battery itself is two 12V, 5A-HR batteries tied in series for 24V. It is a sealed lead-acid system with a very low internal impedance, in the order of 250 milliohms or less. It will support 32K of MOS memory
for 2 HRS minimum and, with the above charging circuit, will recharge in 14-16 HRS.
Fig 5-22

THE BATTERY BACK-UP OPTION
5.1.3 2. Operation

Two additional signals are required for operation. The +5V output from the MOS regulator and the Key Enable line are used to drive a relay which completes the battery circuit. In the event that the battery drains too low to support the memory, the 5V output of the regulator drops down, opening up the relay, and, hence, no leakage current can flow from the battery. Key Enable also tells the computer that the battery module is present and operational.

5.1.3 3. Charge Status Indicator

One output signal is provided. This is the Charge Status Indicator. If this signal is low, it means that the battery is on trickle charge and is in the vicinity of 100% charged.

5.1.3 4. External Battery Provision

External battery terminals are provided so that an additional battery can be plugged in to extend the holdup time to whatever is desired. No charging is provided for this battery, but it neither affects nor is affected by the charging state or condition of the internal battery.

5.1.4 Elements of the BA11K

5.1.4 1. BALLK-BA11L Overview

The BA11K box contains the same type of elements as those in the BA11L. In many instances these elements are identical to those used in the BA11L.

5.1.4 2. Major Differences

5.1.4 2.1 Slot Capacity

The BA11K is twice the height of the BA11L with the modules held vertically to contain the equivalent of 5 system units. This allows the addition of optional backplanes (the DD11-P is standard). A maximum of 13 additional slots can be obtained by the addition of a 4 slot and a 9 slot backplane.
5.1.4 2.2 Power Supply

The H765 power supply is utilized in the BA11K to fulfill the power requirements of the BA11K. This power supply is capable of powering the basic PDP-11/34 system, including optional modules which may be used in the DD1 P, plus any optional backplanes and their associated modules. Paragraph 5.3.5 of this guide describes power supply loading and should be consulted prior to expanding or reconfiguring a system.

5.1.4 2.3 Construction

The BA11K uses a standard sheet metal chassis as opposed to the wire frame construction of the BA11L. When access to the BA11K modules is required, the top cover is removed (Figure 5-23).
BA11-K Mounting Box (Top Covers Removed)
5.1.4 2.4 Orientation of Modules

Modules are oriented in the vertical plane in the BA11K. The slot and connector designations are shown in Figure 5-24.

![Diagram of BA11K orientation](image)

**FIGURE 5-24 DD11PK ORIENTATION IN THE BA11K MOUNTING BOX (Top View)**

5.1.4 2.5 Battery Back-Up Option

This option is not currently available for the BA11K mounting box.

5.1.4 3. Similarities to the BA11L

BA11K versions of the PDP-11/34 systems use the following identical elements:

- DD11PK Backplane-Cable Assembly
- KY11 LA Operator's Console
- KD11E Processor Module Set
- M9301 Terminator/ROM
- M9302 Bus End Terminator
- DL11W Serial Line Interface
- MS11JP 16K MOS Memory
- MM11DP 16K CORE Memory

5.1.4 4. System Units

System Units refer to the standard sizes of DEC backplanes.

All 4 slot backplanes equal 1 system unit (1SU). (Sometimes called single system units).

All 9 slots backplanes equal 2 system units (2SU), sometimes referred to as a double system unit.
FIGURE 5-26  DD11PK ORIENTATION IN THE BA11F MOUNTING BOX
5.2 FACTORY CONFIGURATIONS

5.2.1 Introduction

This section deals with the system of documentation which describes the configuration of the user's computer when it is shipped from the factory. This information is valuable to the user from several perspectives:

1. System configuration can be checked quickly and unambiguously.

2. When a module is removed from the system, it is easy to determine where it belongs when the user wishes to reinsert it.

3. If the system is reconfigured (switches changed, modules moved, etc.) and problems are experienced, the user can return to the original, tested configuration.

5.2.2 The Chart System

5.2.2.1 Overview

All 11/34 computer systems contain a module placement chart that describes which slot each module plugs into and also describes additional module information.
<table>
<thead>
<tr>
<th>PDP-11/34</th>
<th>Serial No.</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slot</td>
<td>Conn</td>
</tr>
</tbody>
</table>

Figure 5-27  Module Placement Chart
5.2.2  Location of Charts

5.2.2 2.1 BA11L Mounting Box

The module placement chart is located on the top of the power supply cover.

5.2.2 2.2 BA11K Box

The module placement chart is located on the outside of the top cover plate of the mounting box.

5.2.2 3. The Module Placement Chart

Fig. 5-27 illustrates the Module Placement Chart.

5.2.2 3.1 Heading

The upper portion of this chart identifies the chart, serial number of the machine that the chart represents, and the date the machine configuration was determined and verified.

5.2.2 3.2 Slot

This column specifies which slot in the backplane is being described. Slots 1 and 2 must contain the processor and appear only once. Slots 3 - 9 are printed more than once to describe split slots. Split slots utilize more than one module in a slot.

5.2.2 3.3 CONN

CONN describes which connectors a module occupies. Only the first and last connectors of a module are specified. A hex board would have AF appear in this column.

5.2.2 3.4 Option

Option describes the DEC option letter designation of a functional block, which can contain one or more modules. It is possible for an option to occupy several slots.
5.2.2 3.5 Module

This specifies the designator of the actual module which occupies the slot and connector which appear on the same line of the chart.

5.2.2 3.6 Cs REV

This represents the circuit schematic revision letter of the module.

5.2.2 3.7ETCH REV

Each module has a revision letter that describes the printed circuit board's Etch revision status.

5.2.2 3.8 Address

The main address of each module will appear in the address column.

5.2.2 3.9 Vector

The vector of the module will be written in this column.

5.2.2 3.10 Multifunction Modules

If a single module combines two completely separate functions, the module will occupy two lines. The address and vectors can then be stated for each function, when required.

5.2.2 3.11 Sample Module Placement Chart

Figure 5-28 illustrates a sample chart for the standard components for an 11/34 system.
### Module Placement Chart Sample

**FIGURE 5-28**

<table>
<thead>
<tr>
<th>SLOT</th>
<th>COMB</th>
<th>OPTION</th>
<th>MODULE</th>
<th>CS REV</th>
<th>ECH REV</th>
<th>BUS ADDRESS</th>
<th>VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AF</td>
<td>KDIIE</td>
<td>M7266</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AF</td>
<td>KDIIE</td>
<td>M7265</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AB</td>
<td>BM9301</td>
<td>M9301</td>
<td>R</td>
<td>F</td>
<td>773024</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CF</td>
<td>DLIIW(sc)</td>
<td>M7865</td>
<td>A</td>
<td>A</td>
<td>777560</td>
<td>60/64</td>
</tr>
<tr>
<td>3</td>
<td>CF</td>
<td>DLIIW(perc)</td>
<td>M7865</td>
<td>A</td>
<td>A</td>
<td>777546</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>AF</td>
<td>MSIIJP</td>
<td>M7847</td>
<td>A</td>
<td>A</td>
<td>0-017776</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>AB</td>
<td>M7850</td>
<td>A</td>
<td>A</td>
<td></td>
<td>772100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>AB</td>
<td>M9302</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**5.2.2 3.12 Sample Expanded Configuration Module Placement Chart**

Figure 5-29 illustrates a module placement chart, when an expanded configuration system is employed.
### FIGURE 5-29 EXPANDED CONFIGURATION MODULE PLACEMENT CHART SAMPLE

<table>
<thead>
<tr>
<th>SLOT</th>
<th>CONN</th>
<th>OPTION</th>
<th>MODULE</th>
<th>CS REV</th>
<th>CS REV</th>
<th>ADDRESS</th>
<th>VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CF</td>
<td>LP II</td>
<td>M7930</td>
<td>D</td>
<td>D</td>
<td>777514</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>CF</td>
<td>KWI D</td>
<td>M7228</td>
<td>H</td>
<td>E</td>
<td>772540</td>
<td>104</td>
</tr>
<tr>
<td>21</td>
<td>CF</td>
<td>RKII D</td>
<td>M7254</td>
<td>H</td>
<td>C</td>
<td>777400</td>
<td>220</td>
</tr>
<tr>
<td>22</td>
<td>CF</td>
<td>RKII D</td>
<td>M7255</td>
<td>L</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>CF</td>
<td>RKII D</td>
<td>M7256</td>
<td>H</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CF</td>
<td>RKII D</td>
<td>M7257</td>
<td>H</td>
<td>J</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 NEW CONFIGURATIONS

5.3.1 What Constitutes A New Configuration?

Any time a:
- Module switch setting is changed
- Jumper is added, removed or shifted
- Module is added or removed
- Backplane is added or removed,
a new configuration is generated.

This section should be consulted before the user attempts to generate the new configuration.

5.3.2 Switches and Jumpers

5.3.2.1 Introduction

This section deals with the switches and jumpers located on the standard modules of PDP11/34 systems. These topics have been combined because these elements are sometimes used in combination to produce a desired result. In many instances these elements are used exclusively of each other and will be described separately. The following discussion will examine each of the standard system modules individually. Switch locations are illustrated in Figure 5-19, jumper location in Figure 5-21.

5.3.2.1.1 Outline Of Topics

The modules will be presented as follows:

- KD11E processor (2 modules)
- M9301 Terminator/ROM
- DL11W Serial Line Interface/Real Time Clock
- MS11DP 16K MOS Memory (with Parity)
- MM11DP 16K Core Memory (with Parity)
- M9302 BUS End Terminator
- M7850 Parity Controller

5.3.2.2 KD11E Processor

5.3.2.2.1 Switches

Neither of the processor modules (M7265, M7266) contains any switches.

5.3.2.2.2 Jumpers

5.3.2.2.2.1 M7265 - Data Path Module

This module contains two jumpers W1 and W2 which are always present.
These jumpers determine the speed of the processor clock and must not be removed.

5.3.2 2.2.2 M7266 - Control Module

Two jumpers W1 and W2 determine the capability of the processor to detect parity errors. All KDIIE processors will have W1 absent and W2 present to allow the CPU to detect parity errors (signaled by the M7850 parity controller module). The jumper is switched only for unique situations such as module repair.

5.3.2 3. M9301 Terminator/ROM

5.3.2 3.1 Switches

The M9301 contains one switch pack (P1) of 10 switches (S1-S10). Figure 5-30 illustrates the significance of these switches.

5.3.2 3.1.1 Low ROM Enable Switch (P1-S1)

When placed in the off position, this switch disables the lower 256 words of the M9301 ROM. It is intended for use in M9301 variations other than the YA and YB.

This switch should always be in the ON position on 11/34 systems.

5.3.2 3.1.2 Power Up Reboot Enable Switch (P1-S2)

With this switch in the on position, the system will reboot into the M9301 diagnostic and console routine on powering up from a power failed condition. With this switch in the off position, power up will occur normally through vector locations 24 and 26. The boot switch on the console causes a reboot as described for either switch position.

NOTE

MOS configurations without battery backup should have this switch in the ON position. Core or MOS in the battery backup configurations, should have the switch in the OFF position.
Figure 5-30

M9301 Switch Pack

Key:
1 = Fixed One
0 = Fixed Zero
A = Switchable bit which will be a ONE if its corresponding switch is in the OFF position in the M9301 YA & YB versions.

Contents of M9301 ROM Location 173024 (It will always be 173000 in YA & YB versions)

Starting address - This value is determined by the resultant of a "Logical OR" operation.

Switch setting of M9301 switches P-3 to P-10

If any switch is in the ON position, the corresponding Address Bit will be a ONE.
5.3.2 3.1.3 Address Offset Switches (P1=S3 to P1=S10)

These eight switches are logically combined (via a "Logical OR" operation)* with the contents of ROM Address 773024 (which always contains 1730000) during the power up routine, **or when the power switch is moved, **or when the BOOT/INIT switch is depressed. (These are boot routine entries.)

ROM Address 773024 is obtained by ORing 773000 (from the M9301) and 24 (generated by the processor) on the UNIBUS address lines at the time of the power up sequence. This address is then used as the power up vector by the CPU, and will contain the PC which is determined by the address, the offset switches (P1=S3 to P1=S10). This word is the new PC, the CPU will execute its first instruction from that location.

These switches allow access to the locations within the M9301 ROM address space 1730000-173776. The purpose of these switches is to allow the user to enter various ROM routines upon a boot. These routines are discussed in the M9301 software guide.

Figure 5.30 illustrates the positional significance of each switch.

It should be noted the standard setting for all offset switches is in the ON position.

5.3.2 3.1.3.1 Example Using Address Offset Switches

A user wants the program counter to start executing the program in location 173254 whenever a boot entry occurs (in an M9301 YA or YB).

The 173 portion of the program count is always fixed, as is the least significant bit.

The switches affect the last three octal numbers.

To generate a 254 in octal, the following bit pattern must be generated:

<table>
<thead>
<tr>
<th>Octal Number:</th>
<th>2</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern:</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Bit Number:</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

Nine bits are required but bit #0 is always a zero, so it is possible to produce all 3 octal numbers (odd addresses are prohibited).

* Logical or operation is defined (in this case) as the condition of having two inputs which produce one output; if either input is a one, the output will be a one.

** Unless power up reboot enable switch is in the OFF position.
We examine which bit numbers are to be ones and zeros and find bits 8, 6, 4, 1 and 0 are to be zeros. Bits 7, 5, 3, and 2 are to be ones.

We then go to Figure 5-30 and correlate the bit numbers with the switch number and set the switches to be to the off position, as shown in Table 5-1.

<table>
<thead>
<tr>
<th>BIT #</th>
<th>SWITCH #</th>
<th>SETTING FOR 254</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>0 = ON</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1 = OFF</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>0 = ON</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1 = OFF</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0 = ON</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1 = OFF</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>1 = OFF</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0 = ON</td>
</tr>
</tbody>
</table>

5.3.2 3.2 Jumpers

Five jumpers (W1, W2, W3, W4, W5) are always absent in all M9301 modules used in PDP 11/34 Systems. These jumpers allow the user to terminate Bus Grant lines when the M9301 is used with processor modules that do not contain this feature (11/05, 11/10, 11/20, etc).

5.3.2 4. DL11W M7856 Serial Line Interface/Real Time Clock

5.3.2 4.1 Switches

5.3.2 4.1.1 Overview

The DL11W contains 5 switch packs which allow the user to:

- Locate the serial line interface registers from 774000 to 777770
- Select the baud rate
- Change the vector
- Select the number of data bits to be transmitted or received.
- Select the number of stop bits
- Specify parity or non-parity operation
. Specify odd or even parity if parity is specified
. Set active or passive modes of 20MA current loop operation independently for transmitter and receiver interface circuits
. Enable or disable the break function
. Enable or disable error bits
. Enable or disable the real time clock

Switches are not necessarily clustered in one switch pack for each function. The following discussion examines the DL11W by switch selectable function.

5.3.2 4.1.2 Address Selection of Serial Line Interface
Device Registers

5.3.2 4.1.2.1 Device Register Address Format

Figure 5-31 specifies the device register address bits.

<table>
<thead>
<tr>
<th>Bit #</th>
<th>17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>1 1 1 1 1 1 1 X X X X X X X Y Y Y</td>
</tr>
<tr>
<td></td>
<td>FIXED SWITCH SELECTABLE Φ = RCSR</td>
</tr>
<tr>
<td></td>
<td>2 = RBUF</td>
</tr>
<tr>
<td></td>
<td>4 = XCSR</td>
</tr>
<tr>
<td></td>
<td>6 = XBUF</td>
</tr>
</tbody>
</table>

KEY: 1 = 1
Φ = 0
X = Switch Selectable Bit
Y = Software selectable, determines which device register is accessed

RCSR = Receiver Control and Status Register
XCSR = Transmitter Control and Status Register
RBUF = Receiver Buffer Register
XBUF = Transmitter Buffer Register

X SWITCH BIT CONVENTION
If a switch in this register is OFF, X = 1.
If a switch in this register is ON, X = 0.

FIGURE 5-31 DL11W DEVICE REGISTER ADDRESS FORMAT
5.3.2 4.1.2.2 Address Switch Designations (see Table 5-2)

**TABLE 5-2 DL11W ADDRESS SWITCH DESIGNATIONS**

<table>
<thead>
<tr>
<th>ADDRESS BIT</th>
<th>PACK SWITCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

5.3.2 4.1.2.3 Standard Settings

The standard setting of the DL11W address switches is shown in Table 5-3.

**TABLE 5-3 DL11W ADDRESS SWITCHES-STANDARD SETTINGS**

<table>
<thead>
<tr>
<th>PACK SWITCH</th>
<th>SWITCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>ON</td>
</tr>
<tr>
<td>4</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>ON</td>
</tr>
<tr>
<td>5</td>
<td>OFF</td>
</tr>
</tbody>
</table>

The standard setting allows the DL11W to interface to the system terminal, addresses 777560-777566.

5.3.2 4.1.2.4 Non-Standard Settings

If only one DL11W is in a system, the user should use the standard switch setting unless a custom version of the M9301 is designed by the user.

If additional DL11W modules are added to a standard RDP11/34 system, the additional DL11w device registers must be set to respond to additional address.

To generate a non standard address, determine the desired address, write out the bit pattern, and set the corresponding switches to that pattern.
5.3.2 4.1.2.5 Range of Operation

The DEC address map suggests that the user limit selection of address space to the ranges listed in the Programmer's Guide.

5.3.2 4.1.3 Baud Rate Selection (see Table 5-4)

The baud rate for each system will be dependent on the system terminal (standard teletypes utilize 110 baud). DECwriters can use up to 300 baud.

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>TRANSMITTER</th>
<th>RECEIVER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P4=S10</td>
<td>P3=S1</td>
</tr>
<tr>
<td>110</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>150</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>300</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>600</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>1200</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>2400</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>4800</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>9600</td>
<td>OFF</td>
<td>ON</td>
</tr>
</tbody>
</table>

5.3.2 4.1.4 VECTOR BIT SELECTION

The DL11-W combines two distinct functions and must generate two distinct vectors:

- One vector for the receiver.
- One vector for the transmitter

5.3.2 4.1.4.1 Vector Bits (see figure 5-32)

<table>
<thead>
<tr>
<th>8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z Z Z Z Z Z Z</td>
</tr>
<tr>
<td>H H H</td>
</tr>
</tbody>
</table>

<h1>Switch Selectable</h1>

<h1>Hardware Generated</h1>

0 = RECEIVER
4 = TRANSMITTER

FIGURE 5-32 DL11-W VECTOR BIT FORMAT
KEY:

Z Indicates switch selectable
H Indicates that these bits are set by the hardware automatically, depending on the function being generated.

When switches are in the OFF position \( Z = 0 \)
When switches are in the ON position \( Z = 1 \)

Table 5-5 shows the correlation of the vector bits with their associated switches.

<table>
<thead>
<tr>
<th>VECTOR BIT #</th>
<th>PACK</th>
<th>SWITCH #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

5.3.2 4.1.4.2 Standard Settings Of Vector Bit Switches

Standards settings of the vector bit switches which allow operation as
the system terminal with DEC software are:

Receiver
Transmitter
vector 60
vector 64

To obtain the standard configuration set the switches as shown in
Table 5-6.

<table>
<thead>
<tr>
<th>PACK</th>
<th>SWITCH</th>
<th>SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>ON</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>OFF</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>OFF</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>ON</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>OFF</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>OFF</td>
</tr>
</tbody>
</table>

5.3.2 4.1.4.3 Non Standard Settings

Non standard settings should not be used on the only DL11W in a
standard system. If a user has a special M9301 module or several
DL11W modules, the floating address assignments table in the
Programmer's Guide should be consulted.
5.3.2 4.1.5 Data Format Selection (see Table 5-7)

5.3.2 4.1.5.1 Number Of Data Bits

<table>
<thead>
<tr>
<th>Number of Data Bits</th>
<th>P4-S4</th>
<th>P4-S3</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>ON</td>
<td>ON</td>
<td>DEC Typeset Only</td>
</tr>
<tr>
<td>6</td>
<td>ON</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>OFF</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>OFF</td>
<td>OFF</td>
<td>Normal TTY</td>
</tr>
</tbody>
</table>

5.3.2 4.1.5.2 Number of Stop Bits

Table 5-8 illustrates the switch position required to produce various numbers of stop bits.

<table>
<thead>
<tr>
<th>Number of Stop Bits</th>
<th>P4-S5</th>
<th>Normally Used On</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ON</td>
<td>LA30, VT05, VT50</td>
</tr>
<tr>
<td>2</td>
<td>OFF</td>
<td>Teletypes (with 6, 7, &amp; 8 data bits)</td>
</tr>
<tr>
<td>*1,5</td>
<td>OFF</td>
<td>Typeset Systems Only</td>
</tr>
</tbody>
</table>

*Used with 5 data bits only.

