1196

1/80

PWB/UNIX Operations Manual (Second Edition)

A. G. Petruccelli

Bell Laboratories
Piscataway, New Jersey 08854

ABSTRACT

This manual contains a complete description of PDP® 11 console operations, step-by-step instructions for normal operator functions, as well as descriptions of the PWB/UNIX system console error messages.

The information in this manual was gathered from personal experience, the *PWB/UNIX User's Manual*, Digital Equipment Corporation (DEC®) hardware manuals, and technical memorandums contained in *Documents for PWB/UNIX*.

Because this manual is intended to be as general as possible, it is suggested that each location add specific information about:

- Hardware configuration.
- Telephone line configuration.
- Specific logging and record-keeping practices.
- Contacts for hardware and software problems.
- Site-dependent diagnostic procedures.

This manual is partially based on, and supercedes the *PWB/UNIX Operations Manual* by M. E. Pearlman.

CONTENTS

LIST OF FIGURES

PWB/UNIX Operations Manual (Second Edition)

A. G. Petruccelli

Bell Laboratories
Piscataway, New Jersey 08854

ABSTRACT

This manual contains a complete description of PDP® 11 console operations, step-by-step instructions for normal operator functions, as well as descriptions of the PWB/UNIX system console error messages.

The information in this manual was gathered from personal experience, the *PWB/UNIX User's Manual*, Digital Equipment Corporation (DEC®) hardware manuals, and technical memorandums contained in *Documents for PWB/UNIX*.

Because this manual is intended to be as general as possible, it is suggested that each location add specific information about:

- Hardware configuration.
- Telephone line configuration.
- Specific logging and record-keeping practices.
- Contacts for hardware and software problems.
- Site-dependent diagnostic procedures.

This manual is partially based on, and supercedes the *PWB/UNIX Operations Manual* by M. E. Pearlman.

