# TVME8240A-PMON

## TVME8240A BootROM Monitor

Version 2.3.x

## User Manual

Issue 2.3.0

June 2008

**TEWS TECHNOLOGIES GmbH**
Am Bahnhof 7
25469 Halstenbek, Germany
www.tews.com

Phone: +49 (0) 4101 4058 0
Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com

**TEWS TECHNOLOGIES LLC**
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com

Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

## TVME8240A-PMON

TVME8240A BootROM Monitor

Supported Modules:
TVME8240A

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

| Issue | Description | Date |
|---|---|---|
| 1.0 | First Issue | October 29, 2002 |
| 1.1 | New option “-b” for pci command | December 9, 2002 |
| 1.2.0 | General Revision - new commands added | February 19, 2007 |
| 2.0.0 | Support TVME8240 V2.0 | June 5, 2007 |
| 2.1.0 | Support TVME8240 V2.1 | December 3, 2007 |
| 2.2.0 | New product name TVME8240A | March 27, 2008 |
| 2.3.0 | Support for TVME8240A-21/22 | June 2, 2008 |

# Table of Contents

# 1 Introduction

The TVME8240A-PMON BootROM monitor is a powerful evaluation and debugging tool for PowerPC based systems. The embedded Unix-style command shell offers an easy to use interface to all implemented PMON commands. The command history, command line editing and the environment variable facility makes the usage of the TVME8240A-PMON Monitor comfortable and effective.

The TVME8240A-PMON includes the following facilities:

- Downloading of files and images via a serial or network connecting.
- Booting from network (TFTP) or SCSI disk
- Autoboot from network, FLASH memory and SCSI Disk
- Memory display and modify commands
- Executing and debugging of user programs (breakpoint, step, trace)
- Environment control functions
- Programming of downloaded images into the FLASH memories
- Versatile diagnostic commands

The TVME8240A-PMON BootROM monitor supports the boards listed below:

| | | |
|---|---|---|
| TVME8240A-11 | MPC8245 300 MHz, 64 MB SDRAM,<br>2 + 8 MB FLASH, Fast Ethernet,<br>Front Panel I/O | Standard 6U VME |
| TVME8240A-12 | MPC8245 300 MHz, 64 MB SDRAM,<br>2 + 8 MB FLASH, Fast Ethernet, Ultra SCSI,<br>Front Panel I/O | Standard 6U VME |
| TVME8240A-21 | MPC8245 300 MHz, 256 MB SDRAM,<br>2 + 32 MB FLASH, Fast Ethernet,<br>Front Panel I/O | Standard 6U VME |
| TVME8240A-22 | MPC8245 300 MHz, 256 MB SDRAM,<br>2 + 32 MB FLASH, Fast Ethernet, Ultra SCSI,<br>Front Panel I/O | Standard 6U VME |
| TVME8240A-50 | MPC8245 300 MHz, 64 MB SDRAM,<br>2 + 8 MB FLASH, Fast Ethernet,<br>Front Panel I/O | Standard 6U VME |
| TVME8240A-51 | MPC8245 300 MHz, 64 MB SDRAM,<br>2 + 8 MB FLASH, Fast Ethernet,<br>Front Panel I/O, 32-Bit IP-Interface | Standard 6U VME |

To get more information about the features and use of the supported devices it is recommended to read the manuals listed below.

| |
|---|
| TVME8240A User manual |
| TVME8240A Engineering Manual |

# 2 Preparing the TVME8240A

After manufacturing the system clock is stopped. If you start the TVME8240A the first time you have to start and set the system clock.

The format is “date yyyymmddhhmm.ss”; where yyyy is year, mm is month, dd is day of month, hh is hour (24-hour format), mm is minutes and ss is seconds. This command starts the system clock and sets the current date and time. In the example: February 8, 2005 3:11 PM.

Enter the following command at the PMON prompt:

```
PMON> date 200502081511.00
```

## 2.1 Setup PMON for Network Download

Before PMON is able to talk over the Internet the Ethernet interface must be selected and a valid Internet Protocol (IP) address must be configured. The environment variable *ifconfig* specifies the Ethernet interface to be used. Valid settings are *fxp0* (default) for front panel Ethernet connector and *fxp1* for P2 Back I/O Ethernet connector. The environment variable *ipaddr* specifies the Internet Protocol address, of this node. Optional a network mask (*netmask*) and a gateway address (*gateway*) for indirect access to a network can be specified.

```
PMON> set ifconfig fxp0
PMON> set ipaddr 10.0.29.234
PMON> set netmask 255.0.0.0
```

All environment variables will be stored in the NVRAM and therefore be available after the next start-up.