5.3.2 4.1.5.3 Parity Bit Enable (see Table 5-9)

<table>
<thead>
<tr>
<th>Parity Bit</th>
<th>P4-S6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>Disable</td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

If the parity bit is not enabled, this bit will not be transmitted.
5.3.2 4.1.5.4 Parity Select (see Table 5-10)

<table>
<thead>
<tr>
<th>Parity</th>
<th>P4-S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVEN</td>
<td>OFF</td>
</tr>
<tr>
<td>ODD</td>
<td>ON</td>
</tr>
</tbody>
</table>

5.3.2 4.1.6 ACTIVE-Passive Modes of 20MA Current Loop

5.3.2 4.1.6.1 Transmitter (see Table 5-11)

In the active mode the transmitter will supply and modulate 20MA of current to an external receiving device. In the passive mode the transmitter will only modulate an externally supplied current.

<table>
<thead>
<tr>
<th>PACK</th>
<th>SWITCH</th>
<th>ACTIVE</th>
<th>PASSIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>

The active mode is used for teletypes and is considered the standard setting.

5.3.2 4.1.6.2 Papertape Reader Enable (see Table 5-12)

In the active (normally used) mode this output controls and supplies current to a terminal papertape reader. In the passive mode it only controls the reader. Both modes are controlled by the reader enable bit in the RCSR.

<table>
<thead>
<tr>
<th>PACK</th>
<th>SWITCH</th>
<th>ACTIVE</th>
<th>PASSIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>
The Active mode is used for teletypes and is considered the standard setting.

5.3.2 4.1.6.3 Receiver (see Table 5-13)

In the Active mode the receiver will supply 20mA of current to an external device. The external device will modulate the current while the receiver detects the variations.

In the Passive mode the receiver detects variations of an externally supplied, modulated current.

TABLE 5-13 RECEIVER MODE SWITCHES

<table>
<thead>
<tr>
<th>MODE</th>
<th>PACK#</th>
<th>SWITCH#</th>
<th>ACTIVE</th>
<th>PASSIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>3</td>
<td>6</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Passive</td>
<td>3</td>
<td>7</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>Active</td>
<td>3</td>
<td>8</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Passive</td>
<td>3</td>
<td>9</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>Active</td>
<td>3</td>
<td>10</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>

5.3.2 4.1.7 Break Bit Enable (see Table 5-14)

The Break Bit is set by software. Setting the Break Bit causes a continuous space. See the Programmer's Guide, Sect. 2 - for a complete description of the Break Function.

TABLE 5-14 BREAK BIT SWITCH

<table>
<thead>
<tr>
<th>BREAK BIT Enable</th>
<th>P4=51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>ON</td>
</tr>
<tr>
<td>Disable</td>
<td>OFF</td>
</tr>
</tbody>
</table>

The Break Bit is disabled in standard configuration systems.

5.3.2 4.1.8 Error Bit Enable (see Table 5-15)

See the Programmer's Guide for a description of this bit.

TABLE 5-15 ERROR BIT ENABLE SWITCH
The error bit is disabled in standard configuration systems.

5.3.2 4.1.9 Real Time Clock Enable (see Table 5-16)

The Real Time Clock has a single, unchangeable address of 777546.

The Real Time Clock Enable will not function unless the address of the DL11W device registers is 77756Y, the standard system address.

This feature prevents a user from having more than one standard address line clock per system.

Only one line frequency clock with the 777546 address is allowed per system.

TABLE 5-16 REAL TIME CLOCK ENABLE

<table>
<thead>
<tr>
<th>REAL TIME CLOCK ENABLE</th>
<th>P5-S9</th>
<th>P5-S10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>DISABLE</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>

5.3.2 4.2 Jumpers

The DL11W contains one jumper (Fig. 5-21) which is used for test purposes only. This jumper must always be present.

5.3.2 5. MS11JP = (M7847-YJ) 16K MOS MEMORY (with parity)

5.3.2 5.1 Switches

The M7847 contains one pack (P1) of eight switches (S1-S8). The switches perform two distinct functions:

Memory starting address selection I/O page protection.

5.3.2 5.1.1 Memory Starting Address Selection

Switches P1=S1 through P1=S5 select the starting address of the memory module.
5.3.2 5.1.1.1 Range of Starting Address Selection

Any 16K memory can start at any address in 20,000(8) increments from 000 000(8) to 740,000(8). (see Table 5-17).

5.3.2 5.1.1.2 Capacity of 16K Memory

Each 16K memory can contain a maximum of 16,384(10) words which equals 100,000(8) bytes.

5.3.2 5.1.1.3 Standard Configuration

The Standard Configuration would locate the first memory module's starting address at 000 000. Thus all addresses from 000 000 to 77,777(8) are utilized. The switch setting for the standard configuration memory is shown in Table 5-18.
<table>
<thead>
<tr>
<th>MEMORY #1</th>
<th>STARTING ADDRESS</th>
<th>SWITCH</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>40,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>106,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>120,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>140,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>160,000</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEMORY #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>200,000</td>
</tr>
<tr>
<td>220,000</td>
</tr>
<tr>
<td>240,000</td>
</tr>
<tr>
<td>260,000</td>
</tr>
<tr>
<td>300,000</td>
</tr>
<tr>
<td>320,000</td>
</tr>
<tr>
<td>340,000</td>
</tr>
<tr>
<td>360,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEMORY #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>400,000</td>
</tr>
<tr>
<td>420,000</td>
</tr>
<tr>
<td>440,000</td>
</tr>
<tr>
<td>460,000</td>
</tr>
<tr>
<td>500,000</td>
</tr>
<tr>
<td>520,000</td>
</tr>
<tr>
<td>540,000</td>
</tr>
<tr>
<td>560,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEMORY #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>600,000</td>
</tr>
<tr>
<td>620,000</td>
</tr>
<tr>
<td>640,000</td>
</tr>
<tr>
<td>660,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEMORY #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>700,000</td>
</tr>
<tr>
<td>720,000</td>
</tr>
<tr>
<td>740,000</td>
</tr>
</tbody>
</table>

KEY

0 = Switch ON
1 = Switch OFF
M7847-YJ Standard Configuration Switch Setting Of First Memory

**TABLE 5-18 SWITCH SETTING FOR STANDARD CONFIGURATION MEMORY**

<table>
<thead>
<tr>
<th>STARTING ADDRESS</th>
<th>PACK</th>
<th>SWITCH</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 000</td>
<td>1</td>
<td>1</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td>ON</td>
</tr>
</tbody>
</table>

**5.3.2 5.1.1.4 Expanded Configuration**

If the standard configuration has additional 16k memory increments added on, the first additional memory should have its starting address set to 100 0000(8), as shown in the MOS MEMORY SWITCH SETTING CHART (Table 5-18).

**5.3.2 5.1.1.5 Non Standard Starting Addresses**

Non Standard Starting Addresses can be set in accordance with the MOS Memory switch Setting Chart (Table 5-18) as desired.

**5.3.2 5.1.2 I/O Page Protection Switches**

These switches protect the I/O page from a memory overlap. In some machine these switches are variable. These switches should not be varied in PDP11/34 computer systems.* The three I/O page protection switches are:

- P1=56
- P1=57
- P1=58

**5.3.2 5.1.2.1 Switch Settings**

The I/O Page Protection switches (see Table 5-19) will normally be set as follows in all PDP11/34 computer systems which use an M9301 YA or YB terminator/Rom:

* Unless the memory occupies the address space starting at 7000 0000.
TABLE 5-19 M7847 I/O PAGE PROTECT SWITCH SETTINGS

<table>
<thead>
<tr>
<th>SWITCH</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>OFF</td>
</tr>
<tr>
<td>7</td>
<td>ON</td>
</tr>
<tr>
<td>8</td>
<td>OFF</td>
</tr>
</tbody>
</table>

If a memory starting address is 700,000, the I/O Page Protection switches are set as follows (see Table 5-20):

TABLE 5-20 TOP OF MEMORY I/O PAGE PROTECT SWITCH SETTING

<table>
<thead>
<tr>
<th>SWITCH</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>OFF</td>
</tr>
<tr>
<td>7</td>
<td>OFF</td>
</tr>
<tr>
<td>8</td>
<td>OFF</td>
</tr>
</tbody>
</table>

It should be noted that this feature makes the top 4K of memory inaccessible.

5.3.2 5.2 Jumpers

The M7847-YB memory uses three fixed jumpers, which must always be present.

5.3.2 5.2.1 Jumper Designations

Jumpers on this memory are referred to jumper posts and are documented slightly differently than other jumper arrangements.

Each post is assigned a number. Six posts (W1, W2, W3, W4, W5, W6) are on the board.

Jumpers are assigned the numbers of the posts that they bridge, thus W1-W2 is a jumper which goes from W1 to W2.

5.3.2 5.2.2 Jumper Placement

The 16K MOS memory must have the following jumpers present:

W1-W2
W3-W4
W5-W6

5.3.2 6. MM11DP G652 = H22A CORE MEMORY (with Parity)
5.3.2 6.1 Switches

The MM11DP does not utilize any switches.

5.3.2 6.2 Jumpers

Four sets of jumpers are used on the G652 Memory module:

Starting Address Selection
Interleave Control
I/O Page Protection
Strobe Range Control

5.3.2 6.2.1 Starting Address Selection

Four jumpers W1, W2, W3, W4 can vary the range of the starting address from 000 000(10) to 120K(10) in 8K(10) increments.

Standard Address

The first memory module in standard PDP11/34 machines is configured to start at address zero (see Table 5-21).

<table>
<thead>
<tr>
<th>TABLE 5-21 MM11DP ADDRESS ZERO SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMPER</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Expanded Addresses - Standard Configuration

If additional 16K increments of memory are added to a standard system, the starting address would be 16K, 32K, etc., as shown in Table 5-22.
<table>
<thead>
<tr>
<th>S.A.</th>
<th>E.A. +1</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16K</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>16K</td>
<td>32K</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>32K</td>
<td>40K</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>40K</td>
<td>48K</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td>64K</td>
<td>56K</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>80K</td>
<td>64K</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>96K</td>
<td>80K</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>128K</td>
<td>96K</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>192K</td>
<td>128K</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>128K</td>
<td>128K</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
</tbody>
</table>

**KEY**

S.A. = Starting Address  
E.A. +1 = End Address of Memory + 1  
W1 = Jumper W1  
W2 = Jumper W2  
W3 = Jumper W3  
W4 = Jumper W4  
IN means the jumper is present  
OUT means the jumper is absent
5.3.2 6.2.2 Interleaving Control Jumpers

5.3.2 6.2.2.1 Definition of Interleaving

If more than one core memory is used in system, memory performance can be improved by alternating the appropriations of sequential words, via a process called interleaving which is illustrated in the following example:

**INTERLEAVED OPERATION**

<table>
<thead>
<tr>
<th>Memory 1</th>
<th>Memory 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Word</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3.2 6.2.2.2 Interleave Jumpers

Jumpers W5 and W6 control the interleaving process.

Non-interleaved Operation

W5 and W6 are not present

Interleaved Operation

One board should have W5 absent W6 present. The other board should have W5 present W6 absent.

Standard Configuration

Unless specifically requested by the customer, memories will not be interleaved by the factory.

5.3.2 6.2.3 I/O Page Protection Jumpers

The I/O page is protected by jumpers W7 and W8. These jumpers should be absent in all standard configuration PDP11/34 machines.

5.3.2 6.2.4 Strobe Range Control Jumpers
W9, W10, and W11 are factory settable jumpers only. These jumpers shift the leading edge of the strobe pulse. This feature compensates for component tolerance in various circuits. Two boards which are otherwise identical can have different strobe control jumpers. These jumpers will be documented on the jumper chart of the system as described in Paragraph 5.2.

5.3.2 7. M9302 BUS END TERMINATOR

The M9302 does not contain any switches or jumpers.

5.3.2 8. M7850 PARITY CONTROLLER

5.3.2 8.1 Switches

The M7850 does not contain any switches.

5.3.2 8.2 Jumpers

The M7850 contains four jumpers (W1, W2, W3, and W4) which allow the user to change the address of the control and status register as shown in Table 5-23.

5.3.2 8.2.1 CSR Address

The M7850 CSR address range is 772100 to 772136, which spans 16 locations.

5.3.2 8.2.2 Standard Settings

The standard setting for the first parity controller in a system is 772100, the jumpers are placed as shown in Table 5-24.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>W4</th>
<th>W3</th>
<th>W2</th>
<th>W1</th>
</tr>
</thead>
<tbody>
<tr>
<td>772100</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
</tbody>
</table>

5.3.2 8.2.3 Expanded Systems
One M7850 parity controller is required for each backplane, regardless of the number of slots. This one controller will control either MOS or Core or a mixture of both types memories (any number of units) in that backplane.

The second parity controller is usually number 772102. The third parity controller is 772104, etc.
<table>
<thead>
<tr>
<th>MACHINE ADDRESS (WORDS)</th>
<th>W4</th>
<th>W3</th>
<th>W2</th>
<th>W1</th>
</tr>
</thead>
<tbody>
<tr>
<td>772100</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td>772102</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>772104</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>772106</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>772110</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td>772112</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>772114</td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>772116</td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>772120</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td>772122</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>772122</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>772124</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>772126</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>772130</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td>772132</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>772134</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td>772136</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
</tr>
</tbody>
</table>
5.3.3 Backplanes

5.3.3.1 Introduction

Backplanes fall into three categories in the PDP 11/34 Computer systems:

- Processor
- General Purpose Expander
- Dedicated Purpose (Special Purpose)

5.3.3.2 The Processor Backplane = DD11-PK

The DD11-PK Backplane is used as the Basic PDP 11/34 Processor Backplane. It is the only backplane used in the BA11K Mounting Box and can contain a wide variety of modules (K611E Processor, Terminators, SPC Compatible Modules and Modified Unibus Compatible Modules). The BA11K Mounting Box used the DD11PK Backplane to operate the processor and associated modules; however, the BA11K can accept optional General Purpose Expander Backplanes and/or Special Purpose Backplanes.

No other Processor Backplanes exist for PDP 11/34 systems.

5.3.3.2.1 Anatomy Of A DD11-PK Backplane

Each DD11-PK Backplane contains four distinct types of connector groups. (See Figure 5-14).

- Processor
- Modified Unibus
- Standard Unibus
- SPC

5.3.3.2.1.1 Processor Slots

Slot 1 is wired to accept the M7266 Control Module. Table 5-25 lists Slot 1 Backplane Pins with their respective name.

Slot 2 is wired to accept the M7265 Data Path Module (pin numbers and names are listed in Table 5-26).

Slots 1 and 2 comprise the Processor Interconnections.

5.3.3.2.1.2 Standard Unibus Connectors
Connectors A & B of Slot 9 contains the only accessible Standard Unibus Connectors, on the D011-PK Backplane.

Table 5-27 illustrates the pin designation for Standard and Modified Unibus Pins.

5.3.3 2.1.3 Modified Unibus Connectors

Connectors A & B in slots 3-8 comprise 6, Modified Unibus Connectors. Note that all Modified Unibus Connectors are wired in parallel and all are identical.

5.3.3 2.1.4 Modified vs. Standard Unibus

The Modified Unibus Connectors have taken Standard Unibus Connectors, removed extra Grounds, Bus Grant Signals, and the M0G signal. These pins have been redesignated with Core Memory Voltage Pins, Battery Back-Up Voltage pins for the M0G Memory, parity Signal Pins, An INT SSYN Signal Pin, in addition to several reserved pins and test point pins.

5.3.3 2.1.5 SPC Connectors

Each slot (3-9) contains 4 connectors (C, D, E, and F) which constitutes an SPC connector. The D011 PK Backplane contains 7 SPC connector. Table 5-28 lists the pins of one SPC connector, all connectors are identical.

5.3.3 2.1.5.1 Bus Grants

Within each SPC connector are 4 levels of Bus Grants signals (BG 4 - BG 7). These signals originate within the arbitrator section of the processor, and notify a module of its impending control of the Bus. Modules which are at the same priority level will have their priority determined by their position relative to the processor. The nearer a module is to the processor, the higher its priority will be.

Each of the four grant signals is on a separate signal line that is routed through each SPC module in connector D. A Bus Grant jumper card (G727) must be inserted into each empty SPC slot in connector D in order to maintain Bus Grant continuity.
TABLE 5-25
DD11PK SLOT 1 PIN DESIGNATIONS
To Be Supplied

TABLE 5-26
DD11PK SLOT 2 PIN DESIGNATIONS
To Be Supplied
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>INIT</td>
<td>+SV</td>
<td>RESV +SV</td>
<td>BUS</td>
</tr>
<tr>
<td>INTR</td>
<td>TP</td>
<td>RESV TP</td>
<td>PIN</td>
</tr>
<tr>
<td>D00</td>
<td>GND</td>
<td>BR5</td>
<td>GND</td>
</tr>
<tr>
<td>D01</td>
<td>L</td>
<td>GND</td>
<td>BR4</td>
</tr>
<tr>
<td>D02</td>
<td>L</td>
<td>D03</td>
<td>L</td>
</tr>
<tr>
<td>D04</td>
<td>D05</td>
<td>AC</td>
<td>DC</td>
</tr>
<tr>
<td>D06</td>
<td>D07</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>D08</td>
<td>L</td>
<td>A01</td>
<td>A00</td>
</tr>
<tr>
<td>D10</td>
<td>D09</td>
<td>A03</td>
<td>A02</td>
</tr>
<tr>
<td>D12</td>
<td>D11</td>
<td>A05</td>
<td>A04</td>
</tr>
<tr>
<td>D14</td>
<td>D13</td>
<td>A07</td>
<td>A06</td>
</tr>
<tr>
<td>PA</td>
<td>D15</td>
<td>A09</td>
<td>A08</td>
</tr>
<tr>
<td>PB</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>GND</td>
<td>PB</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>GND</td>
<td>BDST</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>GND</td>
<td>SACK</td>
<td>A15</td>
<td>A14</td>
</tr>
<tr>
<td>NPR</td>
<td>L</td>
<td>A17</td>
<td>A16</td>
</tr>
<tr>
<td>BR7</td>
<td>GND</td>
<td>C1</td>
<td>L</td>
</tr>
<tr>
<td>B6</td>
<td>L</td>
<td>C0</td>
<td>L</td>
</tr>
<tr>
<td>GND</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>MSYN</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>BR6</td>
<td>SSYN</td>
<td>C0</td>
<td>L</td>
</tr>
<tr>
<td>+20 (CORE)</td>
<td>+20</td>
<td>MSYN</td>
<td>C0</td>
</tr>
<tr>
<td>+20 (CORE)</td>
<td>+20</td>
<td>MSYN</td>
<td>-5</td>
</tr>
</tbody>
</table>

**STANDARD UNIBUS**

**AFFlicted PINS**

**MODIFIED UNIBUS**

**KEY**

- A diagonal line in the lower right corner indicates an affected pin.

*Table 5:27 Standard and Modified Unibus Pin Designations*
<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NPG (IN)</td>
<td>+5V</td>
<td>TP</td>
<td>+5V</td>
<td>GND</td>
</tr>
<tr>
<td>NPG (OUT)</td>
<td>-15V</td>
<td>TP</td>
<td>-15V</td>
<td>ASSYN IN H</td>
</tr>
<tr>
<td>PA</td>
<td>GND</td>
<td>SEL</td>
<td>GND</td>
<td>A12</td>
</tr>
<tr>
<td>LTC</td>
<td>D15 L</td>
<td>AOUT LOW</td>
<td>BR7 L</td>
<td>A17</td>
</tr>
<tr>
<td>TP</td>
<td>D14 L</td>
<td>SEL 4</td>
<td>MSYN L</td>
<td>A16</td>
</tr>
<tr>
<td>TP</td>
<td>D13 L</td>
<td>SEL D</td>
<td>BS5 L</td>
<td>A02</td>
</tr>
<tr>
<td>D11</td>
<td>D12 L</td>
<td>AIN 3</td>
<td>BR4 L</td>
<td>A01</td>
</tr>
<tr>
<td>PINT B</td>
<td>D10 L</td>
<td>ASEL 2</td>
<td>ABR OUT</td>
<td>SSYN L</td>
</tr>
<tr>
<td>TP</td>
<td>D68 L</td>
<td>AOUT</td>
<td>G7 SO</td>
<td>A14</td>
</tr>
<tr>
<td>PINT EMBB</td>
<td>D67 L</td>
<td>INIT</td>
<td>A11</td>
<td>TP</td>
</tr>
<tr>
<td>TP</td>
<td>D66 L</td>
<td>INIT</td>
<td>G7 OUT</td>
<td>A13</td>
</tr>
<tr>
<td>DC</td>
<td>D65 L</td>
<td>AIN &amp; A</td>
<td>B6L OUT</td>
<td>AOUT</td>
</tr>
<tr>
<td>HALT REQ</td>
<td>D64 L</td>
<td>TP</td>
<td>G5 SO</td>
<td>A10</td>
</tr>
<tr>
<td>HALT GRT</td>
<td>D63 L</td>
<td>TP</td>
<td>G5 OUT</td>
<td>A09</td>
</tr>
<tr>
<td>PB</td>
<td>D62 L</td>
<td>SEL 3</td>
<td>BC4 SO</td>
<td>ASEL 4</td>
</tr>
<tr>
<td>GND</td>
<td>D61 L</td>
<td>GND</td>
<td>BC4 OUT</td>
<td>GND</td>
</tr>
<tr>
<td>+15/8</td>
<td>D60 L</td>
<td>TP</td>
<td>A06</td>
<td>A04</td>
</tr>
<tr>
<td>AC</td>
<td>D59 L</td>
<td>ASSYN IN H</td>
<td>ABG</td>
<td>A03</td>
</tr>
</tbody>
</table>

Table 5-28
SPC Pin Designations
A similar scheme to that used for Bus Grants is used for the Single NPQ Line. A jumper card is not required since wire jumpers are installed in all C connectors, between pins A1 & B1 (pins designated CA1 and CB1) wherever a NPR Module does not occupy that slot. The jumper wire must be removed to add a module utilizing NPR's.