HARDWARE OPERATIONS — PDP 11/45, 11/70

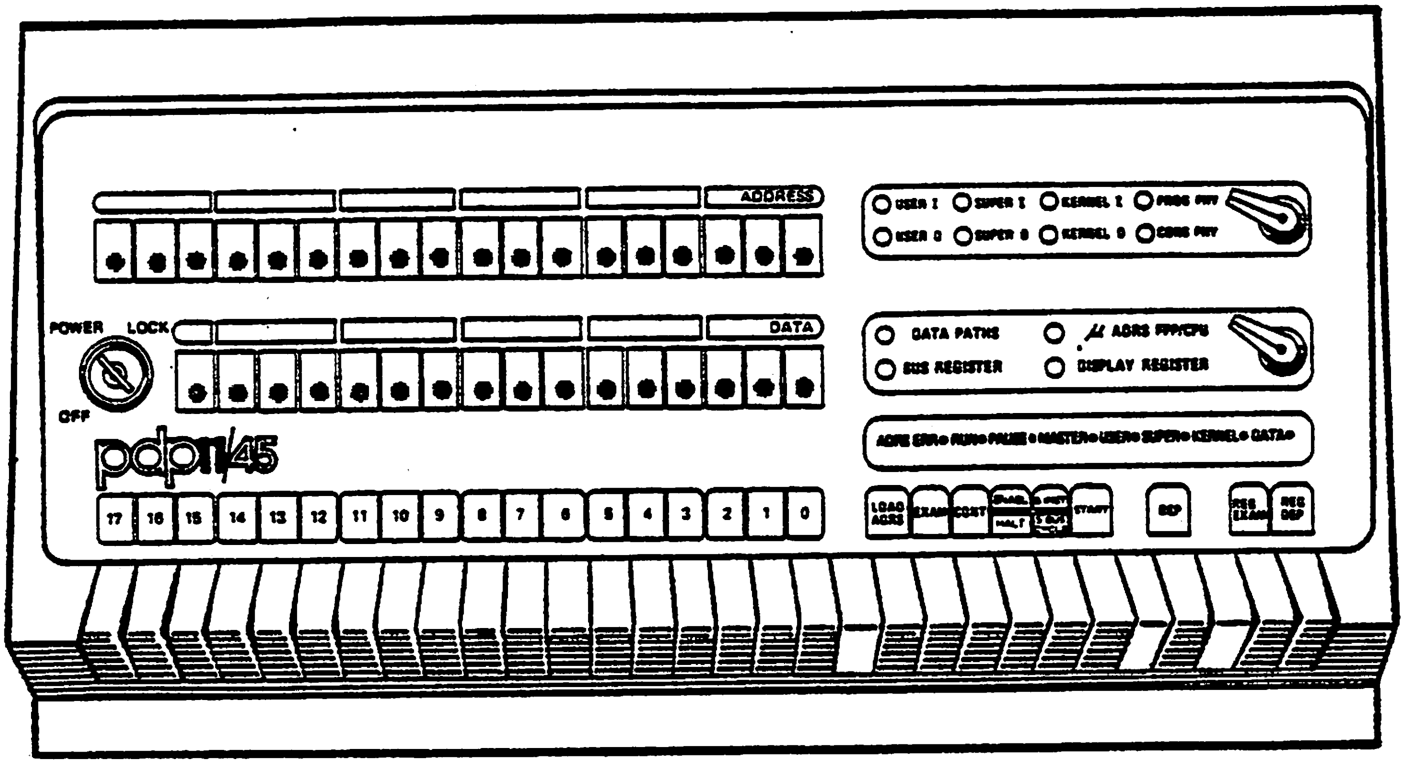


Figure 1. PDP 11/45 CONSOLE

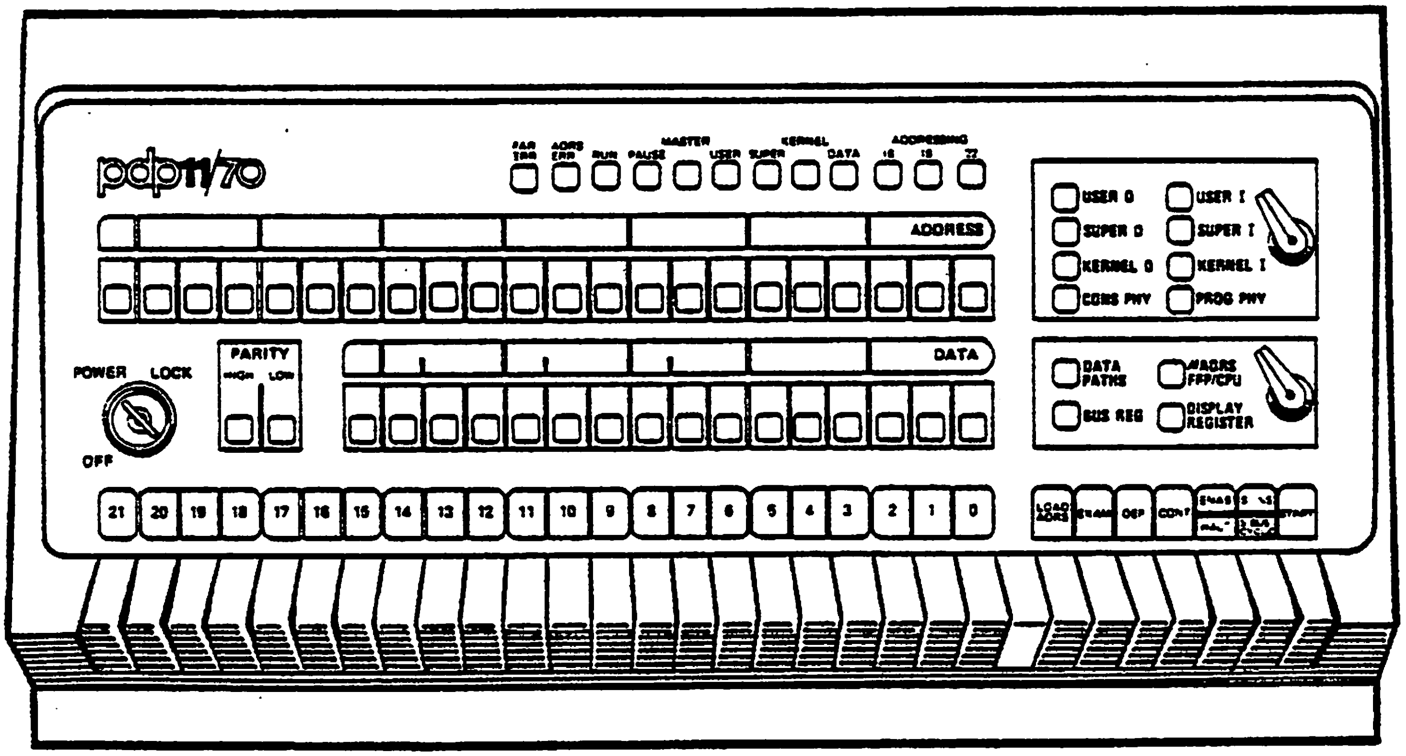


Figure 2. PDP 11/70 CONSOLE

INTRODUCTION

The following documentation is primarily intended to describe the PDP* 11/70 console and its operation. Differences in the PDP 11/45 appear within brackets "[]". Those cases that are applicable to only one of the two systems are clearly labeled as such.

CONSOLE DESCRIPTION

The console is composed of the following:

1. Power Key Switch (OFF/POWER/LOCK).
2. ADDRESS Register — 22-bit [18-bit]* Display.
3. DATA Register — 16-bit Display.
4. PARITY bit HIGH byte & LOW byte Indicator Lights (11/70 only).
5. Switch Register — 22 [18] switches.
6. Error Lights.

   ADRS ERR (Address Error)
   PAR ERR (Parity Error, 11/70 only)
7. Processor State Lights (7 indicators).

   RUN
   PAUSE
   MASTER
   USER
   SUPER
   KERNEL
   DATA
8. Mapping Lights (11/70 only).

   16 BIT
   18 BIT
   22 BIT
9. ADDRESS Display Select Switch (8 positions).

   USER I (Virtual)
   USER D (Virtual)
   SUPER I (Virtual)
   SUPER D (Virtual)
   KERNEL I (Virtual)
   KERNEL D (Virtual)
   PROG PHY (Program Physical)
   CONS PHY (Console Physical)
10. DATA Display Select Switch (4 positions)

    DATA PATHS
    BUS REGISTER
    μ ADRS FPP/CPU (Micro-program Addresses)
    DISPLAY REGISTER

* Differences in the PDP 11/45 appear within brackets "[]".

11. Lamp Test Switch.
12. Control Switches.

    LOAD ADRS (Load Address)
    EXAM (examine)
    DEP (deposit)
    CONT (continue)
    HALT/ENABLE [ENABL]
    S INST/S BUS CYCLE (single instruction/single bus cycle)
    START
    REG EXAM (Register Examine, 11/45 only)
    REG DEP (Register Deposit, 11/45 only)

CONSOLE OPERATION – LAMP TEST SWITCH

The Lamp Test Switch is an unlabeled, white switch located between the Switch Register and the LOAD ADRS Switch. When the Lamp Test Switch is raised, all console indicator lights should go on. An indicator which does not light is defective and should be replaced.

CONSOLE OPERATION – POWER KEY

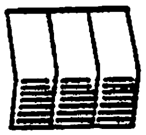
The Power Key controls power to the CPU** and has three positions:

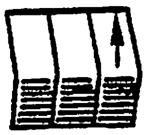
OFF	Power to the processor is OFF.
POWER	Power to the processor is ON, and all console switches function normally. This is the *normal* position while PWB/UNIX is running.
LOCK	Power to the processor is ON, but the 7 control switches LOAD ADRS through START are disabled. All other switches are functional.

CONSOLE OPERATION – SWITCH REGISTER

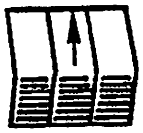
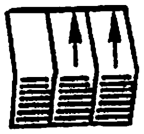
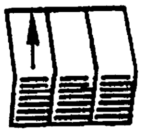
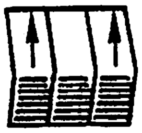
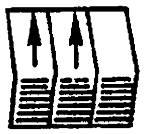
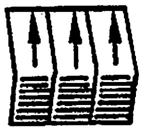
The Switch Register consists of 22 [18] switches labeled 0 through 21 [17] from right to left (numbers correspond to bit positions). They are used to manually enter both addresses and data into the processor.

To enter an address such as 165000_8, the switches must be divided into groups of three, starting from the right. Bits 0-2 in the first group, bits 3-5 in the second, 6-8 in the third, 9-11 in the fourth, 12-14 in the fifth, etc. Each group of 3 switches is used to indicate an octal digit; thus, a number can be represented on the switches as follows:

zero  All 3 switches down.

one  Right-most switch up.

** Central Processing Unit

two		Middle switch up.
three		Middle and right switches up.
four		Left-most switch up.
five		Left and right switches up.
six		Left and middle switches up.
seven		All 3 switches up.