TVME8240A-PMON must be restarted if *ifconfig, ipaddr, netmask* or *gateway* were changed. Please refer to chapter 4.4.4 for detailed information regarding PMON environment variables.

# 3 VME Bus Facility

After startup PMON opens three VME master windows to allow direct access to most popular VME address spaces. If the address range or the address modifiers isn't suitable, any possible VME address space can be mapped with the *vmemaster* command. On the other hand any local address space can be mapped to the VME bus with the *vmeslave* command.

The following VME bus address spaces are mapped:

| Window | VME Space/Width | PMON Address | VME Address | Size |
|---|---|---|---|---|
| 0 | A16/D16 | F1FF0000 | 0000 | 64 KB |
| 1 | A24/D16 | F0000000 | 000000 | 16 MB |
| 2 | A32/D32 | A0000000 | 00000000 | 256 MB |

### See Also

The *info*, *vmemaster* and *vmeslave* commands.

# 4 PMON Commands

## 4.1 Download Files / Images

### 4.1.1 Load File (load)

The load command downloads programs and data from the host.

**Format**

The format for the load command is:

```
load  [-abefistvy][-u baud][-o offs][-c cmd][-h port]
```

where:

| | |
|---|---|
| -n | don’t load symbols |
| -a | suppresses addition of an offset to symbols. |
| -b | suppresses deletion of all breakpoints before the download. |
| -e | suppresses clearing of the exception handlers. |
| -i | ignores checksum errors. |
| -s | suppresses clearing of the symbol table before the download. |
| -t | loads at the top of memory. |
| -y | only load symbols. |
| -v | verbose messages. |
| -c cmd | Sets a command string that the Monitor sends to the host to start a download operation. String cmd is the string that starts the download. Note that the command string must be enclosed in double quotation marks if the string contains any spaces. |
| -o offs | loads at the specified offset. |
| -u baud | set baud rate |
| -h port | load from <port> |

Invoking the load command with no parameters or arguments clears the symbol table, deletes all current breakpoints, allows the Monitor to receive programs or data from the host, and uses the current baud rate by default.

**Functional Description**

The load command accepts programs and data from the host port in Motorola S-record files. The user can set environment variables to change the data port, the format, and the transfer protocol.

The load command normally clears the symbol table, exception handlers, and all breakpoints. The -s and -b options suppress the clearing of the symbol table and breakpoints, respectively. The value of the EPC register is set automatically to the entry point of the program. Therefore, to execute the downloaded program, only the g command is required.

The -c option permits a command string to be sent to the host when the load command is issued. This is intended for use in conjunction with the transparent mode. Note that if the command string contains multiple words, the command must be enclosed in double quotation marks, as shown in the example below.

## The dlecho, dlproto, and hostport Variables

The dlecho, dlproto, and hostport variables control operation of the download. The following table shows how these environment variables affect the operation of the load command.

| Variable | Action |
|---|---|
| dlecho off | Do not echo the lines |
| dlecho on | Echo the lines |
| dlecho lfeed | Echo only a linefeed for each line |
| dlproto none | Do not use a protocol |
| dlproto XonXoff | Send Xon and Xoff to control the host |
| dlproto EtxAck | Expect Etx as end of record, send Ack |
| hostport tty0 | Select tty0 as the port to which the host is connected |
| hostport tty1 | Select tty1 as the port to which the host is connected |

See the section on downloading for more information on these variables and the use of the load command.

## Examples

Download to tty0 using a terminal emulator.

```
PMON> set dlecho off

PMON> set hostport tty0

PMON> set dlproto none

PMON> load -o 800000 -h tty0
Downloading from tty0, ^C to abort

total = 0x8ec38 bytes
Entry Address  = 00800000

PMON>
```

## 4.1.2 Boot (boot)

The boot command loads binary object files over a specified interface device, and optionally: will load an image into on-board flash ROM.

### Format

The format for this command is:

```
boot  [-f flash_addr] [-o offset_load] [-l load_addr]
[-x exe_addr] [args]...
```

where:

| | |
|---|---|
| host:path | internet host name, and file name (seperated by a colon ":" ) |
| or | |
| /dev/file | local device and file name path....e.g., /sd0/bsd |
| -f flash_addr | base address (in flash ROM) to load an image via host:path (above) |
| -o offset_load | tells flash loader to assign this transfer address to the image |
| -l load_addr | Setup the load address for straight binary files. Any address in the 64 MB RAM higher than 0x100000 can be used |
| -x exe_addr | Setup the execution/entry address for straight binary files so only the g command is required to start execution |

Invoking the boot command with no parameters or arguments clears the symbol table, deletes all current breakpoints, and attempts to load the program found in the host and file specified by the bootaddr and bootfile environment variables.