5.3.3 3. Expanding A System Within A DD11P Backplane

The prime considerations which must be taken into account are:

1. Will the Option work in the DD11P Backplane?*

   If the answer is no, another backplane will have to be used. If the system is a BA11L version, this won't be possible to use this option directly, an external Expander Box is required. If the system is a BA11K, refer to paragraph 5.3.3 5 of this manual.

2. Is an open slot available to operate this new option?

   When a system is filled, the user could:
   a. Consider buying an Expander Box.
   b. Consider removing an existing module, and replace it with the new module.

3. Will the power supply output capabilities be exceeded?

   Refer to paragraph 5.3.4 for a discussion of power supply loading considerations.

4. The Bus Grant Card of the empty slot must be removed (unless the slot is where a Core Memory Stack will hang over).

5. If the device that is being installed is an NPR type device (Non Processor Request), the wire wrap jumper wire that goes from CA1 to CB1, of the slot that is going to be used, must be removed.

6. If the system is a BA11L with MOS Memory, and additional MOS Memory is added, will the Battery Back-up time be sufficient? (See Figure 5-33)

* The DD11P Backplane will operate Modified Unibus Devices and SPC Devices.
5.3.3 4. General Purpose Expander Backplanes

5.3.3 4.1 Definition

A General purpose Expander Backplane is a backplane which is compatible with both Modified Unibus Devices (Modules) and SPC Pinned Modules.

5.3.3 4.2 Overview

Two General purpose Expander backplanes have been designed for the PDP 11/34 Computer systems:

- DD11-DK 9 Slot Backplane (2 SU)
- DD11-CK 4 Slot Backplane (1 SU)

5.3.3 4.3 DD11-DK

The DD11-DK is quite similar to the DD11-PK except that Slots 1 and 2 are not intended for Processor use. Figure 5-34 illustrates the interconnection provided on the DD11-PK.

Note that the DD11-PK contains the following SRO Sets:

- Standard Unibus
- Modified Unibus
- SPC

A Unibus Cable or jumper is placed in the Standard Unibus connector in slot 1, (AB). A quad height SPC can also be placed in slot 1 (C, D, E, F). Slots 2-9 in this backplane are identical to slots 3-9 in the DD11-PK Backplane. All unused SPC slots (slots 1-9) must contain BG cards. Slot 9, connects AB must contain a Unibus Cable (if other units are jumpered to) or an M9302 Bus End-Terminator if this is the last backplane of a system.

5.3.3 4.4 DD11-CK

The DD11-CK backplane is a 4 slot version of the DD11-DK, which contains the same three types of interconnections (See DD11-DK).

Figure 5-35 illustrates the Interconnections of the DD11-CK. Configuration rules previously mentioned for the DD11-DK apply to this backplane.

* G727 unless a module occupies the SPC slot
### FIGURE 5-34 DD11-DK BACKPLANE

<table>
<thead>
<tr>
<th>Slot</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STANDARD</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>STANDARD</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>STANDARD</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### FIGURE 5-35 DD11-CK BACKPLANE

<table>
<thead>
<tr>
<th>Slot</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STANDARD</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MODIFIED</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STANDARD</td>
<td></td>
<td>SPC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3.3 5. Dedicated Backplane

5.3.3 5.1 Definition
Dedicated backplanes are special purpose backplane which are designed to support only one option.

5.3.3 5.2 Typical Options which Require a Dedicated Backplane

<table>
<thead>
<tr>
<th>RK11-D</th>
<th>Disk Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD11-E</td>
<td>1200 CPM 80 Column Card REA Controller</td>
</tr>
</tbody>
</table>

5.3.3 5.3 Dedicated Backplane Documentation
When an option which requires its own backplane is purchased, DEC usually supplies (or has available) all documentation which is required to operate that option, including backplane pin designations.

5.3.3 6. Expanding The Basic BA11K PDP11/34 System

5.3.3 6.1 Introduction
BA11Ks are expanded beyond their DD11-PK backplanes capacity in either of two ways:

- The DEC sales representative and the user determine an expanded configuration system is desirable at the time of purchase.
- The user decides to expand the system at a later time.

5.3.3 6.2 Expanded Systems - Factory Shipped
When the factory ships an expanded system, that system will be documented as described in paragraph 5.2 by the Chart System, as well as by its accompanying documentation which is optionally available.

5.3.6.3 User Expanded System
When a user purchases an option kit which includes a backplane, or purchases an expander backplane, directions are usually included about
installation, and documentation is usually available, for the entire package.

5.3.3 6.4 Interconnecting Backplanes

Backplanes are connected to each other by jumper cables which plug into the Standard Unibus Connectors, (Figure 5-36).

Slot 9 of the DD11-P (A and B) are connected to slot 1 (A & B) of the next adjacent backplane via a cable. The last slot (A & B) of this unit are plugged into the first slot of the next backplane until the last backplane is reached. The last slot of the last backplane (A & B) should contain the M9302 Terminator.

5.3.4 Connectors

5.3.4 1. Introduction

This section discusses the connectors used in the standard version PDP 11/34 computers.

5.3.4 2. Power Connectors

Figure 5-37 illustrates the DD11-PK Backplane - Cable - Connector assembly. Note there are two large connectors and one small connector. The large connector is a 15 Pin Mate-n-lok connector.

The smaller connector is a 6 pin Mate-n-lok connector.

Table 5-29 lists the pin designations for these connectors.

DD11-DK

Identical power connectors are used on the DD11-DK to the DD11-PK. However, no cable to the operators' console is present.

DD11-CK

The DD11C has identical pinning on one 15 pin connector and one 6 pin connector to the DD11-PK. There is no second 15 pin connector on the DD11-CK or cable to the operator's console.
FIGURE 5-36  INTERCONNECTING BACKPLANES WITH CABLES
## Table 5-29 Power Connector Pin Listings

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>WIRE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>#14 Red</td>
</tr>
<tr>
<td>2</td>
<td>+15V</td>
<td>#14 Gray</td>
</tr>
<tr>
<td>3</td>
<td>+20V</td>
<td>#14 Orange</td>
</tr>
<tr>
<td>4</td>
<td>+5V</td>
<td>#14 Red</td>
</tr>
<tr>
<td>5</td>
<td>GROUND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>6</td>
<td>+15 BAT</td>
<td>#14 Gray</td>
</tr>
<tr>
<td>7</td>
<td>GROUND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>8</td>
<td>GROUND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>9</td>
<td>GROUND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>10</td>
<td>GROUND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>11</td>
<td>+5 BAT</td>
<td>#14 Red</td>
</tr>
<tr>
<td>12</td>
<td>-15V</td>
<td>#14 Blue</td>
</tr>
<tr>
<td>13</td>
<td>-5V</td>
<td>#14 Brown</td>
</tr>
<tr>
<td>14</td>
<td>-15 BAT</td>
<td>#14 Blue</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**2 - 15 Pin Mate "N" Lok**

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>WIRE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LO GND</td>
<td>#26 Black</td>
</tr>
<tr>
<td>2</td>
<td>LTC (Line Clock)</td>
<td>#26 Brown</td>
</tr>
<tr>
<td>3</td>
<td>DC LO</td>
<td>#26 Violet</td>
</tr>
<tr>
<td>4</td>
<td>AC LO</td>
<td>#26 Yellow</td>
</tr>
<tr>
<td>5</td>
<td>(not connected)</td>
<td>Not Used</td>
</tr>
<tr>
<td>6</td>
<td>(not connected)</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

**1 - 6 Pin Mate "N" Lok**

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>WIRE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pin</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DC LO</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AC LO</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HALT REQUEST</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HALT GRANT</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SACK</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>INIT</td>
<td></td>
</tr>
</tbody>
</table>
5.3.4 2.1 Placement of Power Connectors

Receptacles for the backplane cables are located on the DC distribution boards of the BA11K and BA11L power supplies. All 15 pin connectors are parallel wires as are all 6 pins connectors. Placement is not critical.

5.3.4 3. Operators' Console Connector

A ten pin 3M connector plugs into the operators' console from the DD11-PK Backplane. It is shown in Table 5-3A.

5.3.4 4. Remote Power Control Connector

The power distribution boards in both the BA11L and BA11K have a 3 pin (in line) Mate-n-lok connector which connects to remote units. This function allows the power switch on the operators' console to switch power to all units simultaneously.

5.3.5 Power Supply Loading Considerations

5.3.5 1. Introduction

Power supplies should be operated within their rated limits. Exceeding these limits can cause a possible degradation of system performance and reliability.

5.3.5 2. BA11L Power Considerations

5.3.5 2.1 BA11L - H777 Power Supply Capabilities

Table 5-31 indicates how much current can be obtained at any output without damage to the supply.

**CAUTION**

These currents cannot all be used to their maximum capacity simultaneously.

If all outputs used their maximum current, the power supply would have to provide 613 watts of power.
The 4777 Power Supply is rated at 480 watts of input power.
### Table 5-31
BA11 = H777 Power Supply Current Abilities Capabilities

<table>
<thead>
<tr>
<th>Regulator</th>
<th>Output Volt</th>
<th>+5</th>
<th>+5B</th>
<th>+15</th>
<th>-15</th>
<th>+20</th>
<th>-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td></td>
<td>25A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td>3 1/2A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The sum of the current (disregarding sign) of +15V and -15V can be up to 2A.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total combined power of all outputs is the product of voltage and current, thus the total combined output.
5.3.5 2.2 Calculating System Power Requirements

Table 5-32 lists the current and power dissipated by various options. To determine the power requirements:

1. List all modules in the proposed system one per line, starting on line 1 of the power consideration work sheet (Table 5-33).

2. Refer to Table 5-32, find the module listed on line 1 and transpose all of the current requirements and the power dissipated values.

3. Repeat Step 2 for each module in the system.

4. After the current requirements for all modules have been transposed, add the requirements in each column (columns A-H) place the totals on line 15.

5. Compare lines 15 and 18 for all column except D and E. Line 18 should be greater than line 15 in all cases including column H (power dissipated). If line 15 was greater in any case, the configuration is not acceptable.

6. Add the magnitude of the +15V currents and the -15 current, as provided for in column D (+15V) lines 15, 16, and 17.

7. Compare line 17 and 18 (column D). If line 17D is greater than line 18D, the configuration is unacceptable.

8. If none of the allowable limits was exceeded, the configuration, from a power standpoint, is acceptable.

5.3.5 2.3 Battery Back-Up Loading

When power fails, and the system is in the Power On or Standby Modes, the Battery Back-Up System takes over. The Battery Back-Up only furnishes power to a portion of the MOS Memory, the refresh circuits. This reduces 5V power consumption to the 5VB line only. The +15V and -15V lines now only supply power to the MOS Memory (via +15VB and -15VB) which demands half the +15V power it would during normal operation.

Figure 5-40 illustrates the relationship of MOS Memory Loading to the duration of the Battery Back-Up.
## PDP-11 FAMILY MODELS AND OPTIONS POWER REQUIREMENTS

<table>
<thead>
<tr>
<th>Model/Option</th>
<th>Description</th>
<th>5V (CPU)</th>
<th>+5V (Options)</th>
<th>+15V</th>
<th>+20V</th>
<th>-5V</th>
<th>+15V</th>
<th>Power Dissipated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H744</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H745</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H754</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S409730+YA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st 5=8</td>
<td>KD11-B</td>
<td>0.0A</td>
<td>0.25A</td>
<td>4.4A</td>
<td>0.5A</td>
<td>0.05A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM11-U</td>
<td>6.1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 M920s</td>
<td>Total Amperes</td>
<td>2.5A</td>
<td>0.25A</td>
<td>4.4A</td>
<td>0.51A</td>
<td>0.05A</td>
<td>25W</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM11-U</td>
<td>16.6A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M920</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Amperes</td>
<td></td>
<td>21A</td>
<td>7.3A</td>
<td>4.4A</td>
<td>0.51A</td>
<td></td>
<td>45W</td>
<td></td>
</tr>
<tr>
<td>Model/Option</td>
<td>Description</td>
<td>5V (CPU)</td>
<td>+5V (Options)</td>
<td>-15V</td>
<td>+28V</td>
<td>+5V</td>
<td>+15V</td>
<td>Power Dissipated</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------</td>
<td>----------</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>*MF11-U/MM11-U</td>
<td>16K Sense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Active)</td>
<td>Core Memory</td>
<td>6.1A</td>
<td>4.4A</td>
<td>8.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Standby)</td>
<td>(Double SU)</td>
<td>4.5A</td>
<td>4.4A</td>
<td>8.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF11-UP/MM11-UP</td>
<td>16K Sense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Active)</td>
<td>Core with Parity</td>
<td>7.3A</td>
<td>4.4A</td>
<td>8.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Standby)</td>
<td>(Double SU)</td>
<td>5.7A</td>
<td>4.4A</td>
<td>8.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF1-L (MM11=L)</td>
<td>8K Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Active)</td>
<td>Memory</td>
<td>3.4A</td>
<td>6.3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Standby)</td>
<td>(Double SU)</td>
<td>1.7A</td>
<td>6.3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF1-LP (MM11=LP)</td>
<td>8K Parity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Active)</td>
<td>Core Memory</td>
<td>4.9A</td>
<td>6.3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Standby)</td>
<td>(Double SU)</td>
<td>1.7A</td>
<td>6.3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MM11=S</td>
<td>Same as MM11=L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Except in SU Configuration (1 SU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Non-Interleaved.
<table>
<thead>
<tr>
<th>Option</th>
<th>Mounting Code</th>
<th>Description</th>
<th>Power Harness</th>
<th>+5V</th>
<th>+15V</th>
<th>+20V</th>
<th>+5V</th>
<th>+15V</th>
<th>Power Dissipated</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA11-D</td>
<td>1 SU</td>
<td>Digital to Analog Converter Subsystem</td>
<td>7009562</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3W</td>
</tr>
<tr>
<td>BA614</td>
<td>(AA11-D)</td>
<td>D/A Converter</td>
<td>310A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30W</td>
</tr>
<tr>
<td>BM792-Y</td>
<td>SPC</td>
<td>Bootstrap Loader</td>
<td>0.3A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3A</td>
</tr>
<tr>
<td>CD11-A/B</td>
<td>1 SU</td>
<td>1000 CPM, 80 Col. Card Reader Controller</td>
<td>7010117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28A</td>
</tr>
<tr>
<td>D11-E</td>
<td>1 SU</td>
<td>1200 CPM, 80 Col. Card Reader Controller</td>
<td>7010117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28A</td>
</tr>
<tr>
<td>CM11</td>
<td>SPC</td>
<td>200 CPM, 80 Col. Card Reader Controller</td>
<td>1.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15A</td>
</tr>
<tr>
<td>CR11</td>
<td>SPC</td>
<td>300 CPM, 80 Col. Card Reader Controller</td>
<td>1.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15A</td>
</tr>
<tr>
<td>1=DB</td>
<td>1 SU</td>
<td>Unibus Link</td>
<td>4.0A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40W</td>
</tr>
<tr>
<td>DA11-F</td>
<td>1 SU</td>
<td>Unibus Window</td>
<td>7010117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5W</td>
</tr>
<tr>
<td>DB11-A</td>
<td>1 SU</td>
<td>Bus Repeater</td>
<td>7009562</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32W</td>
</tr>
<tr>
<td>Option</td>
<td>Mounting Code</td>
<td>Description</td>
<td>Power Harness</td>
<td>+5V</td>
<td>-15V</td>
<td>+20V</td>
<td>-5V</td>
<td>+15V</td>
<td>Power Dissipated</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>------------------------------</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>DD11-B</td>
<td>1 SU</td>
<td>Peripheral Mounting Panel</td>
<td>7010117</td>
<td>8.4A</td>
<td>0.24</td>
<td></td>
<td></td>
<td></td>
<td>93W</td>
</tr>
<tr>
<td>DM11-AA,AB,AC</td>
<td>2 SU</td>
<td>16-Line Prog Async Max</td>
<td>7010118</td>
<td>10.8A</td>
<td>0.645</td>
<td>0.4</td>
<td>0.25</td>
<td></td>
<td>57W</td>
</tr>
<tr>
<td>DM11-AD,AB</td>
<td>2 SU</td>
<td>16-Line Prog Async Max (EIA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DJ11-AA</td>
<td>1 SU</td>
<td>16-Line Async Max (EIA)</td>
<td>7010117</td>
<td>4.7A</td>
<td>0.25</td>
<td></td>
<td></td>
<td>0.25</td>
<td>57W</td>
</tr>
<tr>
<td>DJ11-AB</td>
<td>1 SU</td>
<td>16-Line Async Max (TTL)</td>
<td></td>
<td>4.7A</td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DJ11-AC</td>
<td>1 SU</td>
<td>16-Line Async Max (20 mA)</td>
<td></td>
<td>5.3A</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL11</td>
<td>SPC</td>
<td>Async Interface</td>
<td></td>
<td>1.8A</td>
<td>0.15</td>
<td></td>
<td>0.05</td>
<td>21W</td>
<td></td>
</tr>
<tr>
<td>DM11-BB</td>
<td>(DM11)</td>
<td>16-Line Modem Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM11-DA, DB, DC (DM11)</td>
<td></td>
<td>Line Adapters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM11-AD</td>
<td>Hex SPC</td>
<td>DDCMP Network Link</td>
<td></td>
<td>2.5A</td>
<td>0.31</td>
<td></td>
<td></td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>DM11-DA</td>
<td>Hex SPC*</td>
<td>Network Link EIA Line Unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM11-NA, ND</td>
<td>Hex SPC*</td>
<td>Network Link Local Line Unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM11-AA</td>
<td>1 SU</td>
<td>Auto Calling System Unit</td>
<td>7009562</td>
<td>1.0A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM11-DA</td>
<td>(DM11-AA)</td>
<td>Auto Calling Unit</td>
<td></td>
<td></td>
<td>0.4A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DQ11-DA, EA</td>
<td>1 SU</td>
<td>Sync Interface</td>
<td>7010117</td>
<td>6.0A</td>
<td>0.07</td>
<td></td>
<td></td>
<td>0.04</td>
<td>60W</td>
</tr>
<tr>
<td>DQ11-AB</td>
<td>1 SU</td>
<td>Error Detection Option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.2A</td>
<td></td>
</tr>
<tr>
<td>DQ11-BB</td>
<td>(DQ11-AB)</td>
<td>Protocol Handling Option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.6A</td>
<td></td>
</tr>
</tbody>
</table>