The arrows in Figures 3 and 4 depict which switches should be up to enter the address 165000_8 on the 11/70 and address 173020_8 on the 11/45 respectively.

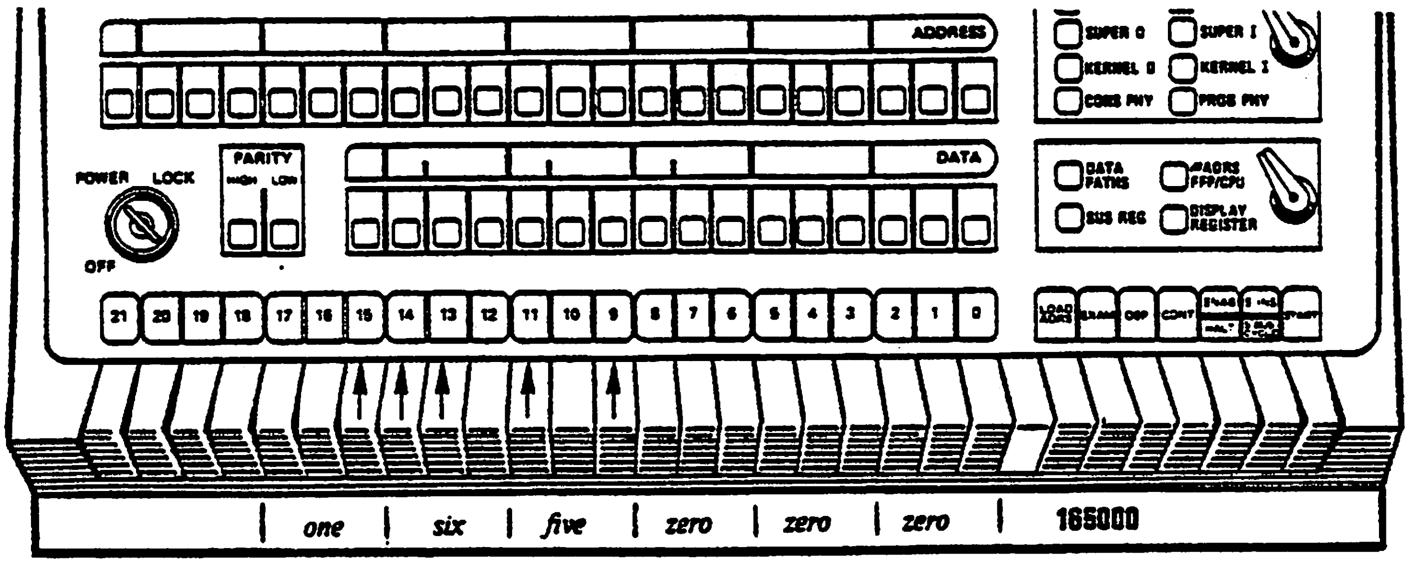


Figure 3. Address 165000_8 on the PDP 11/70.

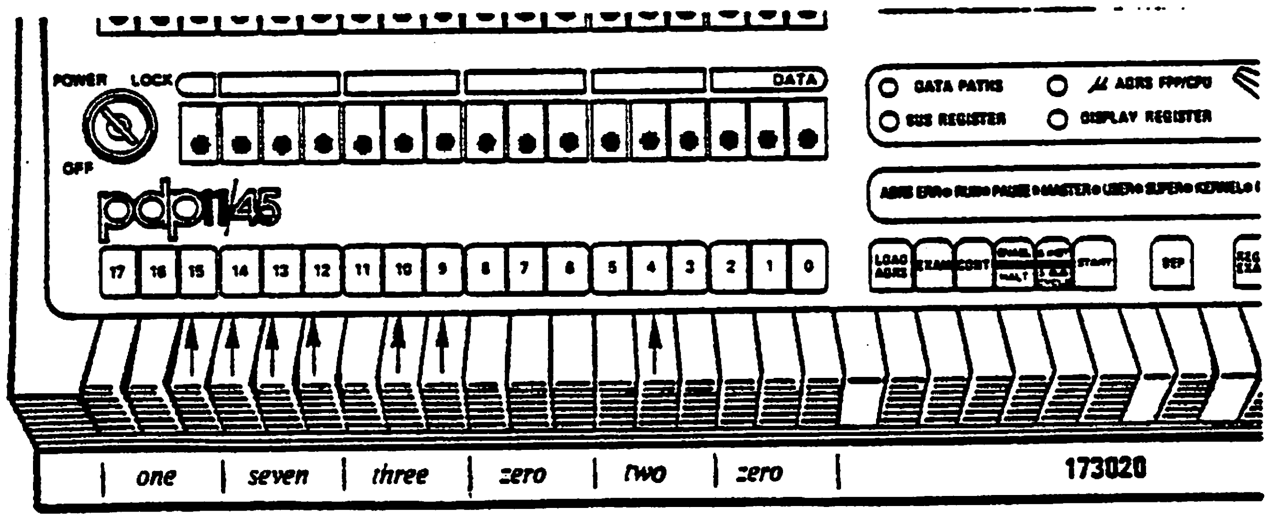


Figure 4. Address 173020_8 on the PDP 11/45.

CONSOLE OPERATION – CONTROL SWITCH FUNCTIONS

LOAD ADRS (Load Address).

When the LOAD ADRS Switch is depressed, the contents of the Switch Register are loaded into the Address Display. The address displayed in the Address Display Lights depends on the position of the Address Select Switch.

EXAM (Examine).

Depressing the EXAM Switch causes the contents of the current location, specified in the Address Display, to be displayed in the DATA Display Register when the Data Select Switch is in the DATA PATHS position.†

† The address in the Address Display will be mapped or unmapped depending on the position of the Address Select Switch. The location shown in the Address Display Lights is also a function of that switch. See Section 11.4 of [1] for more information.

DEP (Deposit).
Raising the **DEP** Switch causes the current contents of the Switch Register to be deposited into the address specified by the current contents of the Address Display.

CONT (Continue).
Depressing the **CONT** Switch causes the CPU to resume execution. The **CONT** Switch has no effect when the CPU is in **RUN** state.

HALT/ENABLE [ENABL].
The **HALT/ENABLE [ENABL]** Switch is a two position switch used to stop machine execution or to enable the system to run.

S INST/S BUS CYCLE (Single Instruction/Single Bus Cycle).
This switch affects only the operation of the **CONT** Switch. It controls whether the machine stops after instructions or bus cycles.‡ The position of this switch has no effect unless the **HALT/ENABLE [ENABL]** Switch is in the **HALT** position. It is used chiefly for debugging. See Section 11.7 of [1] for more information.