### Functional Description

The boot command is a wrapper for netboot or scsiboot and uses the TFTP (Trivial File Transfer Protocol) to load an executable binary file from a remote host over Ethernet, or the built-in scsi bootloader module for loading from a local disk drive.

TVME8240A-PMON boot can read files in ELF format as used in:

- OpenBSD PowerPC 32-Bit ELF
- Wind River VxWorks
- Some Linux for PowerPC implementations

TVME8240A-PMON extracts any symbol table information from these files, and adds it to the target symbol table.

The boot command normally clears the symbol table, exception handlers, and all breakpoints. The -s and -b options suppress the clearing of the symbol table and breakpoints, respectively. The value of the cpc register is set automatically to the entry point of the program. Therefore, to execute the downloaded program, only the g command is required. This is particularly useful when an image has been loaded into flash ROM.

The boot command may return a large number of different error messages, relating to network problems or file access permissions on the remote host. For a file to be loaded via TFTP it must be publicly readable, and it may have to be in a directory which is acceptable to the remote server.

The -f (flash_load) option tells the boot wrapper to network load an image into the FLASH ROM area designated as an address argument. The FLASH ROM area specified must be large enough to accept the image. The area specified will first be erased, then the image loaded and then verified. TVME8240A-PMON uses some temporary RAM to hold the image so it is important that the amount of free RAM exceed the size of the image (and the FLASH area). This is rarely violated.

The -o (offset) option provides for assigning a transfer & run (execute) address header for the image that is being loaded into FLASH ROM. This option is critical when the image being loaded into FLASH ROM is, in fact, destined to be executed later out of RAM at another address. TVME8240A-PMON itself uses this feature because it is stored in FLASH ROM (usually) at one address, but moved and run from a RAM address on most platforms. The -o offset value specifies the actual address where execution is to begin AFTER an image is transferred by the g command.

Boot an image into FLASH ROM example:

```
PMON> boot -f FF000100 -o 800000 10.0.0.1:bootrom_uncmp
Loading file: 10.0.0.1:bootrom_uncmp (elf)
0x100000/584760 + 0x18ec38/12836(z)
Programming flash 0x00100000:0x00091e5c into 0xff000100
...
Programming FLASH. Done.
```

In the above example, the boot -f command was used to load a new *VxWorks* boot image into the 8 MB 64-bit wide expansion FLASH. The image was resident on a local network server, in the */tftpboot* directory under the name of *bootrom_uncmp* and was a standard PowerPC ELF-32 bit image created with Tornado 2.0.2. The image file is loaded and we get to see the binary file characteristics echoed to the console while the load is happening. The boot -f command also informs us of the transfer address (0x00100000:0x00091e5c) and size, as well as confirmation of the stored location (into 0xff000100). Next it tells us which flash ROM block(s) are being erased to hold the new image, followed by confirmation that the flash ROM is being programmed, and then reports its success.

Boot straight binary image into DRAM from address 0x100000 and set entry address to 0x100020.

```
PMON> boot -l 100000 -x 100020 10.0.0.1:tvme8240a.kdi
Loading file: 10.0.0.1:tvme8240a.kdi-
0x139b42/1284930 bytes loaded
Entry address is 00100020
```

When reading the symbol table TVME8240A-PMON may complain that it does not have enough room to store the program's symbols. To increase the size of the heap, use the set *heaptop* command to reserve more space and, if necessary, re-link your program with a higher base address. The boot command will also detect cases where the program being loaded would overwrite PMON's crucial data or heap: again re-linking your program at a different address will cure the problem.

While it is disk loading each section of the file, boot displays the memory address (in hex) and size (in decimal) of that section. Typically these sections will be in the order .text, .data and .bss.

## See Also

The *scsiboot*, *netboot* and *flash* commands.

## 4.1.3 Network Boot (netboot)

The *netboot* command loads binary object files over Ethernet interface.

### Format

The format for this command is:

```
netboot [-abeinstwy] [-f addr] [-o offs] [-l load_addr]
[-x exe_addr] [host:[ path]]
```

where:

| | |
|---|---|
| -a | don't add offset to symbols |
| -b | suppresses deletion of all breakpoints before the download. |
| -e | suppresses clearing of the exception handlers. |
| -i | ignore checksum errors |
| -n | suppresses the loading of symbols from the file. |
| -s | suppresses clearing of the symbol table before the download. |
| -t | load at to of memory |
| -y | loads only the symbols from the file. |
| -w | reverse endianness |
| host | is the internet host from which to read the file. |
| path | is the file name to be loaded from the host. |
| -f addr | base address (in flash ROM) to load an image via host:path (above) |
| -o offs | tells flash loader to assign this transfer address to the image |
| -l load_addr | Setup the load address for straight binary files. Any address in the 64 MB RAM higher than 0x100000 can be used |
| -x exe_addr | Setup the execution/entry address for straight binary files so only the g command is required to start execution |

Invoking the *netboot* command with no parameters or arguments clears the symbol table, deletes all current breakpoints, and attempts to load the program found in the host and file specified by the bootaddr and bootfile environment variables.