*with cutout to clear Unibus cables and M930 Unibus jumper.
<table>
<thead>
<tr>
<th>Option</th>
<th>Mounting Code</th>
<th>Description</th>
<th>Power Harness</th>
<th>Current Needed</th>
<th>Power Dissipated</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV11-DA</td>
<td>SPC</td>
<td>Sync Interface</td>
<td>2.0 A</td>
<td>0.15</td>
<td>0.05</td>
</tr>
<tr>
<td>DUP11-DA</td>
<td>Hex, SPC</td>
<td>Sync Interface (DD16MP, S126, BS)</td>
<td>3.6 A</td>
<td>0.6</td>
<td>0.325</td>
</tr>
<tr>
<td>DV11-AA</td>
<td>254</td>
<td>16-Line Sync Mod</td>
<td>15.0 A</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>DV11-BA</td>
<td>(DV11-AA)</td>
<td>8-Line Group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KG11=L</td>
<td>SPC</td>
<td>Comm Arith Unit</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>KW11-L</td>
<td>CPU</td>
<td>Line Clock</td>
<td>0.8 A</td>
<td></td>
<td>8%</td>
</tr>
<tr>
<td>KW11-P</td>
<td>SPC</td>
<td>Proc Line Ck</td>
<td>1.0 A</td>
<td></td>
<td>12%</td>
</tr>
<tr>
<td>LC11-A</td>
<td>SPC</td>
<td>LA30 Control</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LP11-R</td>
<td>SPC</td>
<td>1200 LPM Printer</td>
<td>1.0 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LP11-S</td>
<td>SPC</td>
<td>900 LPM Printer</td>
<td>1.0 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LP11-W</td>
<td>SPC</td>
<td>240 LPM Printer</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LP11-V</td>
<td>SPC</td>
<td>300 LPM Printer</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LS11-A</td>
<td>SPC</td>
<td>60 LPM Printer</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>LV11-B</td>
<td>SPC</td>
<td>Electrostatic Printer</td>
<td>1.5 A</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>Option</td>
<td>Mounting Code</td>
<td>Description</td>
<td>Power Harness</td>
<td>+5V</td>
<td>+15V</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>--------------------------------------</td>
<td>---------------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>MR11-DB</td>
<td>2 SPC</td>
<td>Bootstrap</td>
<td></td>
<td></td>
<td>0.6A</td>
</tr>
<tr>
<td>PC11</td>
<td>SPC</td>
<td>Paper Tape</td>
<td></td>
<td>1.5A</td>
<td></td>
</tr>
<tr>
<td>PR11</td>
<td>SPC</td>
<td>Paper Tape (Reader)</td>
<td></td>
<td>1.5A</td>
<td></td>
</tr>
<tr>
<td>RH11</td>
<td>DBL SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RK11-D</td>
<td>SU</td>
<td>Disk and Ctrl</td>
<td>7610115</td>
<td>7.5A</td>
<td></td>
</tr>
<tr>
<td>TA11-A</td>
<td>SPC</td>
<td>Dual Cassette Interface</td>
<td></td>
<td>1.5A</td>
<td></td>
</tr>
<tr>
<td>M9301</td>
<td>DBL HEIGHT</td>
<td>Bootstrap Terminator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M9302</td>
<td>DBL HEIGHT</td>
<td>Terminator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS110</td>
<td>HEX, QUAD</td>
<td>SOS Memory</td>
<td></td>
<td>1.25A</td>
<td></td>
</tr>
<tr>
<td>DL11W</td>
<td>SPC</td>
<td>Serial Line Int/Real Time Clock</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MM11D</td>
<td>DBL HEX</td>
<td>Core Memory 16K</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MM11B/C</td>
<td>DBL HEX</td>
<td>Core Memory 4K</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>M7850</td>
<td>DBL HEIGHT</td>
<td>Parity Controller</td>
<td></td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>----</td>
<td>---------------------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Line</td>
<td></td>
<td></td>
<td>Module +5</td>
<td>+15</td>
<td>+16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mos Memories only)</td>
<td>+5V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+5V0</td>
<td></td>
<td>+20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 SUM OF +15 &amp; -15 (MAGNITUDE ONLY)</td>
<td>25A</td>
<td>35A</td>
<td>2A</td>
<td>6A</td>
<td>4A</td>
</tr>
</tbody>
</table>

TABLE 5-31
RAIL POWER CONSIDERATION WORK SHEET
5.3.5 3. BA11K Power Consideration

5.3.5 3.1 BA11-K H765 Power Supply Capabilities

Table 5-34 indicates the current capabilities of the H765 Power Supply.

**CAUTION**

These currents cannot all be used to their maximum capacity simultaneously.

If all outputs used their maximum current, the power supply would have to provide 1244 Watts of input power.

The H765 power supply is rated at 1000 Watts of input power.

5.3.5 3.2 Calculating BA11K System Power Requirements

Table 5-34 lists the current and input power dissipated by various options.

To determine the power requirements,

1. List all modules in the proposed system, one per line, starting on line 1 on the BA11K power considerations work sheet. (Table 5-35)

2. Refer to Table 5-32, find the module listed on line 1 and transpose all of the current requirement and the BA11K Input power dissipation column values.

3. Repeat step 2, for each module in the system.

4. Upon completion of step 3, add the values in each column, and place the sum of each column on line 31 (E+H).

5. Compare lines 31 and 32, if line 31 is greater in any column, the configuration is not acceptable.

Note that the BA11K does not support the Battery Back-Up option at this time, and will not be discussed.
### TABLE 5-34
**BA11-K OUTPUT POWER CHARACTERISTICS (H765 POWER SUPPLY)**

<table>
<thead>
<tr>
<th>Regulator</th>
<th>Voltage and Tolerance</th>
<th>Output Current (Max)</th>
<th>Maximum Peak-to-Peak Ripple</th>
</tr>
</thead>
<tbody>
<tr>
<td>H744</td>
<td>+5Vdc + 250mV</td>
<td>25A (each regulator, 50A total)</td>
<td>200mV</td>
</tr>
<tr>
<td>H745</td>
<td>-15Vdc + 750mV</td>
<td>1PA</td>
<td>450mV</td>
</tr>
<tr>
<td>H754</td>
<td>+20Vdc + 1V</td>
<td>8A</td>
<td>5%*</td>
</tr>
<tr>
<td></td>
<td>-5Vdc + 250mV</td>
<td>1A - 8A**</td>
<td>5%*</td>
</tr>
<tr>
<td>54 .1086</td>
<td>+15Vdc + 1.5V</td>
<td>4A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+8Vdc</td>
<td>1A</td>
<td></td>
</tr>
</tbody>
</table>

* At backplane. Typical ripple +3%.

** Maximum +5V current is dependent upon +20V current. It is equal to 1A plus the current of the +20V supply, up to a total of 8A.

Maximum input power 11000 watts

**NOTE**

Total of current drawn from 54-110V, +15 volts and +8 volts cannot exceed 4 Amps.
<table>
<thead>
<tr>
<th>Column</th>
<th>CPU Backplane only</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Modules</td>
<td>+5</td>
<td>+5</td>
<td>+15</td>
<td>-15</td>
<td>+20</td>
<td>-20</td>
<td>-20</td>
<td>-20</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Power</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dissipated</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Max, Allowable</td>
<td>25A</td>
<td>25A</td>
<td>1.5A</td>
<td>10A</td>
<td>8A</td>
<td>1A-8A*</td>
<td>1000A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Values</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Maximum -5V current is dependent upon +20V current. It is equal to 1A plus the current of the +20V power supply, up to a total of 8A.*
INSTALLATION

GUIDE
TABLE OF CONTENTS

CHAPTER 6

6.1 ENVIRONMENTAL REQUIREMENTS
6.2.1 Humidity and Temperature
6.2.2 Air Conditioning
6.2.3 Special Mounting Conditions
6.2.4 Static Electricity

6.3 ELECTRICAL REQUIREMENTS
6.3.1 System Grounding

6.4 UNPACKING

6.5 INITIAL INSPECTION
6.5.1 Inspection Areas

6.6 SWITCH SETTINGS
6.6.1 M9301
6.6.2 DL11-W = M7856
6.6.3 MOS MEMORY = M7847

6.7 MODULE PLACEMENT

6.8 FIRST TIME START-UP

6.9 CHECK STANDBY OPERATION

6.10 CABINET MOUNTING
6.10.1 5 1/4" Box BA11-L
6.10.2 10 1/2" Box BA11-K

FIGURES

6.1 5 1/4" and 10 1/2" Box AC voltage connector specifications
6.2 861-B and 861-C Power Controller connectors for cabinet mounted models
6.3 5 1/4" Computer Packing
6.4 10 1/2" Computer Packing
6.5 5 1/4" Computer Sub-Assemblies
6.6 10 1/2" Computer Sub-Assemblies
6.7 DD11-PK 9 slot backplane
6.8 15 pin mate "n" lock connector 6.9 6 pin mate "n" lock connector
6.10 10 pin 3M connector
6.11 M9301 Bootstrap Terminator Module
6.12 DL11-W (M7856) Module
6.13 DL11-W Device Address
6.14 DL11-W vector Address
6.15 M7847 MOS Memory Module
6.16 5 1/4" Computer Module Placement
6.17 10 1/2" Computer Module Placement
6.18 Operator's Console
6.19 Cabinet Mounted 5 1/4" Box BA11-L
6.20 Cabinet Mounted 10 1/2" Box BA11-K
6.21 Rear Mounting Bracket for BA11-K
WARNING

AC POWER IS APPLIED TO SOME AREAS OF THE 11/34 COMPUTER WHEN THE LINE COPD IS PLUGGED INTO THE AC RECEPTACLE.

ENSURE COMPUTER IS UNPLUGGED BEFORE WORK IS DONE INSIDE THE COMPUTER HOUSING.
6.1 SCOPE

This chapter provides information on PDP11/34 environmental and electrical requirements, equipment inspection and first time start up installation of the basic PDP11/34 central processor units.

A space is provided to the left of every section or sequenced step, to be checked or initialed upon completion of that section, to ensure that all sections have been read and none have been skipped or omitted.

Installation procedures for the various available peripherals are provided in the maintenance manuals supplied with the peripherals.

6.2 ENVIRONMENTAL REQUIREMENTS

6.2.1 Humidity and Temperature

The PDP11/34 systems are designed to operate in a temperature range from 41 F (5 C) to 122 F (50 C) at a relative humidity of 10 to 95 percent, without condensation. However, system configurations that use I/O devices, such as magnetic tape units, card readers, disks, etc., require an operational temperature range of from 60 F (15 C) to 80 F (27 C) with 40 to 60 percent relative humidity. Nominal operating conditions for a system configuration are a temperature of 70 F (20 C) and a relative humidity of 45 percent.

6.2.2 Air Conditioning

When used, computer room air conditioning equipment should conform to the requirements of the "Standard for Installation of Air Conditioning and Ventilating Systems "Non-residential" N.F.P.A No. 75.

6.2.3 Special Mounting Conditions

If the PDP11/34 is to be subjected to rolling, pitching, or variation of the mounting surface (e.g., aboard a ship), the cabinet in which the PDP11/34 is housed should be securely anchored to the installation floor by mounting bolts. Since such installations require modifications to systems cabinets, Digital Equipment Corp. must be notified upon placement of the purchase order so that necessary modifications can be made.

6.2.4 Static Electricity
Static electricity can be an annoyance to personnel and can, in extreme cases, affect the operational characteristics of the PDP11/34 computers and related peripherals. If carpeting is installed on the installation room floor, it should be a type designed to minimize static electricity. Flooring consisting of metal panels, or flooring with metal edges, should be adequately grounded.

6.3 ELECTRICAL REQUIREMENTS

The PDP11/34 computer can be operated from a nominal 115V, 50/60HZ or 230V, 50/60HZ AC power source. The primary AC operational voltages should be maintained within the defined tolerances. Line voltages should be maintained within 90VAC-132VAC or 180VAC-264VAC, and the 50/60HZ line frequency should not vary more than 1 HZ.

BA11-L, 5 1/4" Box

The minimal configuration of the 11/34 housed in this box is approximately 10VA of input power ( A @115VAC, A@230VAC).

The voltage and frequency configuration is determined by the configuration of the H777 power supply.

BA11-K, 10 1/2" Box

The minimal configuration of the PDP11/34 housed in this box is approximately 15VA of input power ( A@115VAC, A@230VAC).

The voltage and frequency configuration is determined by the AC input box.

50HZ 230V = 7009811-2
60HZ 115V = 7009811-1
50HZ 115V = 7009811-1

The primary power outlets at the installation site must be compatible with the PDP11/34 primary power input connectors, or if used with the primary power input connectors of the 861 power controller. The power receptacles must be provided by the customer.
The PDP11/34 uses only two types of connectors, depending on whether it is configured for 115V or 230VAC operation. Figure 6.1 shows the plug portion of each connector and a table is provided giving particular information about both plugs and receptacles.

**Connector Specifications**

<table>
<thead>
<tr>
<th>Description</th>
<th>HEMA Configuration</th>
<th>Poles</th>
<th>Wires</th>
<th>Plug</th>
<th>Receptacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>115V, 15 AMP</td>
<td>5-15</td>
<td>2</td>
<td>3</td>
<td>90-08938</td>
<td>12-05351</td>
</tr>
<tr>
<td>230V, 15 AMP</td>
<td>6-15</td>
<td>2</td>
<td>3</td>
<td>90-08853</td>
<td>12-11204</td>
</tr>
</tbody>
</table>

*ADD P SUFFIX FOR PLUG
ADD R SUFFIX FOR RECEPTACLE

Figure 6.1 5 1/4" and 10 1/2" Box AC Voltage Connector Specifications
For cabinet mounted 11/34 models, the 861 Power Controller is used. It requires different connectors from the 5 1/4" and 10 1/2" basic box. The 861-C Power Controller is used for 115VAC operation and the 861-B is used for 230VAC operation. Figure 6.2 illustrates these connectors and the associated chart provides connector specifications.

### Figure 6.2 861-B and 861-C Power Controller Connectors for Cabinet Mounted Models

<table>
<thead>
<tr>
<th>Model Number</th>
<th>Power</th>
<th>Rating</th>
<th>Plug</th>
<th>Receptacle (Supplied by Customer)</th>
<th>DEC Part No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>861-C</td>
<td>115 V Single Phase</td>
<td>30 A</td>
<td>L5-30P</td>
<td>L5-30R</td>
<td>12-01191</td>
</tr>
<tr>
<td>861-B</td>
<td>230 V Single Phase</td>
<td>20 A</td>
<td>L6-20P</td>
<td>L6-20R</td>
<td>12-11194</td>
</tr>
</tbody>
</table>
6.3.1 Ensure proper system grounding

The PDP-11/34 three-prong power connector, when inserted into a properly wired receptacle, should ground the computer chassis. It is unsafe to operate the computer unless the case is grounded because normal current leakage from the power supply flows to the metal parts of the chassis. If the integrity of the ground circuit is questionable, the user is advised to measure the potential between the computer case and a known ground with a voltmeter, or to notify your field service representative.

Computer systems are often sensitive to the interference present on some AC power lines. If the computer is to be installed in an electrically noisy environment, it may be necessary to provide primary power to the computer on a separate power line from lighting, air conditioning, etc., so that computer operation is not affected by voltage surges or fluctuations.

Any questions regarding power requirements and installation wiring should be directed to the Digital Sales Engineer or Field Service Engineer.

6.4 UNPACKING

Basic computers are shipped in the packages illustrated in Figure 6.3 for the 5 1/4" version and Figure 6.4 for the 1" 1/2" PDP11/34. Please study these figures before unpacking the computer.
FIGURE 6.3  5 1/4" COMPUTER PACKING
Upon completion of unpacking, extend the wire from assembly containing the logic and power supply sub-assemblies. See Figure 6.5 for the 5 1/4" Computer and Figure 6.6 for the 10 1/2" Computer.
FIG. 6.6  10½" COMPUTER SUB-ASSEMBLIES
6.5.1 Examine the following areas:

1. General overall appearance, i.e., scratches, dents, chipped paint, dust or foreign material.

2. Check for loose or missing hardware, i.e., screws, nuts, etc.

3. Toggle front panel switches to ensure each switch operates free and unrestricted.

4. Examine power and console harnesses for proper connection to the power supply and front console, respectively. See Figures 6.7, 6.8, 6.9, and 6.10.

FIGURE 6.7 DD11-PK (PIN SIDE) 9 SLOT BACKPLANE
<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>WIRE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>#14 Red</td>
</tr>
<tr>
<td>2</td>
<td>+15V</td>
<td>#14 Gray</td>
</tr>
<tr>
<td>3</td>
<td>+20V</td>
<td>#14 Orange</td>
</tr>
<tr>
<td>4</td>
<td>+5V</td>
<td>#14 Red</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>6</td>
<td>+15V Bat</td>
<td>#14 Gray</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>10</td>
<td>Not Used</td>
<td>#14 Black</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>#14 Black</td>
</tr>
<tr>
<td>12</td>
<td>+5V Bat</td>
<td>#14 Red</td>
</tr>
<tr>
<td>13</td>
<td>-15V</td>
<td>#14 Blue</td>
</tr>
<tr>
<td>14</td>
<td>-5V</td>
<td>#14 Brown</td>
</tr>
<tr>
<td>15</td>
<td>-15V Bat</td>
<td>#14 Blue</td>
</tr>
</tbody>
</table>

**Figure 6.8** MATE "N" LOCK CONNECTOR 15 PIN
<table>
<thead>
<tr>
<th>PIN</th>
<th>Signal</th>
<th>Wire Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LO GND</td>
<td>#20 Black</td>
</tr>
<tr>
<td>2</td>
<td>LTC</td>
<td>#20 Brown</td>
</tr>
<tr>
<td>3</td>
<td>DC LO</td>
<td>#20 Violet</td>
</tr>
<tr>
<td>4</td>
<td>AC LO</td>
<td>#20 Yellow</td>
</tr>
<tr>
<td>5</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Not Used</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6.9** Mate 'N' Lock Connector 6 Pin
<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INIT</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>SACK</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>HALT GRANT</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>AC LO</td>
</tr>
<tr>
<td>8</td>
<td>HALT REQUEST</td>
</tr>
<tr>
<td>9</td>
<td>DC LO</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
</tr>
</tbody>
</table>

**FIGURE 6-10 3M CONNECTOR**
6.6 SWITCH SETTINGS

The settings of switches and jumpers, as shipped from the factory, are shown on the decal on top of the box. Refer to the following figures for physical location of jumpers and switches, and brief explanation of their function.

6.6.1 M9301, The Boot Strap Terminator

See Figure 6.11

Function of P-1

P1-S01 LOW ROM ENABLE, ADDRESS 765XXX
P1-S02 POWER UP REBOOT ENABLE
P1-S03 ROM 008 P1-S04 ROM 007
P1-S05 ROM 006
P1-S06 ROM 005
P1-S07 ROM 004
P1-S08 ROM 003
P1-S09 ROM 002
P1-S10 ROM 001

P1-S03 thru P1-S10 enable address range for the LOW ROM of 765000-765776, HIGH ROM 773000-773776

With battery back-up, a good battery, and TPl on the M9301 connected, the system does not go into the bootstrap routine on power restart. If there is no battery back-up (i.e. system uses core memory) or if the battery is low and TPl is connected the system will go into the bootstrap routine on a power restart. Therefore, in a system without battery back-up, TPl on the M9301 must be disconnected if the user does not want to go into the bootstrap routine on all power restarts.
To
Be
Supplied

Figure 6.11  M9301 BOOTSTRAP TERMINATOR MODULE
6.6.2 Remove the DL11-W (may be optional) M7856, combination serial line/real time clock

Refer to Figure 6.12.

1. Function of P-1 - Active/Passive mode of 2IIMA loop

- P1-S01
- P1-S02
- P1-S03
- P1-S04
- P1-S05
- P1-S06
  TRANSMITTER - OFF

NOTE

P1-S01 thru P3-S10 are shown in the active state. P3-S06 thru P3-S10 are the active/passive switches for the receiver, see Function of P3.

2. Function of P-2 - Used to determine VECTOR ADDRESS.

- P2-S01 - NOT USED
- P2-S02 - NOT USED
- P2-S03 - BIT 05 - ON
- P2-S04 - BIT 03 - OFF
- P2-S05 - BIT 06 - OFF
- P2-S06 - BIT 04 - ON
- P2-S07 - BIT 07 - OFF
- P2-S08 - BIT 08 - OFF

NOTE

Shown with a vector address of 60---normally the Vector Address of the console device.
3. Function of P-3 - Selectable PCUR/TRANS Bands, Receive Active/Passive Modes

<table>
<thead>
<tr>
<th>P3-S01</th>
<th>TRANSMITTER BAUD SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3-S02</td>
<td>RECEIVER BAUD SELECT</td>
</tr>
<tr>
<td>P3-S03</td>
<td>RECEIVER BAUD SELECT</td>
</tr>
<tr>
<td>P3-S04</td>
<td>TRANSMITTER BAUD SELECT</td>
</tr>
<tr>
<td>P3-S05</td>
<td>RECEIVER BAUD SELECT</td>
</tr>
<tr>
<td>P3-S06</td>
<td>RECEIVER = ON</td>
</tr>
<tr>
<td>P3-S07</td>
<td>RECEIVER = OFF</td>
</tr>
<tr>
<td>P3-S08</td>
<td>RECEIVER = ON</td>
</tr>
<tr>
<td>P3-S09</td>
<td>RECEIVER = OFF</td>
</tr>
<tr>
<td>P3-S10</td>
<td>RECEIVER = ON</td>
</tr>
</tbody>
</table>

**NOTE**

P4-S10 is a Transmitter Baud Select Switch, see function of P4. P3-S06 thru P3-S10 are shown in the active position.