START.
The functions of the **START** Switch depend upon the setting of the **HALT/ENABLE [ENABL]** Switch. If the CPU is in the **HALT** position, the processor is reset. If in the **ENABLE [ENABL]** position, execution is started unless it is already in the **RUN** state.

REG EXAM (Register Examine, 11/45 only).
Depressing the **REG EXAM** Switch causes the contents of the General Purpose Register specified by the low order five bits of the Bus Address Register to be displayed in the Data Display Register. See Section 9.6.8 of [2] for interpretation of these contents.

REG DEP (Register Deposit, 11/45 only).
Raising the **REG DEP** Switch causes the contents of the Switch Register to be deposited into the General Purpose Register specified by the current contents of the CPU Bus Address Register. The CPU Bus Address Register should have been previously loaded by a **LOAD ADRS** operation according to the Switch Register settings described in **REG EXAM** above.

CONSOLE OPERATION – ADDRESS SELECT KNOB

The Address Select Knob has 8 positions for observing the address of data being examined or deposited. These positions reference virtual or physical memory as described below:

VIRTUAL	The six positions: **USER I, USER D, SUPER I, SUPER D, KERNEL I,** and **KERNEL D** indicate the current address as a 16-bit Virtual address when the Memory Management Unit is turned on (i.e. UNIX is running), otherwise it indicates the true 16-bit Physical Address.* These positions are generally used for debugging.
PROG PHY	This position displays the 22-bit [18-bit] Physical Address of the current bus cycle that was generated by the Memory Management Unit. This address is generally used for debugging.

‡ The *bus* (or UNIBUS®) is the primary control and communications path connecting most of the PDP 11 system's components and peripherals.

* These positions make it convenient to examine and change programs which are subject to relocation, without requiring any knowledge of where they have actually been relocated in physical memory. See Section 9.6.8, page 9-21 of [2] for more details. See Section 6.4 of [1] or Chapter 10 of [2] for more information on the Memory Management Unit.

CONS PHY	This position displays a 22-bit [16-bit] Physical Address to be used for console operations such as LOAD ADRS, EXAM, and DEP. This is the *normal* position while PWB/UNIX is running.

CONSOLE OPERATION — DATA SELECT KNOB

The contents of the 16-bit Data Display Register are controlled by the following positions of the Data Select knob:

DATA PATHS	The *normal* display mode. This position enables examined or deposited data to be shown in the Data Display.
BUS REG	The internal CPU register used for bus cycles.
μADRS FPP/CPU	The ROM** address, FPP† control micro-program (bits 15 to 8) and the CPU control micro-program (bits 7 to 0).
DISPLAY REGISTER	The contents of the Display Register. This has an address of 17 777 570_8.

CONSOLE OPERATION — STATUS INDICATOR LIGHTS

Error Indicators.

PAR ERR	(11/70 only) Lights to indicate a parity error during a reference to memory.
ADRS ERR	Lights to indicate any of the following addressing errors: • Reference of non-existent memory. • Access control violation. • Reference of unassigned memory pages.

Processor State.

RUN	The CPU is executing program instructions. If the instruction being executed is a *wait* instruction, the RUN light will be on.
PAUSE	The CPU is inactive because the current instruction execution has been completed as far as possible without more data from the UNIBUS or memory or the CPU is waiting to regain control of the UNIBUS (UNIBUS mastership).
MASTER	The CPU is in control of the UNIBUS (UNIBUS Master only when it needs the UNIBUS).

Mode.

USER	The CPU is executing program instructions in USER mode.
SUPER	The CPU is executing program instructions in Supervisor mode.
KERNEL	The CPU is executing program instructions in KERNEL mode.
DATA	If on, the last memory reference was to D (data) address space in the current CPU mode. If off, the last reference was to I (instruction) address space.

** Read Only Memory.

† Floating Point Processor.

Address (11/70 only).

16-bit	Lights when the CPU is using 16-bit mapping.
18-bit	Lights when the CPU is using 18-bit mapping.
22-bit	Lights when the CPU is using 22-bit mapping. This should be lit when running UNIX.

CONSOLE OPERATION – STARTING AND STOPPING

Starting.
While the **HALT/ENABLE [ENABL]** Switch is in the **HALT** position (down), depress the **START** Switch to reset the processor. At this time, an address can be entered on the Switch Register as described above. After the correct switches have been lifted (check the **ADDRESS** Register Display Lights), depress the **LOAD ADRS** Switch to load that address as the starting point of execution, lift the **HALT/ENABLE [ENABL]** Switch to the **ENABLE [ENABL]** position, then depress the **START** Switch to commence execution. Once execution has begun, depressing the **START** Switch again has no effect.

Stopping.
To halt execution of the processor, depress the **HALT/ENABLE [ENABL]** Switch to the **HALT** position. Processing will cease, but the contents of all memory locations will be retained. The switch can then be lifted to the **ENABLE [ENABL]** position with no effect on the system.

Continuing.
After the computer has been stopped, execution can be resumed from the point at which it was halted by using the **CONT** Switch. The function of the **CONT** Switch depends on the position of the **HALT/ENABLE [ENABL]** Switch:

MODE	POSITION	USAGE
ENABLE [ENABL]	Up	CPU resumes normal execution.
HALT	Down	This mode is used for debugging purposes and forces execution of only a single instruction or a single bus cycle. See Section 11.7 of [1] for more details.

BOOT PROCEDURES

INTRODUCTION

The object of the boot procedure is to load a copy of the UNIX operating system, from tape or disk, into memory and execute it. This procedure can be easily facilitated via an optionally supplied DEC‡ hardware bootstrap loader. Depending upon which bootstrap loader is on your system, if any, the address of a dedicated routine can be loaded via the Switch Register and execution started. If your configuration does not include this device, the boot procedure must be manually entered via the Switch Register. See *romboot*(8) of [3] for program listings.

Throughout the remainder of this section, the symbol (CR) is used to denote a carriage return key at the terminal and the symbol *CSW* represents the Console Switches.