### Functional Description

The *netboot* command uses the TFTP (Trivial File Transfer Protocol) to load an executable binary file from a remote host over Ethernet.

TVME8240A-PMON can read files in ELF 32-Bit format as used in:

- Wind River VxWorks
- Some Linux for PowerPC implementations
- OpenBSD PowerPC 32-Bit ELF

TVME8240A-PMON extracts any symbol table information from these files, and adds it to the target symbol table, unless overridden on command line.

The *netboot* command normally clears the symbol table, exception handlers, and all breakpoints. The -s and -b options suppress the clearing of the symbol table and breakpoints, respectively. The value of the cpc register is set automatically to the entry point of the program. Therefore, to execute the downloaded program, only the g command is required.

The *netboot* command may return a large number of different error messages, relating to network problems or file access permissions on the remote host. For a file to be loaded via TFTP it must be publicly readable, and it may have to be in a directory which is acceptable to the remote server.

The -f option tells the boot wrapper to network load an image into the FLASH ROM area designated as an address argument. The FLASH ROM area specified must be large enough to accept the image. The area specified will first be erased, then the image loaded and then verified. TVME8240A-PMON uses some temporary RAM to hold the image so it is important that the amount of free RAM exceed the size of the image (and the flash area). This is rarely violated.

The -o (offset) option provides for assigning a transfer & run (execute) address header for the image that is being loaded into FLASH ROM. This option is critical when the image being loaded into FLASH ROM is, in fact, destined to be executed later out of RAM at another address. TVME8240A-PMON itself uses this feature because it is stored in FLASH ROM (usually) at one address, but moved and run from a RAM address on most platforms. The -o offset value specifies the actual address where execution is to begin AFTER an image is transferred by the g command.

Boot an image into FLASH ROM Example:

```
PMON> netboot -f 70000100 -o 800000 10.0.0.1:bootrom_uncmp
Loading file: 10.0.0.1:bootrom_uncmp (elf)
0x100000/584760 + 0x18ec38/12836(z)
Programming flash 0x00100000:0x00091e5c into 0x70000100
Erasing FLASH block  0 Done.
Erasing FLASH block  1 Done.
...
Erasing FLASH block 36 Done.
Programming FLASH. Done.
PMON>
```

In the above example, the netboot -f command was used to load a new *VxWorks* boot image into the 8 MB 64-bit wide expansion FLASH. The image was resident on a local network server, in the */tftpboot* directory under the name of *bootrom_uncmp* and was a standard PowerPC ELF-32 bit image created with Tornado 2.0.2. The image file is loaded and we get to see the binary file characteristics echoed to the console while the load is happening. The boot -f command also informs us of the transfer address (0x00100000:0x00091e5c) and size, as well as confirmation of the stored location (into 0x70000100). Next it tells us which FLASH ROM block(s) are being erased to hold the new image, followed by confirmation that the FLASH ROM is being programmed, and then reports its success.

Boot straight binary image into DRAM from address 0x100000 and set entry address to 0x100020.

```
PMON> netboot -l 100000 -x 100020 10.0.0.1:tvme8240a.kdi
Loading file: 10.0.0.1:tvme8240a.kdi-
0x139b42/1284930 bytes loaded
Entry address is 00100020
PMON>
```

When reading the symbol table TVME8240A-PMON may complain that it does not have enough room to store the program's symbols. To increase the size of the heap, use the set heaptop command to reserve more space and, if necessary, re-link your program with a higher base address. The boot command will also detect cases where the program being loaded would overwrite PMON's crucial data or heap: again re-linking your program at a different address will cure the problem.

While it is loading each section of the file, boot displays the memory address (in hex) and size (in decimal) of that section. Typically these sections will be in the order .text, .data and .bss.

### See Also:

scsiboot, boot and load commands.

## 4.1.4 Disk Boot (scsiboot)

The *scsiboot* command loads binary object files over a specified local SCSI interface device.

### Format

The format for this command is:

```
scsiboot  [-xbensy] [[-m]-d name] [-l len] [-a addr]
          [-o offs] [args]...
```

where:

| | |
|---|---|
| -x | don't exec boot loaded file |
| -b | suppresses deletion of all breakpoints before the load. |
| -e | suppresses clearing of the exception handlers. |
| -y | loads only the symbols from the file. |
| -d name | specify scsi device (e.g., sd0) |
| -l len | length in disk blocks |
| -a addr | memory destination for loaded file |
| -o offs | offset in disk blocks |
| -m | manually specify boot dev/file (otherwise use env var: bootdev) |
| -- [args]... | args to be passed to client, e.g. bsd -s |

Invoking the *scsiboot* command with no parameters or arguments clears the symbol table, deletes all current breakpoints, and attempts to load the program found on the device specified by the bootdev, bootaddr, bootlen, and bootfile environment variables.