4. Function of P4

<table>
<thead>
<tr>
<th>P4-S01</th>
<th>&quot;BREAK&quot; ENABLE, DISABLE THE TRANSMITTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4-S02</td>
<td>PARITY SELECT (ODD OR EVEN)</td>
</tr>
<tr>
<td>P4-S03</td>
<td>NUMBER OF DATA BITS SELECT (CHARACTER LENGTH)</td>
</tr>
<tr>
<td>P4-S04</td>
<td>NUMBER OF DATA BITS SELECT (CHARACTER LENGTH)</td>
</tr>
<tr>
<td>P4-S05</td>
<td>NUMBER OF STOP BITS</td>
</tr>
<tr>
<td>P4-S06</td>
<td>PARITY ENABLE</td>
</tr>
<tr>
<td>P4-S07</td>
<td>TO ENABLE THE ERROR BITS TO THE UNIHUS</td>
</tr>
<tr>
<td>P4-S08</td>
<td>NOT USED</td>
</tr>
<tr>
<td>P4-S09</td>
<td>NOT USED</td>
</tr>
<tr>
<td>P4-S10</td>
<td>TRANSMITTER BAUD SELECT</td>
</tr>
</tbody>
</table>

5. Function of P5 - Device Address Selection and Line Clock Enable

<table>
<thead>
<tr>
<th>P5-S01</th>
<th>ADDRESS BIT 08 OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5-S02</td>
<td>ADDRESS BIT 09 OFF</td>
</tr>
<tr>
<td>P5-S03</td>
<td>ADDRESS BIT 10 OFF</td>
</tr>
<tr>
<td>P5-S04</td>
<td>ADDRESS BIT 07 ON</td>
</tr>
<tr>
<td>P5-S05</td>
<td>ADDRESS BIT 06 OFF</td>
</tr>
<tr>
<td>P5-S06</td>
<td>ADDRESS BIT 05 OFF</td>
</tr>
<tr>
<td>P5-S07</td>
<td>ADDRESS BIT 03 ON</td>
</tr>
<tr>
<td>P5-S08</td>
<td>ADDRESS BIT 04 OFF</td>
</tr>
<tr>
<td>P5-S09</td>
<td>LINE CLOCK ENABLE OFF</td>
</tr>
<tr>
<td>P5-S10</td>
<td>LINE CLOCK ENABLE ON</td>
</tr>
</tbody>
</table>

**NOTE**

Shown with Device Address of 777560.
with Line Clock Enable. For the Line Clock to be used, it must be enabled and the DLI1-w address must be 777560.

Unless otherwise specified by the customer the first DLI1-w will be used as a console device, and will be set up to the following parameters.

**DEVICE ADDRESS = 777560**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>OFF</td>
</tr>
<tr>
<td>09</td>
<td>ADDRESS BIT 10</td>
</tr>
<tr>
<td>08</td>
<td>SWITCH SELECTABLE BUS</td>
</tr>
<tr>
<td>07</td>
<td>ADDRESS BITS</td>
</tr>
<tr>
<td>06</td>
<td>ON = LOGICAL 0</td>
</tr>
<tr>
<td>05</td>
<td>OFF = LOGICAL 1</td>
</tr>
<tr>
<td>04</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>ADDRESS BIT 03</td>
</tr>
</tbody>
</table>

**OCTAL ADDRESS**

<table>
<thead>
<tr>
<th>ADDRESS BIT</th>
<th>7</th>
<th>7</th>
<th>7</th>
<th>5</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOT SWITCH SELECTABLE**

**SWITCH SELECTABLE**

**NOT SWITCH SELECTABLE**

**FIGURE 6.13 DEVICE ADDRESS**

**VECTOR ADDRESS = 000060** See Figure 6.14

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>OFF</td>
</tr>
<tr>
<td>07</td>
<td>DATABIT 08</td>
</tr>
<tr>
<td>06</td>
<td>OFF</td>
</tr>
<tr>
<td>05</td>
<td>SWITCH SELECTABLE</td>
</tr>
<tr>
<td>04</td>
<td>ON</td>
</tr>
<tr>
<td>03</td>
<td>DATABIT 03</td>
</tr>
</tbody>
</table>
FIGURE 6.14 VECTOR ADDRESS

BAUD SELECTION either 110 or 300

1. BAUD 110
   TRANSMITTER BAUD

   SWITCH          POSITION
   P4=S10          ON
   P3=S01          ON
   P3=S04          ON

   RECEIVER BAUD

   SWITCH          POSITION
   P3=S02          OFF
   P3=S03          OFF
   P3=S05          OFF

2. BAUD 300
   TRANSMITTER BAUD
SWITCH  POSITION
P4-S10   ON
P3-S01   OFF
P3-S04   OFF

RECEIVER BAUD

SWITCH  POSITION
P3-S02   OFF
P3-S03   ON
P3-S05   ON

ACTIVE OR PASSIVE

The TRANSMITTER, RECEIVER, and READER ENABLE switches will all be ACTIVE. See PARAGRAPH 6.6.2 - Function of P1 and Function of P3 for proper switch settings.

BREAK

WILL BE DISABLED
P4-S01 OFF POSITION

PARITY

WILL BE DISABLED
P4-S06 OFF POSITION

PARITY SELECT
P4-S02 OFF POSITION

NUMBER OF DATA BITS (CHARACTER LENGTH)
P4-S04 OFF
P4-S03 OFF
Set for eight (8) DATA BITS.

STOP BITS
P4-S05 OFF
Set for two (2) STOP BITS.

ENABLE THE ERROR BITS TO THE UNIBUS
P4-S07 OFF
Set to the DISABLE POSITION

LINE CLOCK
P5-S09 OFF
P5-S10 ON
LINE CLOCK ENABLED
6.6.3 REMOVE THE M7847 MOS Memory Module

**Figure 6.15 M7947 MOS Memory Module**

Setting of P1 for first 16K of MOS memory,

- P1=S01 OFF
- P1=S02 OFF
- P1=S03 OFF
- P1=S04 ON
- P1=S05 ON
- P1=S06 OFF
- P1=S07 ON
- P1=S08 OFF

**NOTE**

For extra banks of memory, see the Configuration Guide for address selection of those banks.

6.7 ENSURE PROPER MODULE PLACEMENT

See Figure 6.16 for 5 1/4" Computer and 6.17 for 10 1/2" Computer Module Placement.
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M7265</td>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>M7266</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M9301 **</td>
<td>*DIIW SERIAL LINE/LINE CLOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BOOTSTRAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>MS11-JP 16K 18-BIT MOS MEMORY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>M7850 PARITY CONTROL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>EXPANSION 3 HEX SPS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>M9302 BUS TERMINATOR</td>
<td>EXPANSION 1 QUAD SPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* OPTIONAL IN PDP-11/34
* * MAYBE M9301-YA OR YB
1 DOUBLE HEIGHT MODULE
OCCUPIES SLOTS A AND B

**FIGURE 6.16 5 1/4" COMPUTER MODULE PLACEMENTS**
**Double System Unit**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M7265</td>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>M7266</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M9301 Bootstrap</td>
<td>DL11W Serial Line/Line Clock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MS11-JP 16K-18 bit MOS Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>M7850 PARITY CONTROL</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>M9302 UNIBUS Terminator</td>
<td>G727</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Expansion: 3 System Units**

---

* OPTIONAL IN 11/34
**MAY BE M9301-YA OR M9301-YB
1 DOUBLE HEIGHT MODULE, OCCUPIES SLOTS A & B

---

**WARNING**

AC POWER IS APPLIED TO SOME AREAS OF THIS COMPUTER WHEN THE LINE CORD IS PLUGGED INTO THE AC RECEPTACLE.

---

6.8 FIRST TIME START UP

---

**Figure 6-17 105" Computer Module Placement**
The following step by step procedure should be followed strictly, ensuring no steps are skipped or omitted. If any problem occurs or proper indications are not received, please refer to the Trouble Shooting Guide of this manual or notify your Field Service representative.

NOTE

Read each step completely before attempting it!

The START-UP procedure assumes the following:

1. A console device controller, DL11-8 or equivalent.
2. Teletype or equivalent is "ON LINE".
3. Operators Front Console, KY11-LA, is installed.

6.8.1.1 First Time Start-Up Procedure

STEP 1

APPLY AC POWER

Before applying AC power, ensure proper AC voltage and frequency are applied to computer. Check decal at rear of the box.

<table>
<thead>
<tr>
<th>VARIATION</th>
<th>DECAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>60Hz</td>
<td>115V</td>
</tr>
<tr>
<td>50Hz</td>
<td>240V</td>
</tr>
<tr>
<td>50Hz</td>
<td>115V</td>
</tr>
</tbody>
</table>

STEP 2

Place the power switch in the "OFF" position. Insert the line cord plug into the AC receptable place the "HALT/CONT" switch in the "CONT" position. Rotate the "OFF/ON/STANDBY" rotary switch to the "ON" position. See Figure 3.0 for location of switches. Figure 2.10, FIGURE 6.10 OPERATOR'S CONSOLE

STEP 3

Check for the following indications:

1. Fan rotation
2. "DC" Power Indicator lit.
3. "BATT" status indicator on or blinking if battery backup
option is present. It will be OFF if not present.

4. The console device received a keyboard dispatch character (S) on its display.

5. "RUN" indicator lit.

STEP 4

Verify communications between the console keyboard and the computer.

SYMBOLS
"D" Denotes DEPOSIT
"E" Denotes EXAMINE
"L" Denotes LOAD ADDRESS
"SB" Denotes SPACE BAR
"CR" Denotes CARRIAGE RETURN

ITEM 1. LOAD ADDRESS. On the console keyboard type the following: "L", "SB", 200, "CR". This loads memory Location 200 into an internal register.

ITEM 2. EXAMINE LOCATION 200. Type the following: "E", "SB", "CR". The number 200 will be displayed followed by the data contained in location 200.

ITEM 3. DEPOSIT all "1's" into location 200. Type the following: "D", "SB", 177777, "CR". This causes Location 200 to be loaded with all "1's".

ITEM 4. EXAMINE Location 200 for all "1's". Type the following: "E", "SB". The number 200 should be displayed followed by 177777.

ITEM 5. RESTORE original contents of 200. Type the following: "D", "SB", followed by the data obtained in Item 2, "CR".

6.8.2

Run the PDP11/34 diagnostic. Bootstrapping a diagnostic from a peripheral device will be done at this time. Refer to the "Operator's" section of this manual to accomplish this.

After booting, load the starting address 200. Type the following: "L", "SB", 200, "CR" followed by; "S", "CR". Let the diagnostic run for approximately 30 minutes. Periodic end of pass indicators will be typed.

6.9 CHECK STANDBY OPERATION - MOS MEMORY MUST BE INSTALLED.
6.9.1

After verification of proper computer operation by the diagnostics, the standby check can be made.

STEP 1. Place the HALT/CONT switch in the "HALT" position. This causes the computer to halt. The "RUN" indicator will extinguish.

STEP 2. Rotate the "OFF/ON/STANDBY" rotary switch to the "STANDBY" position. Check the following indications:
1. "DC" indicator will extinguish.
2. "BATT" status indicator will be lit or blinking.

REMAIN in the "STANDBY" position for approximately 5 minutes.

STEP 5. Place the HALT/CONT switch in the "CONT." position. Rotate "OFF/ON/STANDBY" switch back to the "ON" position.
1. At the console keyboard, type the following. "L", "SB", 200, "CR". This load address is location 200, the starting address of the diagnostic. Follow this by, "S", "CR". The diagnostic should run.

6.10 CABINET MOUNTING

6.10.1

Mounting the 5 1/4" BALL-L Box. See Fig. 6.19

Hardware
1. 16 10-32 by 1/2" screws=phillips head
2. 16 10-32 tinnerman nuts
3. 16 10-32 str. washers

6.10.1.1 - Remove wire frame assembly from outer shell. See Fig. 6.5.

6.10.1.2 - Push tinnerman nuts onto the frame of the cabinet, on the 38th, 39th, 40th, and 41st holes, from the bottom of the cabinet. Repeat on all four posts of the cabinet.

6.10.1.3 - Align the outer shell with the tinnerman nuts and insert the 10-32 by 1/2" screws, with star washers.
6.10.1.4 - Place the box into the outer shell.

6.10.2 - Mounting the 10 1/2" BALL-K Box. See figures 6.2(d) and 6.21.

Hardware

1. 8 10-32 by 1/2" screws = phillips head-stainless steel
2. 8 10-32 tinnerman nuts
3. 8 10-32 star washers
4. 2 "L" brackets
5. 4 10-32 by 1/2" screws
6. 2 BARS.

6.10.2.1 - Remove Shipping bracket

6.10.2.2 - Remove chassis slides from 10 1/2" box.

The right and left side are different. Ensure the slides are mounted in the cabinet the same way they are taken off the box.

6.10.2.3 - Push the tinnerman nuts on to the frame of the cabinet, on the 39th and 40th holes, from the bottom of the cabinet. Repeat on all four posts of the cabinet.

6.10.2.4 - Mount the "L" brackets to the chassis slide. See Fig. 6.21.

Leave the screws slightly loose so that the slide can be adjusted to the size of the cabinet.

Figure 6.21 shows the right side chassis slide, looking from the front of the box. For the left side, swap sides, place the "MAR" on the left and the "L" bracket on the right.

6.10.2.5 - Align the chassis slides over the tinnerman nuts and insert the 10-32 by 1/2" screws (stainless steel) through the slide into the tinnerman nuts.

Tighten the screws left loose in the previous step, after adjusting to the cabinet size.
6.10.2.6 - Place the box into the chassis slide.
FIGURE 6.19  CABINET MOUNTED 5 1/4" BOX BALL-L
FIGURE 6.20 CABINET MOUNTED 10 1/2" BOX BALL-BEARING BRACKET SEE Fig. 5-22

*REMOVE WHEN COMPUTER IS POWERED UP. (SHIPPING BRACKET). + MOUNT ONLY THE TOP TWO (2) SCREWS ON ALL FOUR BRACKETS.
FIGURE 6.21 REAR MOUNTING BRACKET (RIGHT SIDE)
FOR BANK 10 1/2" BOX.
7.0 TROUBLESHOOTING GUIDE

7.1 Introduction

There are two possible courses of action in the event of an apparent failure - call DEC Field Service and report the failure, or attempt to isolate the cause of the failure and correct it.

7.1.1 DEC Field Service Support

DEC Field Service will service the PDP-11/34 on a per call basis or on a contract basis. For more details contact your local DEC Field Service Branch Office.

Before you call Field Service to perform corrective maintenance, there are certain fast checks that may allow you to repair the problem simply. In the event that these do not isolate the cause of failure there is specific information that you can provide that is required by Field Service regarding the symptoms of the failure. See section 2 - How to Report a Failure for cursory check and required failure information.

7.1.2 Self-Help

Alternatively, it may be possible for you to correct the failure yourself. This should not be attempted unless you are familiar with the PDP-11/34 hardware. For information regarding the availability of training contact DEC Educational Service (617-897-5111 x3819).

Troubleshooting and repairing the PDP-11/34 will require the availability of spare parts, special tools, and diagnostics (See TABLE 7-1).

CAUTION: Extreme care must be taken when troubleshooting as it is possible to come in contact with hazardous electrical voltage, void any existing warranty, and further damage the equipment.

Qualified technicians may utilize any of a number of techniques in order to isolate and correct failures. See section 3 - Troubleshooting the PDP-11/34 AND TABLE 7-1.

7.2 HOW TO REPORT A FAILURE
PRIOR TO CALLING YOUR LOCAL DEC Field Service Branch Office there are some cursory checks and observations that should be made.

7.2.1 Quick Checks

Often, hardware failures are caused by things that are readily diagnosed and easily corrected by someone without detailed technical knowledge. Some cursory checks that may help detect the cause of a failure are as follows:

1. Is power applied?
2. Are all circuit modules firmly seated and in accordance with configuration rules? (Grant continuity cards present as required?)
3. Are all options energized or selected where applicable? (Teletype switched to LINE? DECTape switched to REMOTE? etc.)
4. Are all option switch settings proper? (DIII-2: baud rate, data format, address, vector; memory: address, etc.)
5. Is there any apparent operator error? (Issue DEPOSIT command without Load Address Command first, etc.)

7.2.2 Failure Symptoms

Should you call DEC Field Service to report an error, there are symptoms or indications that you can provide which are used by Field Service in troubleshooting the PDP-11/34. Having this information available when calling Field Service may greatly reduce the amount of time required to repair your system. The information required is as follows.

1. Is the RUN light ON or OFF?
2. Is the DC ON light ON or OFF?
3. Has the console device printed a "$"?
4. Did the console device echo any commands that you issued?
5. Did the computer successfully execute the bootstrap?

7.3 Troubleshooting the PDP-11/34

The following procedure is intended as a guideline only. It is starting point that applies one possible troubleshooting technique to
a given hardware system. It is not meant to be an all-encompassing cookbook procedure which excludes any other actions. It should be modified and built upon to conform with your individual preferences, based on your background and experience.

7.3.1 What This Method Consists Of

If the symptoms and failure indications seem to point to a specific (or general) location, investigate that possibility. If not, or if your investigation leads nowhere, you should start from ground zero. Assume as little as possible and pretend the system is going together for the first time.

This method of troubleshooting the computer system is based on the technique of bringing a system up one part at a time, in a logical manner, (rebuilding the system). If a small part of the system can be found to be functional it can be used to prove out another part and thereby expand your working system. We can continue to build the system in this manner, replacing faulty modules as they are located, until we have the entire system functioning.

7.3.2 What To Do

(Note - This section assumes that the system consists of at least the following: Bailey Box, DD11 Backplane, KD11-E processor, M9301 terminator, M9302 terminator, DL11-W combined option, console device (i.e., LA36, LT33, etc.) KY11-LA operator's console, and MM11 or MS11 memory).

It may be possible to correct the problems without "rebuilding" the system by first performing some cursory check. Some of these are:

- Is power applied?
- Are all modules firmly seated and in accordance with configuration rules? (i.e., Grant continuity cards present where needed, etc.)
- Are all Unibus modules energized or selected where applicable? (i.e., Teletype switched to LINE, DECTape switched to REMOTE, etc.)
- Are all module switch selections proper? (i.e., DL11-W baud rates, data format, address, and vector, memory address, etc.)
- This there any apparent operator error? (i.e., issue deposit command without Load Address command first, etc.)

If you must "rebuild" the system some of the steps that you take are:
Strip the system by removing all modules except: Processor, M9301 terminator, Operator's console, DL11-W, console device, and M9302 terminator. (Set the M9301 switches to enter the console emulator on power up.) Check power supply outputs.

Upon powering the console device should print the contents of R0, R4, SP, PC, and a prompt character ($) indicating it is ready to input a command. (Refer to Operator's Guide in User's Manual for an explanation of the meaning of these numbers.) If you do not get a prompt character, the following possibilities exist: 1= the CPU has failed the M9301 resident Go Nogo test. (The RUN light should be on, if it is off it indicates detection of an abnormal error, i.e., double bus error, etc.), 2= the DL11-W and/or device are malfunctioning, 3= the M9301 is malfunctioning, 4= backplane is bad. (Note: to check the CPU, replace one board at a time, starting with M7265.

If you get a prompt character, examine location 173000. If you can do this, start execution at location 173000. You should get another printout of R0, R4, SP, PC, and $. If you don't or if you can't do this, the following may be bad: 1= DL11-W, 2= M9301, 3= CPU, 4= backplane.

If you get a "$", add a memory to the system. You should still be able to get a prompt character with power on. Verify that you can deposit and examine in memory. (Use $, V, D, and E commands to deposit 6 into location 4 and 000000 into location 6 and then check their proper loading). If you can't, check 1= backplane (try to use a different slot), 2= memory, 3= CPU, 4= M9301.

Give the system a boot command for a device not presently on the system. It will not execute a successful bootstrap, however, it will do some tests before it attempts to execute a bootstrap. If the CPU halts activate the BOOT switch. Compare the printout of the PC with the M9301 Memory Map to determine which test failed as an indication to faulty CPU or faulty memory. A bad M9301 or backplane is also possible.

If you do not get a halt you should get a print out of the contents of R0, R4, SP, PC and "$". Ensure the PC contains the address of the top of memory. If not check: 1= memory, 2= M9301, 3= CPU, 4= backplane. If it does, add more memory and repeat (give a boot command, ...) until all memory is in the system.