BOOTING FROM A ROM

The following procedure is used when booting from a ROM:

1. The Power Key Switch should be in the **POWER** position.
2. The Address Select Knob should be in the **CONS PHY** position.
3. The Data Select Knob should be in the **DATA PATHS** position.
4. Ensure the **HALT/ENABLE [ENABL]** Switch is in the **HALT** (down) position.
5. Depress the **START** Switch to reset the processor.
6. Set the *CSW* to the address of your ROM bootstrap loader procedure (i.e., 165000_8, 173020_8, etc.). If you don't know which address, ask your DEC Customer Engineer (CE).
7. Depress the **LOAD ADRS** Switch to deposit this address into the Switch Register. Ensure the address was loaded correctly by checking the contents of the Address Display Register.
8. Depending upon the ROM, you may have to set the *CSW* to another address specifying from which device you wish to boot (i.e., 000070_8, 000060_8). Do NOT depress the **LOAD ADRS** Switch again, the bootstrap procedure will read this address.
9. Lift the **HALT** Switch to the **ENABLE [ENABL]** position.
10. Depress the **START** Switch to commence execution.
11. A "#" will be printed at the console terminal. You type a **0**, UNIX reponds with an **=**, and you type **unix** followed by a carriage return.

    #0= unix(CR)

    If UNIX was booted properly, four lines of information about the currently running system will be printed:

    - The current operating system.
    - The available user memory.
    - The system's name.
    - The environment mode (single-user).

‡ Digital Equipment Corporation.

MANUAL BOOT PROCEDURE

If your configuration does not include a hardware bootstrap loader, you will have to toggle the boot program into the processor via the *CSW*. The *romboot*(8) manual page in [3] contains program listings for booting off of a variety of devices. The below procedure for manually booting off of an RP04 disk drive will illustrate how to enter one of these programs.

1. Ensure the Power Key is in the **POWER** position.
2. The Address Select Knob must be in the **CONS PHY** position.
3. The Data Select Knob must be in the **DATA PATHS** position.
4. Ensure the **HALT/ENABLE [ENABL]** Switch is in the **HALT** (down) position.
5. Depress the **START** Switch to reset the processor.
6. Choose an arbitrary starting address to begin loading the program. This address must not be too low because the program's execution will overwrite it, and it cannot be too high because the processor will not be able to access it (in the memory management area of memory). The address 004000_8 (only switch 11 up) works well.
7. Set the *CSW* to this starting address and depress the **LOAD ADRS** Switch. You can now begin entering the program.

   Set *CSW* to 012700, lift **DEP** Switch.
   Set *CSW* to 176700, lift **DEP** Switch.
   Set *CSW* to 012720, lift **DEP** Switch.
   Set *CSW* to 000021, lift **DEP** Switch.
   Set *CSW* to 012760, lift **DEP** Switch.
   Set *CSW* to 010000, lift **DEP** Switch.
   Set *CSW* to 000030, lift **DEP** Switch.
   Set *CSW* to 010010, lift **DEP** Switch.
   Set *CSW* to 012740, lift **DEP** Switch.
   Set *CSW* to 000071, lift **DEP** Switch.
   Set *CSW* to 105710, lift **DEP** Switch.
   Set *CSW* to 002376, lift **DEP** Switch.
   Set *CSW* to 005007, lift **DEP** Switch.

   NOTE: The above octal digits represent the program for booting off of an RP04 disk drive only. For any other device, you must use the appropriate program listed in *romboot*(8) in [3].

8. You can check to be sure the program was entered correctly by setting the *CSW* to your starting address (e.g., 004000_8), depressing the **LOAD ADRS** Switch, and depressing the **EXAM** Switch. The first octal digit you entered (012700) should appear in the Data Display Register. By subsequent use of the **EXAM** Switch, the entire program can be listed for inspection.
9. After you are sure the program was entered correctly, reload the starting address (e.g., 004000_8) by setting the *CSW* and depressing the **LOAD ADRS** Switch.
10. Lift the **HALT** Switch to the **ENABLE [ENABL]** position.
11. Depress the **START** Switch.
12. A "#" will be printed at the console terminal. You type a 0, UNIX reponds with an =, and you type **unix** followed by a carriage return.

    #0= unix(CR)

    If UNIX was booted properly, four lines of information about the currently running system will be printed:

- The current operating system.
- The available user memory.
- The system's name.
- The environment mode (single-user).

OPERATOR INSTRUCTIONS

INTRODUCTION

There are two main modes of operation of a PWB/UNIX system: Single-User and Multi-User.

When in Single-User mode, all dial-up ports and hard-wired terminals are disabled and only the console terminal may interact with the processor. This mode of operation enables any changes necessary to be made to the system without any other processing taking place.

Multi-User is the mode in which PWB/UNIX is normally run.

SINGLE USER ENVIRONMENT

After successfully booting the UNIX Operating System, as described in **BOOT PROCEDURES** within this document, a "#" will be typed as a prompt to indicate that the system is ready to receive commands. You may then type any of the commands available followed by a (CR). When the system has completed execution of the command, it will prompt with the "#" again on the next line. The Single User environment is used primarily to do any system maintenance, modification, or repair operations to prepare the system for multi-user mode. The typical sequence of commands to bring the system up into multi-user mode are:

- fsck −t /tmp/junk
- date *MMddhhmmyy*
- init 2

Fsck.
This program will interactively repair any damaged file systems that result from a crash of the operating system. It is also useful to ensure that the file systems have no damage before going into multi-user mode or taking file saves. Usually, you will want to respond "yes" to all the prompts; however, in the event of a system crash, the damage may be extensive enough to warrant recovery from a backup pack. The procedure for this is discussed in **FILE SAVES** in this document. The **−t** option is used to eliminate a prompt for the name of a scratch file if the file system is large. See *fsck*(1M) of [3] for details on the various options available and [4] for a description of all the different errors that can occur.

An example of a check on a consistent file system is illustrated below:

```
# fsck /dev/rrp61

/dev/rrp61
File System: usr Volume: p0603

** Phase 1 – Check Blocks and Sizes
** Phase 2 – Check Pathnames
** Phase 3 – Check Connectivity
** Phase 4 – Check Reference Counts
** Phase 5 – Check Free List
2441 files 16547 blocks 31889 free
#
```

A file system that has experienced some damage can be repaired interactively as shown below. The y is the operator response.

```
# fsck /dev/rrp60

/dev/rrp60
File System: fs1 Volume: p0603

** Phase 1 — Check Blocks and Sizes
POSSIBLE FILE SIZE ERROR I= 2500

** Phase 2 — Check Pathnames
** Phase 3 — Check Connectivity
** Phase 4 — Check Reference Counts
UNREF FILE I= 2500  OWNER= 255 MODE= 100755
SIZE= 0 MTIME= Dec 31 19:00 1969
CLEAR? y

** Phase 5 — Check Free List
2441 files 16547 blocks 889 free

***** FILE SYSTEM WAS MODIFIED *****
#
```

All mountable file systems should be listed in the file /etc/checklist which fsck uses, and these file systems checked each time the system is rebooted.