### Functional Description

The *scsiboot* command is used to load an executable binary file with the built-in scsi bootloader module from a local disk drive.

TVME8240A-PMON *scsiboot* can read files in ELF format as used in:

- Algorithmics' SDEMIPS
- newer SGI compilers
- systems compliant with the MIPS/ABI standard
- older MIPS ECOFF format
- Wind River VxWorks
- OpenBSD PowerPC 32-Bit ELF

TVME8240A-PMON extracts any symbol table information from these files, and adds it to the target symbol table, unless overridden on command line.

The *scsiboot* command normally clears the symbol table, exception handlers, and all breakpoints. The -s and -b options suppress the clearing of the symbol table and breakpoints, respectively. The value of the PC register is set automatically to the entry point of the program. Therefore, to execute the downloaded program, only the g command is required.

The *scsiboot* command may return a large number of different error messages, relating to disk problems or file access permissions on the local machine.

When reading the symbol table TVME8240A-PMON may complain that it does not have enough room to store the program's symbols. To increase the size of the heap, use the set *heaptop* command to reserve more space and, if necessary, re-link your program with a higher base address. The *scsiboot* command will also detect cases where the program being loaded would overwrite PMON's crucial data or heap: again re-linking your program at a different address will cure the problem.

While it is loading each section of the file, *scsiboot* displays the memory address (in hex) and size (in decimal) of that section. Typically these sections will be in the order .text, .data and .bss.

### See Also:

*netboot*, *boot* and *load* commands.

### 4.1.5 Define a Symbol (sym)

The sym command sets a symbolic name for a variable.

#### Format

The format for this command is:

```
sym name value
```

where:

| | |
|---|---|
| name | is the name of the variable for which a value is to be set. |
| value | is the value to which the variable is set. |

#### Functional Description

The *sym* command sets a symbolic name to the specified value.

Normally the file load commands clear the symbol table. However, there is an option to override the clearing of the symbol table (see the load, netboot and scsiboot commands for details).

Symbols can be displayed using the ls command.

Examples illustrating the use of this command follow.

```
PMON> sym start 9fc00240

PMON> sym flush_cache 9fc016f0

PMON> l start 4

start+0x240 3c09a07f    lui     t1,0xa07f
start+0x244 3c08003c    lui     t0,0x3c
start+0x248 3529ff20    ori     t1,t1,0xff20

PMON> l 9fc0027c 5

start+0x27c 03a1e825    or      sp,sp,at
start+0x280 0ff005bc    jal     flush_cache
start+0x284 24040000    addiu   a0,zero,0x0
start+0x288 0ff005bc    jal     flush_cache
start+0x28c 24040001    addiu   a0,zero,0x1
```

#### See Also

ls, load, l, and sh commands.

## 4.1.6 List Symbols (ls)

The *ls* command lists the current symbols in the symbol table.

### Format

The format for this command is:

```
ls [-ln] [sym|[-[v|a] adr]
```

where:

| | |
|---|---|
| -l | provides a long listing, showing the address value for each symbol. |
| -n | lists the symbols in ascending order of address. |
| *sym* | is a pattern filter for the symbols to be shown. Both character wildcards ("?") and word wildcards ("*") are permitted. |
| -v | is the verbose option, showing the value in hexadecimal, decimal, and octal. |
| -a | shows the address in symbolic form. |
| *adr* | is the address for which a symbol or offset from a symbol is sought. |

Invoking the ls command without any options or parameters lists the symbols in alphabetical order without displaying the actual address for each symbol.

### Functional Description

The *ls* command lists the symbols in the symbol table.

The -l option produces a long listing, which includes the address value of each symbol. The -n option causes the symbols to be listed in ascending order of address. The -a address option lists the symbol at the next lowest address. The -v address option prints the result in hex, decimal, and octal. The -v option is useful for computing the value of an expression that may include registers, symbols, and absolute values. Examples illustrating the use of the *ls* command follow.

| | |
|---|---|
| `PMON> ls`<br>`flush_cache start` | List symbols in alphabetic order. |
| `PMON> ls -l`<br>`9fc016f0 flush_cache`<br>`9fc00240 start` | List symbols in alphabetic order with addresses. |
| `PMON> ls -ln`<br>`9fc00240 start`<br>`9fc016f0 flush_cache` | List symbols and addresses in ascending order of address. |
| `PMON> ls s*`<br>`start` | List symbols starting with the letter "s." |
| `PMON> ls -a 9fc00260`<br>`9fc00240 start+0x20` | List symbol at the next lowest address. |
| `PMON> ls -a @cpc`<br>`a0020020 = start+0x20` | List symbol at the next lowest address from Current PC. |

# 4.2 Display / Set Memory and Registers

## 4.2.1 Display/Set Registers (r)

The r command sets or displays register values.