Add the bootstrap device to the system. You still should get a "$" with power up. Attempt to load the processor diagnostic. If you can't, check the bootstrap device, M9301, backplane, diagnostic medium (i.e., bad disk, torn paper tape, etc.) and M9302.

Once you can load diagnostics, run the available diagnostics for all modules in the system as it now exists. If they all
pass, add options one at a time, running all applicable
diagnostics. If they all pass the total system is now
configured, run a system exerciser, (i.e., DEC/X=11). If
this is okay, refer to Unibus Troubleshooting.
General Notes

- 1. CPU errors - replace M7265 first
- 2. Backplane errors - try different slot

* 11/39 ONLY
Table 7-1 Troubleshooting Requirements

Spare Parts

KY11LA
M7265 CPU
M7266
*M9301 Bootstrap Terminator
*M9302 No Sack Terminator
*M9306 Terminator
H777 Power Supply
*DL11-W Combined Option
*MS11 or Memory
MM11
* M7850 Parity Controller

* as required

Special Tools

Memory margin switch box - required for MM11 memories

diagnostics

CPU Test (MAINDEC = )
M9301 Test (MAINDEC = )
DL11-W Test (MAINDEC = )
MOS Memory Test (MAINDEC = )
DEC X/11 (MAINDEC = )
TRAPS Test (MAINDEC = )
Additional tests as required for your options
APPENDIX A

A.1 EXTENDED ADDRESSING

The console emulator routine normally allows accesses to only the lower 28K of memory and to the I/O page (160000(8) to 177776(8)). However, it is possible by use of memory management to use the console emulator to access beyond 28K for the examine and deposit functions. The reader should be familiar with the concepts of memory management in the KDII-E processor.

A.1.1 Definition of Virtual and Physical Addresses

The processor manipulates 16-bit numbers within general registers and memory locations which it often uses as addresses. These addresses are designated virtual addresses as opposed to physical addresses which are asserted on the Unibus to which devices are hardwired to respond.

A.1.2 Address Mapping Without Memory Management

With memory management disabled (as is the case following depressing the boot switch), a simple hardware mapping scheme converts virtual addresses to physical addresses. Virtual addresses in the 0 to 28K range are mapped directly into physical addresses in the range from 0 to 28K. Virtual addresses on the I/O page (160000-177776) are mapped into physical addresses in the range from 124K to 128K (360000(8)-377776(8)).

A.1.3 Address Mapping With Memory Management

With memory management enabled, a different mapping scheme is used. In this scheme, a relocation constant is added to the virtual address to create a physical or 'relocated' address.
Virtual address space consists of eight 4K banks where each bank can be relocated by the relocation constant associated with that bank. The procedure specified in this section allows the user to:

1. Create a virtual address to type into the Load Address command.
2. Determine the relocation constant to relocate the calculated virtual address into the desired physical address.
3. Enable or disable the memory management hardware.

A.1.4 Creation of a Virtual Address

The easiest way to create a virtual address is divide the 18-bit physical address into two separate fields - a virtual address and a physical bank number. The virtual/address is represented by the lower 13 bits and the physical bank by the upper five bits. This creates a virtual address in virtual bank 0. The calculated relocation constant is placed in the relocation register associated with virtual bank 0.

For example, assume a user wishes to access location 533720. The normal access capability of the console is 0 to 256K. This address (533720) is between the 28K limit and the I/O page (760000-777776), and consequently must be accessed as a relocated virtual address, with memory management enabled. The virtual address is 13720(8) in physical bank 25(8) and is derived as follows:

\[
\begin{array}{cccccccc}
5 & 3 & 3 & 7 & 2 & 0 \\
101 & 011 & 011 & 111 & 010 & 000 \\
\end{array}
\]

\[
\text{PHYSICAL ADDRESS}
\]

\[
\begin{array}{c}
2 \\
5 \\
\text{PHYSICAL} \\
4 \text{K BANK} \\
(25) \\
\end{array}
\]

\[
\begin{array}{c}
13720 \\
\text{VIRTUAL ADDRESS} \\
in \text{VIRTUAL BANK 0}
\end{array}
\]

All locations in either bank #25 (520000-537776) or bank #37 (I/O pages 760000-777776) may be accessed through virtual address 000000-017776 and 160000-177776, respectively. The relocation and descriptor registers in the RD11-E are still accessible since their addresses are within the I/O page.
The relocation constant for physical bank 25 is \(0\times5200\). This constant is added in the relocation unit to the virtual address, as shown, yielding \(533720\).

\[
\begin{aligned}
\times013720 & \quad \text{Virtual address} \\
\times520000 & \quad \text{Relocated Constant (Table A-1)} \\
\times533720 & \quad \text{Physical Address}
\end{aligned}
\]

When memory management is enabled all CPU accesses are relocated. Instructions and data access to the console emulator routine will be relocated through virtual bank 7 since their virtual accesses exist in this bank (see Table A.1 for the corresponding addresses of each of the eight virtual banks). Note that accesses to the I/O page (virtual bank 7) are not automatically relocated with memory management while accesses to the I/O page are automatically relocated when memory management is not utilized.

A.1.5 Memory Management Registers

The relocation constant that is added to the virtual address is stored in a relocation register. One such register exists for each of the eight virtual banks. In addition to the relocation registers, each bank has its own descriptor register which provides information regarding the types of accesses allowed (read only, read or write, or no access).

The memory management logic also provides various forms of protection against unauthorized access. The corresponding descriptor register must be set up along with the relocation register to allow access anywhere within the 4K bank.

A.1.6 Address Assignments

The Unibus addresses of the relocation registers and the descriptor registers are given in Table A.1. The relocation constant to be loaded into the relocation register for each 4K bank is provided in Table A.2. The data to be loaded in the descriptor register to provide read/write access to the full 4K bank is always \(077406\).

The Unibus address of the control register to enable memory management is 177572. This register is loaded with the value \(020001\) to enable memory management and 0 to disable it.

To complete the example previously described (location 533720), the console routine would be as follows:

\[
\begin{aligned}
\$L & \quad 1723406 & /\text{Setting relocation register for virtual bank 6} \\
\$D & \quad 52000 & /\text{To access extended memory} \\
\$L & \quad 1723560 & /\text{Setting relocation register for virtual bank 7}
\end{aligned}
\]
SD  7600
SL  17230
SD  77406
SL  172316
SD  77406
SL  177572
SD  1
SL  13720
SE  examine

/To access the I/O page
/Address of descriptor register, virtual
/bank 0
/Address of descriptor register, virtual
/bank 7
/Address of control register; enable
/memory management
/Virtual address of location desired
/The data in location 533720
/will be typed
Table A.1

Unibus Address Assignments

<table>
<thead>
<tr>
<th>Virtual Address</th>
<th>Virtual Bank</th>
<th>Relocation Register</th>
<th>Descriptor Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>160000-177776</td>
<td>7</td>
<td>172356</td>
<td>172316</td>
</tr>
<tr>
<td>140000-157776</td>
<td>6</td>
<td>172354</td>
<td>172314</td>
</tr>
<tr>
<td>120000-137776</td>
<td>5</td>
<td>172352</td>
<td>172312</td>
</tr>
<tr>
<td>100000-117776</td>
<td>4</td>
<td>172350</td>
<td>172310</td>
</tr>
<tr>
<td>060000-077776</td>
<td>3</td>
<td>172346</td>
<td>172306</td>
</tr>
<tr>
<td>040000-057776</td>
<td>2</td>
<td>172344</td>
<td>172304</td>
</tr>
<tr>
<td>020000-037776</td>
<td>1</td>
<td>172342</td>
<td>172302</td>
</tr>
<tr>
<td>000000-017776</td>
<td>0</td>
<td>172340</td>
<td>172300</td>
</tr>
</tbody>
</table>
Table A.2

<table>
<thead>
<tr>
<th>Physical Bank Number</th>
<th>Relocation Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>0.076000</td>
</tr>
<tr>
<td>36</td>
<td>0.074000</td>
</tr>
<tr>
<td>35</td>
<td>0.072000</td>
</tr>
<tr>
<td>34</td>
<td>0.070000</td>
</tr>
<tr>
<td>33</td>
<td>0.068000</td>
</tr>
<tr>
<td>32</td>
<td>0.064000</td>
</tr>
<tr>
<td>31</td>
<td>0.062000</td>
</tr>
<tr>
<td>30</td>
<td>0.060000</td>
</tr>
<tr>
<td>29</td>
<td>0.056000</td>
</tr>
<tr>
<td>28</td>
<td>0.054000</td>
</tr>
<tr>
<td>27</td>
<td>0.052000</td>
</tr>
<tr>
<td>26</td>
<td>0.050000</td>
</tr>
<tr>
<td>25</td>
<td>0.046000</td>
</tr>
<tr>
<td>24</td>
<td>0.044000</td>
</tr>
<tr>
<td>23</td>
<td>0.042000</td>
</tr>
<tr>
<td>22</td>
<td>0.040000</td>
</tr>
<tr>
<td>21</td>
<td>0.036000</td>
</tr>
<tr>
<td>20</td>
<td>0.034000</td>
</tr>
<tr>
<td>19</td>
<td>0.032000</td>
</tr>
<tr>
<td>18</td>
<td>0.030000</td>
</tr>
<tr>
<td>17</td>
<td>0.026000</td>
</tr>
<tr>
<td>16</td>
<td>0.024000</td>
</tr>
<tr>
<td>15</td>
<td>0.022000</td>
</tr>
<tr>
<td>14</td>
<td>0.020000</td>
</tr>
<tr>
<td>13</td>
<td>0.016000</td>
</tr>
<tr>
<td>12</td>
<td>0.014000</td>
</tr>
<tr>
<td>11</td>
<td>0.012000</td>
</tr>
<tr>
<td>10</td>
<td>0.010000</td>
</tr>
<tr>
<td>9</td>
<td>0.006000</td>
</tr>
<tr>
<td>8</td>
<td>0.004000</td>
</tr>
<tr>
<td>7</td>
<td>0.002000</td>
</tr>
<tr>
<td>6</td>
<td>0.002000</td>
</tr>
<tr>
<td>5</td>
<td>0.002000</td>
</tr>
<tr>
<td>4</td>
<td>0.001600</td>
</tr>
<tr>
<td>3</td>
<td>0.001400</td>
</tr>
<tr>
<td>2</td>
<td>0.001200</td>
</tr>
<tr>
<td>1</td>
<td>0.001000</td>
</tr>
<tr>
<td>0</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
Loading a new relocation constant into the relocation register for virtual bank 0 will cause virtual addresses 000000-177776 to access the new physical bank. A second bank can be made accessible by loading the relocation constant and descriptor data into the relocation and descriptor registers for virtual bank 1 and accessing the location through virtual address 020000-037776. Seven banks are accessible in this manner, by loading the proper constants, setting up the descriptor data, and selecting the proper virtual address. Bank 7 (I/O page) must remain relocated to physical bank 37 as it is accessed by the CPU to execute the console emulator routine.

Memory management is disabled by clearing (loading with 0s) the Control Register 177572. It should always be disabled prior to typing a 'boot' command.

The start command automatically disables memory management and the CPU begins executing at the physical address corresponding to the address specified by the previous Load Address command. Depressing the boot switch automatically disables memory management. The contents of the relocation registers are not modified.

The HALT/CONTINUE switch has no effect on memory management.
APPENDIX A

Introduction:

All information in any digital computer is represented in terms of binary numbers. Since binary numbers are awkward to work with a bit clustering technique is used for binary to octal conversion, yielding more convenient numbers. Octal to binary and decimal to/from binary conversion techniques are also discussed herein. Illustrative examples for all are included.

Comparing Binary Numbers to Decimal Numbers:

It is helpful to analyze the decimal counting system to understand binary numbers. Consider the number 1452. This number can be broken down into pieces: it could be said to be \(1 \times 1000 + 4 \times 100 + 5 \times 10 + 2\), quantities defined as thousands, hundreds, tens, and units. There are ten different states in the decimal system, \(0-9\). Counting up once all the symbols have been used produces a carry into the next column, where the first column starts at the beginning. See the following example. (ex. 1)

<table>
<thead>
<tr>
<th>tens column</th>
<th>units column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>all available symbols</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>carry to 8</td>
<td>8</td>
</tr>
<tr>
<td>tens column</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

Making a careful observation, notice the value placed on the tens column is equal to the number of symbols available, mathematically expressed, this is \(10\) or \(10^1\) to the first power. The hundreds column
is represented as $10^2$ or 10 squared. Each following column is an increasing power of 10, again because 10 is the number of symbols available for expressing numbers.

A comparison of this scheme to binary #’s is simple. Two symbols are now available, one and zero. Counting is as follows: (ex. 2).

\[
\begin{array}{l}
0 \\
1 \\
10 \\
11 \\
100 \\
101 \\
110 \\
111 \\
\end{array}
\]

Again upon exhausting the symbols through counting, a carry is taken to the next column. By establishing the values of each column, it is quite simple to convert a binary number to a decimal number. The value of the first column is either 0 or 1. The value of the second, 2(1); thus this column is either 2 or 0 in decimal. The third, 2(2), or 4 or 0. By taking a summation of the column headings, that have core in that column, the decimal equivalent of the binary number is produced. (ex. 3).

\[
\begin{array}{l}
2(4)=16 \\
2(3)=8 \\
2(2)=4 \\
2(1)=2 \\
2(0)=1 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 1 & 0 \\
\end{array}
\]

Add together the column headings with a one in the respective column.

\[
\begin{array}{c}
16 \\
+ 4 \\
+ 1 \\
\hline \\
21 \\
\end{array}
\]

The decimal equivalent of the binary number $10101$ is 21. (ex. 3)

Converting decimal numbers to binary numbers is also a simple task. Given a decimal number, a one will be put in a binary column if the value of that column can be subtracted from the decimal number. Continue subtracting until the decimal remainder is zero. This is more easily seen in an example. (ex. 4)

Decimal number to be converted = 25.
The highest power of 2 that can be subtracted from 25 is 16 (32 is too large. So...

\[
\begin{array}{l}
2(5)=32 \\
2(4)=16 \\
2(3)=8 \\
2(2)=4 \\
2(1)=2 \\
2(0)=1 \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
1 \\
\end{array}
\]

25
= 16
Anytime a power of 2 can be subtracted from the decimal number, enter a 1 into the corresponding column. Continuing...

\[
\begin{array}{ccccccc}
2(5)=32 & 2(4)=16 & 2(3)=8 & 2(2)=4 & 2(1)=2 & 2(0)=1 \\
0 & 1 & 1 & 0 & 0 & 1
\end{array}
\]

32 - 16 = 16

16 - 8 = 8

8 - 8 = 0

Thus the binary representation of the decimal number 25 = 11001, (ex, 4)

Decimal to/from binary conversion is an operation necessary to enter numerical data from the outside world into the computer. But, as was stated before, there is much more information to be entered into the machine other than numerical data. All of the instructions in the machine are either directly or indirectly coded in binary numbers. These numbers don’t look any different from any other binary number, except to the machine it represents coded information which tells it what to do. If one were to read out the contents of memory (where all of this information is stored), data and instructions are indistinguishable from one another. The machine, though, does know where to find the instructions in memory.

To convert the binary instructions into decimal would not accomplish anything as far as recognizing the instructions. Besides, to do this for hundreds or thousands of instructions would be very time consuming. But recognizing a 16 bit binary instruction is difficult in itself. Compare 10010110010010 to 10010110011001; the numbers differ by 1 bit. A different format is needed to represent these binary numbers.

Binary to Octal Conversion:

We have now discussed the decimal and binary numbering systems, finding that it is convenient for us to deal with decimal numbers but the machine wants binary information. How about a compromise? Examining the properties of octal numbers will allow us to conveniently represent binary numbers. Counting in octal produces the following sequence: 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, ... and so on. Notice there is no 8 or 9. The octal number 10 is equal to the number 8 in decimal. In octal then, there are 8 possible symbols.
Now counting in binary and restricting the number of columns to 3 produces:

**Binary to Octal Conversion cont:**

<table>
<thead>
<tr>
<th>Binary</th>
<th>Octal</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Comparing the equivalent binary, octal, and decimal numbers for this binary count looks like this, (ex. 5).

Notice that three columns of binary information can totally represent the 8 possible values of the octal numbering scheme. In other words, any octal digit can be expressed with three bits of binary, and all symbols in either system are utilized. Conversion between the two systems is the easiest of all. Examine an example of octal to binary conversion; (ex. 6).

Octal # =

| 011 | 111 | 110 |

Simply by inspecting the above chart, determine the binary value of each octal digit and string all the bits together. The octal number 376 is equal to 1111110 in binary.

Binary to octal conversion is just as easy. Given an octal number, group the bits in groups of three. Starting from the right hand side (called the least significant bits or lSBs), Any bits remaining on the left hand side will still form part of an octal number. (ex. 7).