WARNING: Never execute fsck on an already mounted file system; it will have a bad effect since you are repairing only the physical disk. The only exception to this is the root file system which is always mounted.

An example of repairing the root file system follows:

```
        # fsck /dev/rp0

        /dev/rp0
        File System: root Volume: p0001

        ** Phase 1 — Check Blocks and Sizes
        POSSIBLE FILE SIZE ERROR I= 416

        POSSIBLE FILE SIZE ERROR I= 610

        POSSIBLE FILE SIZE ERROR I= 614

        POSSIBLE FILE SIZE ERROR I= 618

        POSSIBLE FILE SIZE ERROR I= 625

        ** Phase 2 — Check Pathnames
        ** Phase 3 — Check Connectivity
        ** Phase 4 — Check Reference Counts
        UNREF FILE I= 416  OWNER= uucp MODE= 100400
        SIZE= 0 MTIME= Nov 20 16:23 1979
        CLEAR?  y

        UNREF FILE I= 610  OWNER= csw MODE= 100400
        SIZE= 0 MTIME= Nov 20 16:26 1979
        CLEAR?  y
```

```
UNREF FILE I= 625  OWNER= cath MODE= 100400
SIZE= 0 MTIME= Nov 20 16:26 1979
CLEAR?  y

FREE INODE COUNT WRONG IN SUPERBLK
FIX?  y

** Phase 5 - Check Free List
1 DUP BLKS IN FREE LIST
BAD FREE LIST
SALVAGE?  y

** Phase 6 - Salvage Free List

585 files 5463 blocks 4223 free

***** BOOT UNIX (NO SYNC !) *****
#
```

At this time the processor should be halted and the system rebooted.

Date.
Each time the system is rebooted, the software clock must be reset to the correct time of day. The date should only be set once and only in single-user mode. This will prevent confusion when the accounting routines run. The format for setting the date is:

date *MMddhhmmyy*

where:

MM	is the two digit month.
dd	is the two digit day.
hh	is the two digit hour on a 24 hour clock.
mm	is the two digit minute.
yy	is the optional last two digits of the year.

An example of how to set the date is:

```
date 0601073079
```

which would set the date for:

```
Fri Jun  1 07:30:00 EDT 1979
```

More information concerning the date command can be found on *date*(1) in [3].

Init 2.
After you have performed the above consistency checks on the file systems and set the date, the mode of the operating system can be changed to multi-user. This is accomplished by executing the command: /etc/init 2. This command activates processes that: allow users to log on to the system, turn on the accounting and error logging, mount any indicated file systems, and start the cron and any indicated daemons. The operator may have to manually flip the toggles or pop the buttons on the data sets, depending on what type of data set your site has, to allow users to log in. You can now type a *Ctrl/d* character to log off the console terminal and log back in as a normal user.

MULTI-USER ENVIRONMENT

This mode results from the execution of the command: **/etc/init 2**. A user is permitted to access all mounted file systems and execute all available commands. In this mode, an operator can perform file restore procedures and take periodic status checks of the system. Some of these periodic status checks can include:

- A check of free blocks (**df**) remaining on all mounted file systems to ensure a file system does not run out of space.
- A check on rje (**rjestat**).
- A check on **mail** to root or whatever login receives requests for file restores.
- A check on the number of users on the system (**who**).
- A check of all running processes (**ps ax** or **whodo**) to determine if there is some process using an abnormally large amount of CPU time.

OPERATOR DUTIES

INTRODUCTION

This section is meant to serve as a guide to duties normally performed by computer operators. These duties do not represent what an operator's job duties are; they merely outline the general procedures necessary to ensure that users on the system remain contented.

FILE SAVES

Unless timely copies of the file systems are saved, a major system crash could devastate the system's user community.

Almost nothing is worse than working on a project and, just as you are about completed, having the system crash from a lightning storm somewhere; losing the file and all the work you've completed. What *is* worse is when you request a file restore and find out that the last file save was a week ago and that you have nothing to show for a solid week's work.

The easiest way to prevent this problem is to take daily file saves. Then, at most, only a day's work will be lost.

There are two main ways to perform file saves: by disk and by tape. Most sites utilize **volcopy** to perform these file save functions. See *volcopy*(1M) of [3] for more information on the options available and the use of this command. These procedures should normally be performed while in single-user mode, with the file system unmounted, to preclude any file system activity and subsequent damage on the saved copy.

Disk Save Procedures.

Normally this is an automated procedure and is included as part of the site's local operating instructions. You must have at least two (2) disk drives, one of them a spare. The file system to be copied should be unmounted, except for the **root** file system, and an **fsck** executed to ensure consistency. For ease of mapping, file systems are normally saved in the same sections on the backup pack as they exist on the working pack. This is imperative if you are going to boot from the backup version. It is required that the **root** file system reside on section 0 of the pack. The file save procedure is illustrated below. For this example a save of the **root** file system will be made on the spare drive 3. Operator response is indicated in bold type.

```
# volcopy root /dev/rrp0 p0001 /dev/rrp30 p0105
arg.(p0105) doesn't agree with to vol.()
Type 'y' to override:    y
warning! from fs(root) differs from to fs()
Type 'y' to override:    y
From: /dev/rrp0, to: /dev/rrp30? (DEL if wrong)
END: 6000 blocks.
#
```

You should conclude this procedure by executing **fsck** on the saved copy, just to be sure. Again, a backup of a file system that is corrupted is almost as bad as no save at all.

Tape Save Procedures.