### Format

The format for the r command is:

```
r [reg|* [val|field val]]
```

where:

| | |
|---|---|
| reg | is the name of the register or registers (specified by wildcard characters) to display or modify. |
| val | is the value to which the specified register or registers should be modified. |
| field val | is the value to which the specified field in the specified register should be modified. |
| * | displays the contents of all registers except floating-point registers. |
| f* | displays the contents of all floating-point registers. |

Invoking the r command without any parameters or arguments displays a list of all the general-purpose registers.

### Functional Description

The r command sets or displays register values.

The character and word wildcards, "*" and "?", can be used in the register name. The '?' character matches any single character, while the '*' character matches any number of any characters. This command accepts both hardware and software names.

Examples illustrating the use of the r command follow.

| | |
|---|---|
| `r` | Display all General-purpose registers. |
| `r *` | Display all register values. |
| `r 08` | Display r08 |
| `r r*` | Display r00 through r31. |
| `r f*` | Display all fpu register. |
| `r f17` | Display fpu register 17 |
| `r 04 45` | Set register 04 to 00000045. |

## Examples

Display all regular registers (PowerPC example)

```
PMON> r

r00-07  00000000 03dfffc0 00000000 00000000 00000000 00000000 00000000 0000000
r08-15  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r16-23  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r24-31  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Display all registers. Note that the actual registers that are displayed by the "`r *`" command depends on what type of processor you are using. This display was generated using a PowerPC.

```
PMON> r *

r00-07  00000000 03dfffc0 00000000 00000000 00000000 00000000 00000000 00000000
r08-15  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r16-23  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r24-31  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

f00-03  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f04-07  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f08-11  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f12-15  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f16-19  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f20-23  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f24-27  0000000000000000 0000000000000000 0000000000000000 0000000000000000
f28-31  0000000000000000 0000000000000000 0000000000000000 0000000000000000
fsr = 0000000000000000
cpc = 00100000, lr = 00000000, ctr = 00000000, sr = 00003032
PMON>
```

Set the register 'r02' to a value of 0x00000100

```
PMON> r 02 00000100
PMON>
```

## See Also

l command for disassembling instructions from memory.

## 4.2.2 Modify Memory (m)

The *m* command displays and modifies memory.

### Format

The format for this command is:

```
m [-bhw] [adr [hexval|-s str]..]
```

where:

| | |
|---|---|
| -b | is a flag signifying that byte accesses are to be used. |
| -h | is a flag signifying that halfword (16-bit) accesses are to be used. |
| -w | is a flag signifying that word (32-bit) accesses are to be used. |
| adr | is the memory address to display or modify without entering interactive mode. |
| hexval | is the value to insert at the specified address. |
| -s | is a flag signifying that the following parameter is a string value. |
| str | is a string value to copy to the specified address. |
| | enters interactive mode. |
| = | in interactive mode, reads current address again. |
| ^\|- | in interactive mode, moves back one word. |
| . <cr> | exit interactive mode. |

Entering no values with this command causes the command to operate in interactive mode.

### Functional Description

This command can display and then modify memory locations interactively. This command can also set memory to a specified value directly.

If invoked with one or more values following the address, the command is executed immediately, without entering the interactive mode.

If the command is invoked without a value, the command enters the interactive memory mode. In interactive memory mode, the user enters a command at the cursor. The interactive memory mode first displays the address and its current value. Interactive memory mode then lets the user select one of the commands listed in the following table.

If the -s option is specified, the Monitor displays the memory contents as an ASCII string. A multiple-word string may be specified by enclosing the multiple-word string in quotation marks.

Examples illustrating the use of the *m* command follow.

```
PMON> m a0020000        Display memory at address in interactive mode.

a0020000 00 _           User can enter command at cursor (_).
```

```
PMON> m a0020000 1 2 3 4     Set address 0xa0020000 to 1, address
                             0xa0020001 to 2, etc., in non interactive
                             mode.


PMON> m a0020000             Display memory at 0xa0020000.

a0020000 01 CR
a0020001 02 CR
a0020002 03 CR
a0020003 04 .



PMON> m a0020000 22          Set address 0xa0020000 to 0x22.