Binary # =

| 10 | 101 | 011 | 110 |

In groups of three ==

| 10 | 101 | 011 | 110 |

Counting to octal ==

| 2  | 5  | 3  | 6  |
The octal representation of the binary number 1010101110 is 2536.
PDP-11 STANDARD ABBREVIATIONS
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>absolute</td>
</tr>
<tr>
<td>A/D</td>
<td>analog-to-digital</td>
</tr>
<tr>
<td>ADC</td>
<td>add carry</td>
</tr>
<tr>
<td>ADRS</td>
<td>address</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>ASL</td>
<td>arithmetic shift left</td>
</tr>
<tr>
<td>ASR</td>
<td>arithmetic shift right</td>
</tr>
<tr>
<td></td>
<td>automatic send/receive</td>
</tr>
<tr>
<td>B</td>
<td>byte</td>
</tr>
<tr>
<td>BAR</td>
<td>bus address register</td>
</tr>
<tr>
<td>BBSY</td>
<td>bus busy</td>
</tr>
<tr>
<td>BCC</td>
<td>branch if carry clear</td>
</tr>
<tr>
<td>BCS</td>
<td>branch if carry set</td>
</tr>
<tr>
<td>BEQ</td>
<td>branch if equal</td>
</tr>
<tr>
<td>BG</td>
<td>bus grant</td>
</tr>
<tr>
<td>BGE</td>
<td>branch if greater or equal</td>
</tr>
<tr>
<td>BGT</td>
<td>branch if greater than</td>
</tr>
<tr>
<td>BHI</td>
<td>branch if higher</td>
</tr>
<tr>
<td>BHIS</td>
<td>branch if higher or same</td>
</tr>
<tr>
<td>BIC</td>
<td>bit clear</td>
</tr>
<tr>
<td>BIS</td>
<td>bit set</td>
</tr>
<tr>
<td>BIT</td>
<td>bit test</td>
</tr>
<tr>
<td>BLE</td>
<td>branch if less or equal</td>
</tr>
<tr>
<td>BLOS</td>
<td>branch if lower or same</td>
</tr>
</tbody>
</table>
BLT       branch if less than
BMI       branch if minus
BNE       branch if not equal
BPL       branch if plus
BR        branch, bus request
BRD       bus register data
BSP       back space
BSR       bus shift register
          back space record
BSY       busy
BVC       branch if overflow clear
BVS       branch if overflow set
CBR       console bus request
CLC       clear carry
CLK       clock
CLN       clear negative
CLR       clear
CLV       clear overflow
CLZ       clear zero
CMP       compare
CNPR      console non-processor request
CNTL      control
COM       complement
COND      condition
CONS      console
CONT      contents
          continue
CP        central processor
CSR       control and status register
D  data
D/A  digital-to-analog
DAR  device address register
DATI  data in
DATIP  data in, pause
DATO  data out
DATOB  data out, byte
DBR  data buffer register
DCDR  decoder
DE  destination effective address
DEC  decrement
Digital Equipment Corporation
DEL  delay
DEP  deposit
DEPF  deposit flag
DIV  divide
DMA  direct memory access
DSEL  device select
DST  destination
DSX  display, X-deflection register
EAE  extended arithmetic element
EMT  emulator trap
ENB  enable
EOF  end-of-file
EOM  end-of-medium
ERR  error
EX  external
EXAM  examine
EXAMF  examine flag
EXEC   execute
EXR    external reset
F      flat (part of signal name)
FCTN   function
FILO   first in, last out
FLG    flag
GEN    generator
IDIVR  integer divide routine
INC    increment
INCF   increment flag
IND    indicator
INH    inhibit
INIT   initialize
INST   instruction
INTR   interrupt
INTRF  interrupt flag
I/O    input/output
IOT    input/output trap
IOX    input/output executive routine
IP     instruction register
IRPD   instruction register decoder
ISR    instruction shift register
JMP    jump
JSR    jump to subroutine
LIFO   last in, first out
LKS    line time clock status register
LOC  location
LP   line printer
LSB  least-significant bit
LSBY least-significant byte
LSD  least-significant digit
LTC  line time clock
MA   memory address
MAR  memory address register
MBR  memory buffer register
MEM  memory
ML   memory location
MOV  move
MSB  most-significant bit
MSBY most-significant byte
MSD  most-significant digit
MSEL memory select
MSYN master sync
ND   negative driver
NEG  negative
NOR  normalize
NPG  non-processor grant
NPR  non-processor request
NPRF non-processor request flag
NS   negative switch
ODT  octal debugging technique
OP   operate
     operation
OPR  operator
     operand
PA  parity available
PAL  program assembly language
PB  parity bit
PC  program counter
PD  positive driver
PDP  programmed data processor
PERIF  peripheral
PGM  program
PP  paper tape punch
PPB  paper tape punch status register
PPS  paper tape punch status register
PR  paper tape reader
PRB  paper tape reader buffer register
PROC  processor
PRS  paper tape reader status register
PS  processor status
positive switch
PTR  priority transfer
PTS  paper tape software system
PUN  punch
RD  read
RDR  reader
REG  register
REL  release
RES  reset
ROL  rotate left
ROM  read-only memory
ROR  rotate right
R/S  rotate/shift
RTI  return from interrupt
RTS  return from subroutine
R/W  read/write
R/W5R read/write shift register
S    single
SACK selection acknowledge
SBC  subtract carry
SC   single cycle
SE   source effective address
SEC  set carry
SEL  select
SEN  set negative
SEV  set overflow
SEX  sign extend
SEZ  set zero
SI   single instruction
SP   stack pointer
      spare
SR   switch register
SRC  source
SSYN slave sync
ST   start
STPM set trap marker
STR  strobe
SUB  subtract
SVC  service
SWAB swap byte
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>trap address, track address</td>
</tr>
<tr>
<td>TEMP</td>
<td>temporary</td>
</tr>
<tr>
<td>TDR</td>
<td>timing, driver</td>
</tr>
<tr>
<td>TK</td>
<td>teletype keyboard</td>
</tr>
<tr>
<td>TKB</td>
<td>teletype keyboard buffer register</td>
</tr>
<tr>
<td>TKS</td>
<td>teletype keyboard status register</td>
</tr>
<tr>
<td>TP</td>
<td>teletype printer</td>
</tr>
<tr>
<td>TPB</td>
<td>teletype printer buffer</td>
</tr>
<tr>
<td>TPS</td>
<td>teletype printer status register</td>
</tr>
<tr>
<td>TRT</td>
<td>trace trap</td>
</tr>
<tr>
<td>TSC</td>
<td>timing state control</td>
</tr>
<tr>
<td>TSS</td>
<td>timing, selection switch</td>
</tr>
<tr>
<td>TST</td>
<td>test</td>
</tr>
<tr>
<td>UTR</td>
<td>user trap</td>
</tr>
<tr>
<td>VEC</td>
<td>vector</td>
</tr>
<tr>
<td>WC</td>
<td>word count</td>
</tr>
<tr>
<td>WCR</td>
<td>word count register</td>
</tr>
<tr>
<td>XDR</td>
<td>X-line driver</td>
</tr>
<tr>
<td>XRCG</td>
<td>X-line read control group</td>
</tr>
<tr>
<td>XWCG</td>
<td>X-line write control group</td>
</tr>
<tr>
<td>YDR</td>
<td>Y-line driver</td>
</tr>
<tr>
<td>YRCG</td>
<td>Y-line read control group</td>
</tr>
<tr>
<td>YWCG</td>
<td>Y-line write control group</td>
</tr>
</tbody>
</table>
List 2
Definitions

absolute
add carry
address
American Standard Code for Information Interchange
analog-to-digital
arithmetic shift left
arithmetic shift right
automatic send/receive
back space
back space record
bit clear
bit set
bit test
branch
branch if carry clear
branch if carry set
branch if equal
branch if greater or equal
branch if greater than
branch if higher
branch if higher than or same
branch if less or equal
branch if less than
branch if lower or same
branch if minus
branch if not equal

ABS
ADC
ADRS
ASCII
A/D
ASL
ASR
ASP
BIC
BIS
BIT
BR
BCC
BCE
BGT
BHI
BHIS
BLE
BLT
BLOS
BMI
BNE
branch if overflow clear
branch if overflow set
branch if plus
bus address register
bus busy
bus grant
bus register data
bus request
bus shift register
busy
byte
central processor
clear
clear carry
clear negative
clear overflow
clear zero
clock
compare
complement
condition
console
console bus request
console non-processor request
contents
continue
control
control and status register
data

data buffer register

data in

data in, pause

dato out

data out, byte

decoder

decrement

delay

deposit

deposit flag

destination

destination effective address

device address register

device select

Digital Equipment Corporation

digital-to-analog

direct memory access

display X-deflection register

divide

emulator trap

enable

end-of-file

end-of-medium

error

examine

examine flag

execute
extended arithmetic element
external
external reset
first in, last out
flag (when used alone)
flag (when used with signal name)
function
generator
increase
increment
increment flag
indicator
inhibit
initialize
input/output
input/output executive routine
input/output trap
instruction
instruction register
instruction register decoder
instruction shift register
integer divide routine
interrupt
interrupt flag
jump
jump to subroutine
last in, first out
least-significant bit
least significant byte
least-significant digit
line printer
line time clock
line time clock status register
location
master sync
memory
memory address
memory address register
memory buffer register
memory location
memory select
most-significant bit
most-significant byte
most-significant digit
move
negate
negative driver
negative switch
non-processor grant
non-processor request
non-processor request flag
normalize
octal debugging technique
operand
operate
operation
LSBY
LSD
LP
LTC
LKS
LOC
MSYN
MEM
MA
MAP
MBR
HL
MSEL
MSB
MSBY
MSD
MOV
NEG
ND
NS
NPG
NPR
NPRF
NOR
ODT
OPR
OP
OP
operator
paper tape punch
paper tape punch buffer register
paper tape punch status register
paper tape reader
paper tape reader buffer register
paper tape reader status register
paper tape software system
parity available
parity bit
peripheral
positive driver
positive switch
priority transfer
processor
processor status
program
program assembly language
program counter
programmed data processor
punch
read
reader
read-only memory
read/write
read/write shift register
register
release
reset
return from interrupt
return from subroutine
rotate left
rotate right
rotate/shift
selection acknowledge
select
service
set carry
set negative
set overflow
set trap marker
set zero
sign extend
single
single cycle
single instruction
slave sync
source
source effective address
spare
stack pointer
start
strobe
subtract
subtract carry
swap byte
RES
RTI
RTS
ROL
POR
P/S
SACK
SEL
SVC
SEC
SEN
SEV
STPM
SEZ
SEX
S
SC
SI
SSYN
SRC
SE
SP
SP
ST
STR
SUB
SBC
SWAB
<table>
<thead>
<tr>
<th>Term</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>switch register</td>
<td>SR</td>
</tr>
<tr>
<td>teleprinter status register</td>
<td>TPS</td>
</tr>
<tr>
<td>teletype keyboard</td>
<td>TK</td>
</tr>
<tr>
<td>teletype keyboard buffer register</td>
<td>TKB</td>
</tr>
<tr>
<td>teletype keyboard status register</td>
<td>TKS</td>
</tr>
<tr>
<td>teletype printer</td>
<td>TP</td>
</tr>
<tr>
<td>teletype printer buffer</td>
<td>TPB</td>
</tr>
<tr>
<td>temporary</td>
<td>TEMP</td>
</tr>
<tr>
<td>test</td>
<td>TST</td>
</tr>
<tr>
<td>timing, driver</td>
<td>TDR</td>
</tr>
<tr>
<td>timing, selection switch</td>
<td>TSS</td>
</tr>
<tr>
<td>timing state control</td>
<td>TSC</td>
</tr>
<tr>
<td>trace trap</td>
<td>TRT</td>
</tr>
<tr>
<td>track address</td>
<td>TA</td>
</tr>
<tr>
<td>trap address</td>
<td>TA</td>
</tr>
<tr>
<td>user trap</td>
<td>UTR</td>
</tr>
<tr>
<td>vector</td>
<td>VEC</td>
</tr>
<tr>
<td>word count</td>
<td>WC</td>
</tr>
<tr>
<td>word count register</td>
<td>WCR</td>
</tr>
<tr>
<td>X-line driver</td>
<td>XDR</td>
</tr>
<tr>
<td>X-line read control group</td>
<td>XRCG</td>
</tr>
<tr>
<td>X-line write control group</td>
<td>XWCG</td>
</tr>
<tr>
<td>Y-line driver</td>
<td>YDR</td>
</tr>
<tr>
<td>Y-line read control group</td>
<td>YRCG</td>
</tr>
<tr>
<td>Y-line write control group</td>
<td>YWCG</td>
</tr>
<tr>
<td>ASCII Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>ACK</td>
<td>acknowledge</td>
</tr>
<tr>
<td>ATL</td>
<td>alternate (mode)</td>
</tr>
<tr>
<td>BEL</td>
<td>bell</td>
</tr>
<tr>
<td>BS</td>
<td>back space</td>
</tr>
<tr>
<td>CAN</td>
<td>cancel</td>
</tr>
<tr>
<td>CR</td>
<td>carriage return</td>
</tr>
<tr>
<td>DC1</td>
<td>device control 1</td>
</tr>
<tr>
<td>DC2</td>
<td>device control 2</td>
</tr>
<tr>
<td>DC3</td>
<td>device control 3</td>
</tr>
<tr>
<td>DC4</td>
<td>device control 4</td>
</tr>
<tr>
<td>DLE</td>
<td>data link escape</td>
</tr>
<tr>
<td>EM</td>
<td>end-of-medium</td>
</tr>
<tr>
<td>ENQ</td>
<td>enquiry</td>
</tr>
<tr>
<td>EOT</td>
<td>end-of-transmission</td>
</tr>
<tr>
<td>ESC</td>
<td>escape</td>
</tr>
<tr>
<td>ETB</td>
<td>end-of-transmission block</td>
</tr>
<tr>
<td>ETX</td>
<td>end-of-test</td>
</tr>
<tr>
<td>FF</td>
<td>form feed</td>
</tr>
<tr>
<td>FS</td>
<td>file separator</td>
</tr>
<tr>
<td>GS</td>
<td>group separator</td>
</tr>
<tr>
<td>LF</td>
<td>line feed</td>
</tr>
<tr>
<td>NAK</td>
<td>negative acknowledge</td>
</tr>
<tr>
<td>NUL</td>
<td>null</td>
</tr>
<tr>
<td>RS</td>
<td>record separator</td>
</tr>
<tr>
<td>SI</td>
<td>shift in</td>
</tr>
<tr>
<td>SO</td>
<td>shift out</td>
</tr>
<tr>
<td>Code</td>
<td>Meaning</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>SOH</td>
<td>start of header</td>
</tr>
<tr>
<td>STX</td>
<td>start of text</td>
</tr>
<tr>
<td>SUB</td>
<td>substitute</td>
</tr>
<tr>
<td>SYN</td>
<td>synchronous idle</td>
</tr>
<tr>
<td>TAB</td>
<td>tab</td>
</tr>
<tr>
<td>US</td>
<td>unit separator</td>
</tr>
<tr>
<td>VT</td>
<td>vertical tab</td>
</tr>
</tbody>
</table>
PDP-11 GLOSSARY

A

*absolute address A binary number that is permanently assigned as the address of a storage location.

*absolute loader A routine that allows the user to load blocks of code and data from paper tape punched in the absolute binary format.

access time The time interval between the instant at which data is called for (or requested to be stored) from a storage device and the instant delivery (or storage) is started.

accumulator A 16-bit register or memory location in which the result of an operation is formed.

*active release Pertains to the Unibus. Indicates that the bus control is passed from the bus master to the processor by means of an interrupt operation. See "passive release."

address A label, name, or number which designates a location where information is stored.

address field That portion of a computer word either containing the address of the operand or containing the information necessary for calculation of the address.

*address selector The M105 logic module used to decode an address from the processor to select up to four-word or eight-byte external registers.

---

* Although these terms are used in other systems, here their definitions apply solely to the PDP-11.
address map
A table, chart or drawing showing the absolute addresses of all locations in the core memory.

algorithm
A prescribed set of well-defined rules or processes for the solution of a problem in a definite sequence.

alphanumeric
Pertaining to a character set that contains any of the 26 letters and 10 numerals.

analog-to-digital converter
A peripheral device that receives an analog signal and transforms it to an equivalent digital value.

AND gate
A circuit with multiple inputs that provides the desired output only when signals representing assertion are present at all inputs.

arithmetic unit
The component of a computer where arithmetic and logical operations are performed.

argument
An independent variable. For example, in looking up a quantity in a table, the number (or numbers) that identifies the location of the desired value.

ASCII
American Standard Code for Information Interchange. A standard code, using a coded character set consisting of eight-bit coded characters, used for data interchange among data processing communication systems and equipment. The code set includes both graphic and control characters.

assemble
To translate from a symbolic program to a binary program by substituting binary operation codes for symbolic operation codes and absolute or relocatable addresses for
symbolic addresses.
A program that performs the translation from symbolic program to binary program.

asynchronous
Not synchronous. An asynchronous device is one which does not require all elements of that device to be operating in time coincidence.

*autodecrement
An address mode in the PDP-11 system that decrements the contents of a selected register before the register is used. This mode can step the register to the next lower byte (decrement by 1) or word (decrement by 2).

*autoincrement
An address mode in the PDP-11 system that increments the contents of a selected register after the register is used. This mode increments by 1 (byte) or 2 (word).

*autoindex
The process of autoincrementing or autodecrementing the value (+1 or +2) by which an address is autoincremented or autodecremented.

B
background processing
Automatic execution of lower priority computer programs when higher priority programs are not using the system resources.

*bidirectional
Capable of traveling in either direction. Refers to Unibus lines on which signals can be transmitted or received.

binary
Pertaining to a number system with a radix of 2.

binary digit
One of the two states (0 or 1) of the binary system. Usually referred to as a bit.
binary program
A short utility program which, when loaded, instructs the computer to read binary-coded data punched on paper tape and store it in core memory.

bit
A shortened form of binary digit; the smallest unit of information.

block transfer
Moving a large amount of data in one operation. For example: data from a disk into memory or vice versa.

bootstrap
A technique or device designed to bring itself into a desired state by means of its own action. For example, a routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

*bootstrap loader
A program that is loaded into the computer to allow a small set of programs in a special tape format to be loaded into the PDP-11.

boundary
See "word boundaries,"

*branch
A point in a routine where one of two or more choices is made under control of the routine. The PDP-11 has many branch instructions and one unconditional branch instruction.

breakpoint
A location at which execution of a program is stopped to allow operator investigation. A debugging routine inserts a breakpoint to control the running of the program being tested and to return control to the debugging routine after execution of the test program segment.

buffer
A storage device used to compensate for a difference in rate of data flow or time of
buffer register

*bus

*bus address

*bus address register

*bus device

*bus driver

*bus master

*bus receiver

*bus request

*bus slave

*bus transceiver

event occurrence when transmitting data from one device to another.

See "buffer."

See "Unibus."

The current address on the bus may be the address of a device, the processor, or a memory location.

A processor register that holds the address from the processor for display and then loads it onto the Unibus at the required time.

Any external device, including core memory, that is connected to the Unibus and has an assigned device address and/or priority level.

A circuit or module used to pass signals to the Unibus in accordance with the transmission line characteristics of the bus.

The bus device that has control of the Unibus.

A circuit or module used to receive signals from the Unibus. These circuits use gates with high input impedance and proper logic thresholds to ensure that the received signal is compatible with the rest of the system.

A request from a peripheral for control of the bus in order to become bus master and initiate an interrupt or perform a data transfer.

The peripheral that is communicating with the bus master.

A module containing both bus driver and bus receiver.
bus transmitter

byte

c C
call
calling sequence

carry

carry bit

central processor

channel

character

checksum

circuits.

See "bus driver."

A group of binary digits usually operated upon as a unit; half of a word; in the PDP-11, bytes are eight bits long. See also, "high-order byte" and "low-order byte."

to transfer control to a specified routine.

A specified set of instructions and necessary data required to call a given routine.

In performing binary addition, one bit of information often has to be carried from one digit of the addition to the next most significant digit. This operation is referred to as a "carry."

Indicates that an operation resulted in a carry from the most significant bit. During subtraction, indicates a borrow from bit 16.

See "processor."

A path, along which, signals can be transmitted; for example, data channel or output channel. Also refers to a more general path composed of a number of components, for example, communications channel.

A single letter, numeral, or symbol that is used to represent information.

A value representing the sum of all bytes in a program. When the program is loaded,
the sum of the bytes can be compared with the checksum to make sure that the entire program has been loaded correctly.

clear

To erase the contents of a storage location by replacing the contents with zeros; to set register and/or flip-flops in a device to the required initial states.

clock

A device that generates regular periodic signals for synchronization.

coding

To write instructions for a computer using symbols meaningful to the computer or to an assembler, compiler, etc.

command

A control signal, usually written as a character or group of characters, that is used to direct the action of a system program.

compile

To produce a binary-coded program from a program written in source (symbolic) language, by selecting appropriate subroutines from a subroutine library (as directed by the instructions or other symbols in the source program).

compiler

A program that produces a binary-coded program from a source (symbolic) program.

complement

The binary opposite of a number, variable, or function. See "one's complement" and "two's complement."

*condition codes

The four least significant bits of the processor status word. These bits monitor different results of previous operations. The four functions monitored are: zero, negative, carry, and overflow.
conditional branch

A branch that takes place only if a predetermined condition has been met.

conditional jump

A jump that occurs only if specified criteria have been met.

console

An external panel on the computer or peripheral where controls and indicators are available for manual monitoring and operating of the device.

control

A circuit or device used to provide a sequence of levels and/or pulses which cause a system or subsystem to carry out certain procedures.

*control and status register

A register, used with a peripheral, that contains information needed to communicate with the peripheral.

controller

See "dedicated controller."

*core memory

A read/write random access memory using ferrite cores as storage elements. In the PDP-11 system, core memory refers to the MM11-E memory normally used as the basic system memory.

crosstalk

Unwanted insertion of a signal from an adjacent channel.

crowbar

A large power bus normally used to pass excess voltage to ground if an overvoltage condition exists.

data

A general term used to denote any or all facts, numbers, letters, and symbols. It connotes basic elements or information which can be processed or produced by a
data buffer register

A register used with a peripheral to temporarily store data that is to be transferred into or out of the processor or other device.

*data paths

That portion of the KA11 processor where normal processing and computation occurs. All modifications and routing of data within the processor are performed by the data paths which consist primarily of the input gating and latches, adder, and output gating circuits.

debug

To detect, locate, and remove mistakes from a program or malfunctions from a computer.

debugging program

An independent, self-contained service program which allows the programmer to communicate with the object program in order to make modifications, additions, and deletions.

tester

A logic device capable of converting from one numbering system to another (such as an octal-to-decimal decoder) or designed to interrogate certain bits from an input word to supply specific information such as an address or operation code.

dedicated controller

A processor or computer system, usually with a read-only memory, that is designed and/or used to control only one specific process. For example, a computer designed to continually monitor, evaluate,
and change a chemical process.

A signal path used for only one purpose.

Indirectly addressed. The contents of the location is the address of the operand rather than the operand itself.

A character that separates and organizes elements of data.

A PDP-11 major state that retrieves destination data from internal or external storage. All necessary address calculations for obtaining the destination data are performed at this time.

Usually refers to an external device which is synonymous with the term "peripheral."

A bit in either the interface logic or the device itself that is set to indicate a specific condition such as ready or busy.

Part of an address that is used to specify that a particular device has been selected for use.

Transfer of data without supervision of the processor. Data is passed directly from one device to another through the Unibus.

Pertaining to the detection and isolation of a malfunction or mistake; usually used in the form "diagnostic programming."