Tape saves are necessary for long term storage or for regular saves if you do not have a spare disk drive. Tapes must be labeled before a file save with **volcopy** can be accomplished. If the save will require two or more tapes, *both* tapes must be labeled *before* the **volcopy** is started. To determine the number of tapes the file save will require, try:

```
# volcopy -bpi1600 -feet2400 filesys /dev/rrp?? volume /dev/rmt1 t0001
You will need 1 reels.
From: /dev/rrp??, to: /dev/rmt1? (DEL if wrong)  Hit DEL
#
```

The above procedure assumes you are using 2400 feet reels, and **/dev/rmt1** indicates 1600 bytes-per-inch density. To accomplish the labeling, follow the below example for **/usr**. It is assumed that **t0001** is the tape volume label. If two or more tapes are required, they should be labeled consecutively both externally and internally. The external label should indicate which sequence number the tape is of the set for the file system. Note the use of the **−n** option. Unless you use this option on an unlabeled tape, the program will scan the entire reel looking for a label to change before it rewinds and labels the beginning. This can be very time consuming on 2400 feet reels.

```
# labelit /dev/rmt1 usr t0001 −n
Skipping label check!
NEW fsname = usr, NEW volume = t0001 −− DEL if wrong!!
#
```

After the tapes are labeled, you should then check the disk file system for errors with **fsck**. The actual copy is accomplished much the same as from disk to disk. The only difference is you may have to respond to more questions if the options of **−bpi** and **−feet** are not included on the command line of **volcopy**.

```
# volcopy usr /dev/rrp1 p0001 /dev/rmt1 t0001
Enter size of reel in feet for <t0001>:   2400
Tape density? (i.e., 800|1600|6250)?   1600
You will need 1 reels.
From: /dev/rrp1, to: /dev/rmt1? (DEL if wrong)
END: 35000 blocks.
#
```

FILE RESTORES

If your installation includes daily filesaves as a normal routine, these backup versions of the file systems can provide a user good insurance against the loss of a lot of previous work due to a system crash and subsequent file system damage.

Restoring from Disk.

When a request is made to restore a file from a backup pack, the operator should locate that pack and determine on which section the requested file system resides. Place that pack on a spare drive and power on the drive. You may choose to mount the file system write protected by specifying the **−r** option of **mount**. At the console terminal the operator should log onto the system as **root**. The following example shows the procedure for restoring the file **/usr/adm/acct/sum/tacct** from a previous backup pack. For this example, drive 4 is a spare drive and **/usr** is on section 1 of the backup pack.

```
# mount  /dev/rp41  /bck −r
WARNING!! − mounting: <usr> as </bck>
# ls −l /bck/adm/acct/sum/tacct   (To verify file existence and identify owner.)
−rw−rw−r−− 1 adm     3216 Oct 3 03:29 /usr/adm/acct/sum/tacct
# cp  /bck/adm/acct/sum/tacct  /usr/adm/acct/sum/tacct
# chown adm /usr/adm/acct/sum/tacct
# umount /dev/rp41
#
```

It is usually a good practice for the operator performing the file restore to mail a message to the requester upon its completion. The procedure for this is:

```
# mail user
I have restored the file /usr/adm/acct/sum/tacct
from Friday's backup.
operator's initials
#
```

Restoring from Tape.
If the file does not exist on any of the backup packs or if your installation does not perform disk file saves, then you will have to recover the file from a tape save. It is assumed that tape saves have been performed in the same manner as disk saves, i.e., with **volcopy**. The subject of file saves is discussed in the section **FILE SAVES** within this document. In order to restore a file from tape, the whole file system must first be placed back on a spare section of the disk. The backup version can then be accessed in the same way as described in **Restoring from Disk** within this document. For this example, it is assumed that the usr file system is the second file on the tape and that section 5 of disk drive 0 is a spare section on that disk. It is also assumed that the tape drive has 1600bpi capability; if not, a similar procedure can be followed for 800bpi recorded tapes.

```
(mount tape on tape drive 0)
# echo < /dev/mt4   (space past first file on tape, no rewind)
# volcopy usr /dev/rmt1 t0001 /dev/rrp5 p0001
Enter size of reel in feet for <t0001>:   2400
Tape density? (i.e., 800|1600|6250)?   1600
You will need 1 reels.
From: /dev/rmt1, to: /dev/rrp5? (DEL if wrong)
END: 35000 blocks.
# mount /dev/rp5 /bck
WARNING!! – mounting: <usr> as </bck>
# cp /bck/adm/acct/sum/tacct /usr/adm/acct/sum/tacct
# umount /dev/rp5
#
```

MESSAGE OF THE DAY

When a user logs into the system, part of the login procedure prints out a message of the day. This message can contain several lines of useful information to the user concerning scheduled down-time for hardware preventive maintenance (PM), clean up messages for space-low file systems, or any other useful warnings to which users may need to be alerted. The trick to maintaining this file is to keep it short and to the point. A user does not want to wait ten minutes while eloquent and wordy dialogue is spewed from the terminal before he or she can begin working.

The contents of this message is stored in the file **/etc/motd**. You may change the contents of this file by using the UNIX text editor (see *ed*(1) in [3]). A sample of adding and deleting a line from this file is shown below.

```
# ed /etc/motd
26
p
9/23: Reboot at 5pm today.
d
a
9/24: Down for PM 1700-2100 on 9/30.
w
37
q
#
```

You can also remove the contents of the entire file by:

```
# cp /dev/null /etc/motd
#
```

SYSTEM SHUTDOWN

Whenever the system must be shutdown, such as for file saves or a reboot, the program **/etc/shutdown** should be used. This program is the graceful way to bring the system into single-user mode. You can specify the amount of grace period between sending a warning message out and actually shutting down. This grace period is the number of seconds of delay. You may, optionally, send your own message. A default message is sent to all logged in users if you don't type your own. The following shows a sample session of shutting the system down.

```
# /etc/shutdown 300   (5 minute grace period)
shutdown: you must be in the root directory (/) to use shutdown
# cd /
# /etc/shutdown 300

SHUTDOWN PROGRAM

Thu Sep  1 18:51:58 EST 1979


Do you want to send your own message? (y or n):   y
Type your message followed by ctrl d....
System coming down for filesaves!
Please log off.
(Cntl/d)

System coming down for filesaves!
Please log off.
(waits for 5 minutes)
SYSTEM BEING BROUGHT DOWN NOW ! ! !

Busy out (push down) the appropriate
phone lines for this system.

Do you want to continue? (y or n):   y
Error logging stopped
Hasp stopped
Process accounting stopped.
```

```
All currently running processes will now be killed.

Changing init states, continue (y or n):  y
pwba
single-user

  PID TTY   TIME CMD
    0  ?  187:48 swapper
    1  ?    0:03 INIT 1
11061  co   0:04 -sh
25023  co   0:03 /etc/shutdown
25052  co   0:23 ps ax

Will a file save be done at this time?
Type either ( y or n ) :  y
Want to run fsck at this time?
Type either ( y or n ) :  y
fsck will now be executed on files in checklist

        •
        •
        •


Halt the system when ready.
```

At the completion of this program you can either halt the system, start the file save routine, reboot the system, or bring it back to multi-user mode.

SYSTEM CRASH RECOVERY

An operating system is considered to have "crashed" when it halts itself without being asked to. The reason for the halt is often unknown and can be hardware failure or software related. It is important, for obvious reasons, to determine the nature of the crash so that it will not happen again. One way to do this is to take a dump of memory on tape so that debugging programs can later decipher what processing was going on at the time the crash occurred. The method for this is:

1. Mount a tape on drive 0 with a write ring in.
2. Set *CSW* to the address 000044_8 (switches 5 and 2 up).
3. Lift the **HALT** Switch to the **ENABLE [ENABL]** position.
4. Depress the **START** Switch.

When the tape has rewound, unmount it and afix a label with the date and time of the crash written on it.

You should now attempt to reboot UNIX as described in **BOOT PROCEDURES** in this document. If the system fails to reboot, the operating system was probably damaged in the crash. Now is the time to pull that vital backup version of the root file system off the shelf and use it for the reboot.

When you have finally rebooted the system, it is likely to have a lot of file system damage. If this damage is extensive enough, you may have to restore the entire file system. A sample of this is:

```
    (mount the backup pack on a spare drive)
    # volcopy fs1 /dev/rrp43 p0625 /dev/rrp3 p0601
    From: /dev/rrp43, to: /dev/rrp3? (DEL if wrong)
    END: 65000 blocks.
    #
```

Be sure to run fsck on all the mountable file systems before setting the date and going to multi-user mode.

SYSTEM ERROR MESSAGES

INTRODUCTION

Sometimes before UNIX crashes, it has time to print some error messages and warnings. You may notice if you are logged in as a normal user, that the system will stop everything for a period of time while it is printing messages to the console terminal (remember to leave at least one console switch up!). There are a wide variety of messages that can occur but there are only two distinct types:

Fatal – System failure is imminent, and
Warning – Something is happening that may lead to a system failure.

SYSTEM ERROR MESSAGES – FATAL

panic: no clock
: Neither the KW11-L nor the KW11-P was found at their standard UNIBUS addresses.

panic: buffers
: Insufficient memory space was found when the system was attempting to allocate the non-addressable buffer pool.

panic: iinit
: An error occurred while the system was reading in the superblock of the root file system.

< loop at User location 6 >
: The initialization and line monitor program, /etc/init, cannot be executed.

panic: IO err in swap
: An unrecoverable error has occurred during a system swap operation.

panic: Out of swap
: Insufficient space was found on the system swap device when attempting to allocate buffering for the arguments to a process overlay.

panic: out of swap space
: Insufficient space was found on the swap device when attempting to swap out a process or a copy of a pure text image.

panic: trap
: An unexpected system fault has occurred. This message is preceded by the type of trap and the location of the currently running process.

death
: A system stack overflow has occurred, typically caused by repetitive interrupting of a faulty I/O device.

panic: parity
: A memory system error has occurred in the realm of the operating system address space. When this occurs in a User process, that process is terminated without a panic.

Timeout table overflow
: The system timeout table - used to implement software interrupts - has overflowed while attempting to add another entry.

panic: devtab
: The list header for the chain of buffers attached to a block type device cannot be found.

panic: blkdev
: The major device number of a block type device exceeds the number of such devices in the system. This error should have been detected earlier.

panic: no fs
: The superblock of a mounted file system cannot be found.

panic: no imt
: A mount point was not found in the system mount table when traversing a file system boundary.

panic: no procs
: A process table entry cannot be found during a process fork when it is known that an entry is available.

SYSTEM ERROR MESSAGES — WARNINGS

no space on dev major/minor
: The corresponding file system has run out of available free blocks.

Out of inodes on dev major/minor
: The corresponding file system contains no more free file control blocks.

bad block on dev major/minor
: A block number not in the valid range of available free blocks on a file system has been detected.

Bad free count on dev major/minor
: A corrupted freelist block has been detected while attempting to allocate a new block for a file.

bad count on dev major/minor
: The super block parameters for free blocks and inodes have become corrupted for this file system.

Inode table overflow
: The system file control block table has overflowed. An access to a currently unused file has failed.

No file
: The system file access control table has overflowed. A new reference to a file has failed.

Out of text
: The system shared text program control table has overflowed. An attempt to execute a currently unused shared text program has failed.

Power fail #
A power fail condition has been detected. If power fail recovery has been specified in the system configuration, the initialization process will be informed.

Stopped
Printed after a power fail condition if recovery has not been specified. The system is halted.

iaddress > 2^24
When updating the file control block for a file, a block number in the inode was found to be greater than that permissible.

proc on q
When making a process runnable, after the occurrence of a wakeup event, it was found that the process was already on the system run queue.

parity
A system memory error has occurred. This message is followed by the contents of the low error address register, the high error address register, the memory system error register and the memory control register.

Stray interrupt at vector
A device has interrupted through a vector not specified in the configuration description.

RP04/5/6 drive # not available
The designated drive is no longer available for I/O operations due to an error condition.

hard err on RP04/5/6 rpds rper1 rper2 rper3
After a certain number of unsuccessful retry attempts, a hard error still exists on a drive access. The register contents are those of the drive status register and the 3 drive error registers.

DMC# lost block
The buffer header for a dmc transfer operation claimed to be completed cannot be found.

RS03/4 not available
An RS03 or RS04 drive is no longer accessible.

dzk: xint
In a system with DZ11 multiplexors with KMC11 assist, a transmitter interrupt has occurred.

Hardware Error.
When a hardware error occurs on a block type device, certain device dependent registers are displayed. The message is of the form:
```
        err on dev major/minor
```
followed by:
```
        bn= # er= #, #
```
This is the block number in error followed by an error register and a control register whose contents can be interpreted in [5] under the appropriate device.

DEVICE	ERROR REGISTER	CONTROL REGISTER
RP06	RPER1	RPCS2
RS04	RSCS2	0
TE16	MTER	MTCS2
RP03	RPCS	RPDS
RK05	RKER	RKDS
RF11	RFCS	RFDAE

REFERENCES

[1] *PDP 11/70 Processor Handbook*, Digital Equipment Corporation, 1977-78.

[2] *PDP11 04/34/45/55 Processor Handbook*, Digital Equipment Corporation, 1976-77.

[3] *PWB/UNIX User's Manual*, Release 2.0, June 1979.

[4] *File System Check Program: Interactive File System Repair*, by T. J. Kowalski, Bell Laboratories.

[5] *PDP11 Peripherals Handbook*, Digital Equipment Corporation, 1978-79.