A character used to represent one of the non-negative integers smaller than the radix. For example, in binary notation (radix 2), a digit is either 1 or 0.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>digital-to-analog converter</td>
<td>A peripheral device that receives a digital value and transforms it to an equivalent analog signal.</td>
</tr>
<tr>
<td>*direct address</td>
<td>An address that specifies the location of an instruction operand.</td>
</tr>
<tr>
<td>*direct address mode</td>
<td>Any PDP-11 address mode that is not deferred.</td>
</tr>
<tr>
<td>*direct memory access</td>
<td>Transfer of data into memory without supervision of the processor. Data is passed directly between the core memory and another device through the Unibus. Transfers are usually accomplished with a non-processor request.</td>
</tr>
<tr>
<td>disable</td>
<td>To render inoperative or to prevent from being used. Normally used with reference to hardware as opposed to &quot;inhibit&quot; which normally refers to signals.</td>
</tr>
<tr>
<td>disk</td>
<td>A mass-storage device. Basic unit is a disk on which data is magnetically recorded. Data can be accessed randomly, and access time is faster than with magnetic tape because required search time is significantly less. Most disks can store considerably greater amounts of data than core memories.</td>
</tr>
<tr>
<td>display</td>
<td>A peripheral device used to portray data graphically. Normally refers to some type of cathode-ray tube system.</td>
</tr>
<tr>
<td>*double-operand</td>
<td>PDP-11 instructions that contain two address fields: one field for the source operand, and one field for the destination operand; two-address instruction.</td>
</tr>
<tr>
<td>double-precision</td>
<td>Pertaining to the use of two computer words to represent one number.</td>
</tr>
</tbody>
</table>
downtime

The time interval during which a device or system is inoperative.

driver

See "bus driver." Also refers to a software routine designed to interface directly to a device. For example, a disk driver.

dump

To copy the contents of all or part of the core memory, usually on to some external storage medium such as hard copy or paper tape.

*dynamic master-slave relationship

Indicates that control of the Unibus may be passed from a master to another peripheral which then becomes master. It is not necessary to first pass control back to the processor. See "master-slave."

E

edit

To arrange and/or alter information for machine input or output.

editor

A program that allows the user to produce edit symbolic files on line.

*effective address

The address actually used in the execution of a computer instruction.

emulator

A hardware device that permits a program written for a specific computer to be run on a different type of computer system.

*emulator trap

A PDP-11 instruction that calls an emulator routine.

enable

To set up conditions so that a specific device, circuit, or signal can be used. The opposite of inhibit.

end-around carry

The action of adding the most
execute

*execute major state

executive routine

exit

explicit address

*external device

*external page

F

fanout

fetch

*fetch major state

file

significant bit of a binary number to the least significant bit position.

To carry out a specified instruction or to run a program on the computer.

A PDP-11 major state during which the specified instruction is performed.

A routine that controls or monitors execution of other routines.

To leave. Specifically, to leave a main program in order to enter a subroutine or vice versa.

See "absolute address."

Peripheral. In the PDP-11 system, any device connected to the Unibus with the exception of the processor.

Addresses above 160000 which are reserved for device register and processor register addresses.

A number indicating the number of unit loads a specific output signal can drive.

The act of obtaining and decoding an instruction from the program.

A PDP-11 major state during which the next program instruction is obtained and decoded, and a determination made as to what major state to enter next, based on the type of instruction decoded.

A collection of related records treated as a unit.
fixed point
The position of the radix point in a number system is constant according to a predetermined convention.

flag
A character that signals the occurrence of some condition, such as the end of a word.

flip-flop
A basic computer circuit or device capable of assuming one and only one of two stable states.

floating point
A number system in which the position of the radix point is indicated by one part (the exponent) and another part represents the most significant digits (the fraction).

flowchart
A graphical representation of the sequence of instructions required to carry out a data processing operation.

foreground processing
The automatic execution of high priority programs that have been designed to preempt the use of the computing facilities.

format
The arrangement of data.

G

*general register
One of eight 16-bit internal registers in the PDP-11 processor. These are used for temporary storage, as accumulators, as index registers, as stack pointers, and other general-purpose functions.

H

hard copy
Information stored on a permanent medium that is readable, such as a printout.
from a line printer or teletype printer.

Physical equipment such as mechanical, electrical, or electronic devices.

A component that reads, records, or erases data on a storage device. Often referred to as a recording head or magnetic head.

The most significant byte in a word; in the PDP-11, indicates the byte occupying bit positions 8 through 15 of a word. The high-order byte is always an odd address.

An address mode that includes the operand as part of the instruction. The operand is the word immediately following the first word in a two- or three-word instruction in the program. This mode is actually the autoincrement mode used in conjunction with the program counter.

An address mode that uses data in a general register as a base for address calculations to permit random access to items in tables or stacks of data.

An address in a computer instruction that indicates a location where the address of the referenced operand is to be found. See "deferred."

To prevent. Normally used with signals rather than hardware to indicate that the signal is prevented from occurring. Also used with memory. For example, the inhibit signal prevents the
initialize

core from changing state.

To set counters, switches, and addresses to zero or other starting values at the beginning of, or at prescribed points in, a computer program.

input

The transferring of data from auxiliary or external storage of the computer.

*instruction register

An internal register in the KA11 processor that stores the instruction fetched from memory so that portions can be decoded as needed during subsequent time states.

interface

The hardware needed to allow communication between the system Unibus and the peripheral.

*interlocked

The interrelation of communication between the Unibus master and slave devices. This relation is such that for each control signal from the master, the slave must send a response before the operation continues.

internal storage

The storage facilities forming an integral physical part of the computer and directly controlled by the processor.

*interrupt

A temporary disruption of normal operation by a special signal from the computer or peripheral.

*interrupt control

The M782 logic module which contains necessary logic circuits to allow a peripheral device to gain control of the Unibus and perform a program interrupt.

*interrupt vector

Two locations containing processor status word and the program counter value which indicates the starting point
of the interrupt routine.
See "peripheral" or "external device."

A departure from the normal sequence of executing instructions in a program. An unconditional jump causes the program to go to the specified location in the program; a conditional jump causes the program to go to the new location only if preestablished criteria have been met.

A set of representations, conventions, and rules used to convey information.

A storage/retrieval method in which the last item stored is the first item retrieved.

A circuit that locks data into the processor input gates so that output states are maintained even when the input signals are removed. The latches are functionally part of the "data paths."

The time delay involved while waiting for specified data to reach a desired point or while waiting for a specified response which must arrive prior to further processing.

The blank section at the beginning of a magnetic or paper tape. In certain cases (such as the absolute loader) the loader is punched in a special format.
least-significant bit
The rightmost bit in a byte or word.

least-significant digit
The rightmost digit of a number.

level
A voltage that remains constant for a long time. There are two possible levels: low or high.

list
Usually refers to related data that occupies successive storage locations. In the PDP-11, the main distinction between a list and a stack is that a stack is automatically maintained by the processor and a list is not.

*literal
Used in programming to indicate that the value in the program is the actual value to be used by the computer. Opposite of "symbolic."

load
To place data into storage.

location
A place in storage or memory where a unit of data or an instruction can be stored.

loop
A sequence of instructions that is executed repeatedly until a termination condition exists.

*low-order byte
The least-significant byte in a word; in the PDP-11, indicates the byte occupying bit positions 0 through 7. The low-order byte is always an even address.

M

machine language
The actual language used by the computer in performing operations; usually refers to either binary or octal codes; also often used to refer to assembler language coding.
**machine language programming**
Writing a program in binary or octal notation, or converting from a symbolic program to a binary program.

**macro instruction**
An instruction in a source language that is equivalent to a specified sequence of assembler instructions.

**main frame**
See "processor."

**major state**
A computer timing cycle. In the PDP-11 system, there are five major states: fetch, source, destination, execute, and service. Not all major states are entered for each instruction.

**manual input**
The entry of data by hand into a device at the time of processing.

**manual operation**
The processing of data in a system by direct manual techniques.

**mask**
A pattern of bits that is used to control the retention or elimination of portions of another pattern of bits; a filter.

**mass storage device**
A bulk storage device, such as a disk.

**master/slave**
The relationship between two devices communicating through the Unibus. The controlling device is master, the responding device is the slave.

**memory**
The storage in the system pertaining to a device in which data can be retrieved. See also "core memory," "read-only memory," and "wordlet memory."

**memory address**
Usually refers to the address in external core memory which is being used at the time for reading or writing.
Mnemonic Symbol

A symbol chosen to assist the human memory, a memory aid. For example, the abbreviation MPY for the word multiply.

Most Significant Bit

The leftmost bit in a byte or word.

Most Significant Digit

The leftmost digit of a number.

Mounting Box

The cabinet used to house the basic KA11 processor, core memory, and other logic circuits. The operator’s console is attached to the front of the box. Other mounting boxes may be used for additional logic cards or memory and normally have a blank front panel. Sometimes "drawer" is used instead of "mounting box" but the latter is the preferred term.

N

Negate

A process of converting the value of a binary function or variable to the equivalent two's complement number.

Nested Interrupt Servicing

An operation by which servicing of an interrupt for a device can be interrupted in order to service a higher priority device. Upon completion, servicing of the lower priority device is automatically resumed. This operation is not limited to two devices; therefore, an interrupt can be interrupted by another device which in turn is interrupted.

Nesting

Including a routine or block of data within another routine or block of data. In the PDP-11, more specifically refers to interrupting of a routine by a subroutine that in turn is interrupted by
another subroutine, etc. The processor keeps track of the data so that as each subroutine is completed, the next one is continued. Also refers to calling a subroutine from a subroutine.

Refers to data transfers between any two peripheral devices, including memory, without supervision of the processor. The two devices use the Unibus during data transfers which are accomplished between Unibus cycles.

0

Object program

The binary-coded program which is the output after translation from the source language; the binary program that runs on the computer.

Octal

A number system with a radix of 8 which is used as a shorthand notation of a binary number.

Offline

Pertaining to equipment or devices not under direct control of the computer.

#offset

A two-digit octal number in an instruction that is multiplied by two and added to the program counter to indicate the location of the next instruction. The offset is normally used only in branch instructions.

One's complement

The binary number obtained by complementing all bits of another binary number. Used as the first step in complement arithmetic so that binary subtraction can be performed by using addition techniques. The second step is to increment the one's
complement which then provides the two's complement number.

online

Pertaining to equipment or devices under direct control of the computer; also pertains to programs operating directly and immediately to user commands.

operand

That portion of an instruction code which is affected, manipulated, or operated upon.

operator

That which indicates the action to be performed on the operand.

OR gate

A circuit with multiple inputs that provides the desired output when a signal representing assertion is present at any input.

origin

The absolute address of the beginning of a program or of a unique area of code or data.

output

Information transferred from the internal storage of a computer to output devices or external storage.

overflow

Generation of a quantity beyond the capacity of the arithmetic or storage facility.

P

parity

A method for checking the correctness of binary characters. An extra bit (called parity bit or p) is added to numbers in systems using parity. If even parity is used, the sum of all 1's in a number including the parity bit is even; if odd parity is used, the sum is odd.

#passive release

Pertains to the Unibus. Indicates that the bus master
releases the bus by dropping the bus busy signal. See "active release".

Any unit of equipment, outside of the processor or Unibus, that provides the system with communication, storage, and/or service. Also called "external device" or "I/O device".

A core memory location containing the actual (effective) address of the desired data; in the PDP-11, pointer often refers to the register containing the pointer address. See "stack pointer".

See "pointer".

A centrally controlled method of calling a number of devices to permit them to transmit information; interrogation of peripherals one at a time to determine which peripheral desires service, not used in the PDP-11 system.

To remove a word from the top of a pushdown/popup list.

See "relocatable".

Logic circuits that protect an operating program in the event computer primary power fails. The circuits automatically store current operating parameters of the program as well as indicators of a power failure. When power is returned, the processor automatically makes use of this information to continue the program.

A named process consisting of one or more operations or program steps that are specified elsewhere in the program.
*priority arbitration

A method used by the processor to compare its own priority with priorities from devices requesting the bus in order to determine which device, if any, is granted control of the Unibus.

*priority interrupt

Refers to the four-level priority interrupt system employed by the PDP-11 system.

*priority transfer

The signal sequence by which a device is selected as next bus master. No actual bus transformer is performed, only selection of the next bus master.

procedure

The course of action taken for the solution of a problem or performance of a specified operation, a portion of an algorithm translated into machine code.

*processor

A unit of a computing system that includes the circuits controlling the interpretation and execution of instructions. The processor does not include the Unibus, core memory, interface, or peripheral devices. The term "main frame" is sometimes used but this term refers to all components (processor, memory, power supply) in the basic mounting box.

*processor status word

An addressable register in the external page indicating the current priority of the processor and the results of the previous operation as indicated by the condition code bits.

program

The complete sequence of instructions and routines necessary to solve a problem or perform a specified action.

*program counter

A general-purpose register (number 7) that contains the
program library

propagation delay

pulse

pulse width

punched paper tape

*push

*pushdown list

R

radix

random access

read

address of the next word to be fetched.

A collection of available computer programs and routines in a specific format.

The time required to transfer information from the input to the output of an electronic device.

A voltage that goes from one level to another, remains there for a short time, and then returns to the original level.

The length of time a pulse voltage is at the second, or transient level.

A paper tape containing a pattern of holes used to represent data; a tape used to feed in or receive information from a computer system.

To place a word on the top of a pushdown/ponup list.

A list that is constructed and maintained on a "last in, first out" basis.

The base of a number system; i.e., the quantity of characters that can be used in each digital position of the number system.

Unordered access, usually used to describe unordered access to data or a device.

To transfer information from an input device to internal storage; also refers to the internal acquisition of data from core memory or other
*read-only memory

A random access memory that contains components which permanently assume a specific state so that data can be read from memory but cannot be erased, changed, or added.

*real time

Any data manipulation, calculation, or control operation that is performed during the monitored task rather than after completion. For example, a satellite control system is a real time system as opposed to a system that analyzes data some period of time after all of the data has been recorded.

*receiver

See "bus receiver".

*recursive

A code or program that permits a subroutine to be called and then, if desired, call itself during operation; a closed subroutine that calls itself.

*reentrant

Pure code which can be interrupted and started again without error. The interrupt service may use the interrupt routine. Upon completion of the interrupt, the routine continues from the interrupt point.

*register

A device capable of storing a specified amount of data, such as one word; usually refers to a flip-flop storage device or core memory location.

*register mode

A PDP-11 address mode in which the operand is contained in one of the eight general-purpose registers.

*relative address

The number that specifies the difference between the absolute address and the base address. Also, the address formed by the sum of the base and the displacement.
**relative mode**

A PDP-11 address mode that specifies the operand address relative to the program counter to permit relocatable addresses. This mode is a combination of the index mode used in conjunction with the program counter.

**relocate**

To move a routine from one portion of storage to another and to adjust the necessary address references so that the routine can be correctly executed at its new location.

See "relocate".

**relocatable**

Communication with a computer by one or more stations that are distant from the computing facility.

**remote access**

Instructions that have an opcode which has no defined function in systems using the processor. If any of these instructions are used, a trap occurs.

**reserved instruction**

The time which elapses between generation of an inquiry at a device and receipt of a response at the device.

**response time**

To return to its original condition. Normally refers to a memory restore cycle. Since the contents of a memory location are destroyed when read, they must be restored after each read cycle. This is accomplished automatically.

**restore**

A set of instructions, arranged in the proper sequence, needed to cause the computer to perform a desired task.

**routine**

A single, continuous execution of a program.
scratch pad memory

Any memory used for temporary storage. Usually refers to internal storage registers that hold partial results or operands until needed to complete a calculation.

*service

In the PDP-11, refers to a major state during which extra operations are performed; in general, refers to servicing an external device that desires to communicate with the computer.

service routine

A program used for general support of the user. For example, I/O routines, diagnostics, and other utility routines.

shift register

A register in which all information stored in the register is shifted one bit position to the right or left according to a specified instruction or action.

sign bit

When using complementary arithmetic, the bit directly to the left of the number indicates the sign (+ or -) of the number. A 1 indicates negative numbers, an 0 indicates positive numbers.

*single-operand

PDP-11 instructions that contain only one address field, that of the destination operand.

software

The collection of programs, procedures, rules, and related documentation associated with operation of a specific computer. For example, compilers, editors, utility programs, and related documentation and run procedures.

*source address

The address of the first operand in a two-address instruction (double-operand instruction).
source language

A symbolic language that is an input to a given translation process.

*sourse major state

A PDP-11 major state that retrieves source data from internal or external storage. All necessary address calculations for obtaining the source data are performed at this time.

*stack

A dynamic, sequential list of data with special provision for access from one end. Storage and retrieval from stacks is called "pushing" and "popping" respectively. In the PDP-11, the stack is automatically maintained by the hardware.

*stack overflow

A condition that indicates a push onto the processor stack below absolute address 400.

*stack pointer

The element used to indicate the top item on a stack. In the PDP-11, general register 6 serves as a stack pointer. The contents of this register is the address of the first (bottom) word on the hardware stack.

statement

A meaningful expression or generalized instruction in a source language.

*status register

A 16-bit register in which the high-order byte is unused and the low-order byte stores the processor status word; a register storing the external device status word.

*status word

See "processor status word".

step

One operation in a routine.

store

To enter data into a device where it can be held and can be retrieved.

string

A connected sequence of
entities such as characters in a command string.

subroutine
A small routine, usually performing only one task, that is called frequently from various points of the main routine.

subroutine, closed
A subroutine not stored in the main part of a program. Such a subroutine is entered by a jump or branch operation, and provision is made at the end of the subroutine to return control to the calling program.

subroutine, open
A subroutine that must be inserted into a program at each place it is to be used.

switch register
An 18-bit register composed of manually operated switches that are used to load either addresses or data into the PDP-11 system. The switch register is on the front of the processor console.

symbolic address
A set of characters used to specify a memory location within a program.

symbolic coding
Writing instructions using mnemonic notation instead of actual machine language (binary) notation.

symbolic language programming
Writing program instructions in a language which facilitates the translation of programs into binary code by making use of mnemonic conventions.

symbolic program
A service program that translates symbolic programs into binary-coded programs. The programmer writes the symbolic program using symbols which are meaningful to him and the symbolic program translates the symbols into binary code which is
synchronize

To ensure that a level or pulse is presented to a system or component at the correct time.

synchronous

All changes occurring simultaneously or in a definite, timed sequence.

*system unit

A mounting unit composed of three 8-slot connector blocks used to mount logic modules. System units are the basic building blocks of the PDP-11 system.

T

*T bit

A bit in the processor status word used in program debugging. This bit can be set or cleared under program control. If set, a processor trap occurs upon completion of the instruction.

table

A collection of data in which each item is uniquely identified by its position relative to the other items, or by some other means.

tag

One or more characters attached to an item or record for the purpose of identification.

terminal

A device in a system through which data can either enter or leave.

*time out

A specified amount of time (10 microseconds) that the system waits for a response from a referenced address, if there is no response within the specified time, an error occurs. Time-out errors are caused. In general, by attempts to reference nonexistent memory or
time sharing

A method of allocating processor time and other computer services among multiple users so that the computer, in appearance, processes a number of programs simultaneously.

toggle

To use console switches for entering data into the computer internal storage or memory, to cause alternation of states as in toggling a flip-flop.

track address

The part of a mass storage device which is the beginning of a specific block of data.

trailer

Identical to "leader" except it is at the end, rather than the beginning of a tape.

*transceiver

See "bus transceiver."

translate

To convert from one language to another.

*trap

An unprogrammed jump to a known location, automatically activated by the hardware if certain predetermined conditions occur, such as illegal instructions, errors, etc.

turnkey

A computer console containing only one control, usually a power switch, that can be turned on or off only with a key.

U

*Unibus

The single, high-speed bus structure shared by the KAll Processor, core memory, and all peripherals.

*unidirectional

Capable of traveling in only
unit load

One direction, refers to the Unibus control transfer lines that carry signals to select the next bus master.

All inputs impose a load on the outputs driving them. A TTL unit load requires 1.6 ma at ground and +40 uA at +3 volts. The load imposed upon an output by an input can be defined as a number of unit loads.

V

vector

Two words, containing the value of the program counter and processor status word, respectively, that direct the processor to a new routine.

vector address

The address of the location containing the vector words.

W

wait loop

A condition caused by the program WAIT instruction to allow the processor to wait for an interrupt. When the processor is in a wait loop, it does not compete for bus control by fetching instructions or operands from memory.

word

A 16-bit unit of data in the PDP-11 that is stored in two successive locations. The word address is always an even address.

word boundary

The division between even numbered addresses. Since each word occupies two storage locations, words can be addressed only on even boundaries; bytes can be addressed on either even or odd boundaries.
word count

word length

wordlet memory

write

The number of words in the block of data to be transferred.

The number of bits in a word.

A small read/write memory used with the read-only memory. The wordlet memory (MW11-A) is used primarily for temporary data and instruction storage.

To transfer information from internal storage to an output device or external storage.
Summary:

Information printed out by the system when initializing (i.e., the value of the PC and stack pointer) are octal addresses. In order for the user of the system to be able to better communicate with the machine and with other computer personnel, a solid working knowledge of octal and binary arithmetic is essential. Remember that almost all numerical information passing between the man-machine interface is octal data. Practice will make this numbering system as convenient as decimal.