PMON> m a0020000             Display memory at 0xa0020000.

a0020000 22 44
a0020001 00
a0020002 00 55
a0020003 00 66
a0020004 00 ^
a0020003 66
a0020004 00 .


PMON> m 80020000 -s even     Set memory starting at 0x80020000 to
                             the string "even."


PMON> m 80030100 -s "PMON 2000"      Set memory starting at 0x80030100
                                     to the multiple-word string
                                     "PMON 2000."
```

## See Also

*fill* command, *l* command, *d* command, and *dump* command.

## 4.2.3 PCI Configuration Register Modify (cm)

The *cm* command displays and modifies PCI configuration registers of the selected PCI device.

### Format

The format for this command is:

```
cm [-bhnw] offs [device [bus]]
```

where:

| | |
|---|---|
| -b | is a flag signifying that byte accesses are to be used. |
| -h | is a flag signifying that halfword (16-bit) accesses are to be used. |
| -w | is a flag signifying that word (32-bit) accesses are to be used (default). |
| -n | is a flag signifying that non-interactive mode is to be used (no write). |
| offs | byte offset within the PCI configuration where to start display and modify |
| device | PCI device number in range from 0 to 1F (default is device 0) |
| bus | PCI bus number in range from 0 to FF (default is bus 0) |
| | enters interactive mode. |
| = | in interactive mode, reads current address again. |
| ^\|- | in interactive mode, moves back one word. |
| . <cr> | exit interactive mode. |

### Functional Description

This command can display and then modify PCI configuration register locations interactively.

If the command is invoked without option *-n*, the command enters the interactive memory mode. In interactive memory mode, the user enters a command at the cursor. The interactive mode first displays the offset and its current value. Interactive memory mode then lets the user select one of the commands listed in the following table.

The default values for the optional arguments: device and bus are 0.

Examples illustrating the use of the *cm* command follow.

```
PMON> cm 0                Display configuration registers from PCI
                          device 0 at bus 0 starting at offset 0.

00000000 00031057 _       User can enter command at cursor (_).


PMON> cm 0 10             Display configuration register from PCI
00000000 903010b5 CR      device 10 (hex) at bus 0
00000004 02800143 CR
```

```
00000008 06800000 .

PMON> cm 10 2 1         Set BAR0 of PCI device 2 at bus 0 to 80000200
00000010 81000000 80000200
00000014 00000001 .
```

### See Also

*pci* command.

## 4.2.4 Display Memory (d)

The d command displays memory contents in hex or ASCII format.

### Format

The format for this command is:

```
d [-b|h|w|s|S] adr [cnt|-rreg]
```

where:

| | |
|---|---|
| -b | displays the memory contents in groups of bytes. |
| -h | displays the memory contents in half-word groups. |
| -w | displays the memory contents in word groups. |
| -s | displays the memory contents as a null terminated string. |
| adr | specifies the base address from which data is displayed. |
| cnt | specifies the number of lines displayed. |
| -rreg | displays the contents of memory as register reg. |

### Functional Description

The d command displays memory, starting at the specified address, in hexadecimal or ASCII format. A -b, -h, -w, or -s option, if specified, sets how the data is displayed. See the examples at the end of this section for illustration of the possible display formats.

The *datasz* and *moresz* Variables in the environment control the display of memory. See set command.

If invoked without a -b, -h, -w, or -s option, the *datasz* variable sets the display format. Setting *datasz* to -b, -h, -w, or -s has the same effect as the command line options of the same names described in this section. The *datasz* variable does not affect any other command displays.

If invoked without *cnt*, the *d* command first displays the number of lines specified in the environment variable *moresz*. The Monitor then pauses and displays the more prompt. See the more command for commands available with the more prompt. Also see the more command for more information on the *moresz* variable.

The following example displays memory starting at 0xa0010000.

```
PMON> d a0010000


a0010000 bf c0 2b 00 bf c0 2b 00 bf c0 2b 00 bf c0 2b 3c ..+...+...+...+<
a0010020 bf c0 2b 20 bf c0 2b a8 bf c0 2b 78 bf c0 2b 60 ..+...+...+x..+`
a0010030 bf c0 2b 48 bf c0 2b a8 bf c0 2b a8 bf c0 2b a8 ..+H..+...+...+.
a0010040 bf c0 2b 78 bf c0 2b 60 bf c0 2b 48 bf c0 2e 78 ..+x..+`..+H...x
a0010050 bf c0 2f 08 bf c0 2e c4 bf c0 2e 80 bf c0 2f 90 ../.........../.
a0010060 bf c0 2f 90 bf c0 2f 90 bf c0 2e 78 bf c0 2e 78 ../.../....x...x
a0010070 bf c0 2e 78 00 00 00 00 00 00 00 00 00 00 00 00 ...x............
```

## 4.2.5 Disassemble Memory (l)

The *l* command disassembles instructions from memory.

### Format

The format for this command is:

```
l [-bct] [adr [cnt]]
```

where:

| | |
|---|---|
| -b | lists only branches. |
| -c | lists only calls. |
| -t | lists the trace buffer. |
| adr | is the base address from which to disassemble instructions. |
| cnt | is the number of lines to disassemble. |

When invoking this command with no options, disassembly starts at the address in the *PC* register and is output to the more command.

### Functional Description

The *l* command disassembles the memory contents, starting either at the *PC* register's current value or at the specified address. The output of this command is passed to the more command, letting the user view one screen full of disassembled output at a time. Optionally, the user can specify a count value, which limits the number of disassembled lines to that number.

The *regstyle* environment variable is an architecture dependent variable which determines whether the Monitor displays hardware or software register names. Hardware register names are simply r00 through r31. Software registers are defined by the PowerPC software conventions

Examples illustrating the use of the l command follow.

```
PMON> l 500
00000500 7c3143a6 mtspr r1, , SPRG1
00000504 bf8002e0 stmw r28, r0, 0x2e0,
00000508 7f8802a6 mfspr r28, , lr
0000050c 7fa00026 mfcr r29,
00000510 7fda02a6 mfspr r30, , srr0
00000514 7ffb02a6 mfspr r31, , srr1
00000518 480108ff bl a 0x108fc,
0000051c 73d4534c andi. r30, r20, 0x534c,
```

### See Also

*d* command, *m* command, *dump* command and *more* commands.

## 4.2.6 Fill Memory (fill)

The fill command writes a hexadecimal pattern or string to a block of memory.

### Format

The format for this command is:

`fill` *`from to`* `{val|-s str}-`

where:

| | |
|---|---|
| from | is the base address for the fill operation. |
| to | is the end address for the fill operation. |
| val | is the hexadecimal value of the byte that is written to the area to be filled. |
| -s str | specifies that the memory block should be filled with an ASCII string rather than a particular value. String str is the ASCII string to be written to the memory block during the fill operation if the -s parameter is specified. |

### Functional Description

The *fill* command fills an area of memory with a specified hexadecimal pattern or repeating string. The pattern can be a single byte or multiple bytes. For the *fill* command to work correctly *to* must be greater than *from*. If the -s option is specified, the next parameter is interpreted as an ASCII string. Multiple-word strings may be specified by enclosing them in quotes.

For example, to clear an area of memory from 0xa0020000 to 0xa0021000, enter:

```
PMON> fill a0020000 a0021000 0
```

To fill an area of memory from 0xa0020000 to 0xa00210000 with the string of values 0x41, 0x42, 0x43, 0x44, and 0x45, enter:

```
PMON> fill a0020000 a00210000 41 42 43 44 45
```

To fill an area of memory from 0xa0020000 to 0xa00210000 with the ASCII string "hello world," enter:

```
PMON> fill a0020000 a0021000 -s "hello world"
```

### See Also

m command.

### 4.2.7 Copy Memory (copy)

The *copy* command copies a specified number of bytes from one location in memory to another.

#### Format

The format for this command is:

```
copy from to size
```

where:

| | |
|---|---|
| from | declares the source address location. |
| to | declares the target address location. |
| size | is the size of the block of memory to be copied. This quantity is specified in bytes. |

#### Functional Description

The copy command replicates a specified number of bytes from one place in memory to another.

If *to* is less than *from*, copying is performed in ascending order starting at *from*. If *from* is less than *to*, copying is performed in descending order starting at *from* + *size*.

When moving a data block down, the source data is copied from the bottom of the block upwards: and when moving a data block up, the source data is copied from the top of the block downwards. By this technique, there is no risk of copying over data in overlapping block move operations; as the data in the overlapping area is copied first.

#### Examples

This example shows how to copy a block of memory, 8 Kbytes in size, with a base address of 0x80020000, to another 8-Kbyte area starting at the address 0x80060000.

```
PMON> copy 80020000 80060000 2000
```

## 4.2.8 Search Memory (search)

The *search* command executes a search for a memory pattern.

### Format

The format for this command is:

```
search from to {val|-s str}-
```

where:

| | |
|---|---|
| from | is the start address for the search operation. |
| to | is the end address for the search operation. |
| val | is the hexadecimal value that is the object of the search. |
| -s str | specifies that the search operation is for a string str. |

### Functional Description

The *search* command searches memory for a pattern. The pattern may be a single byte, multiple bytes, or an ASCII string.

If the -s option is specified, the next parameter is interpreted as an ASCII string. To search for a multiple-word string, enclose the string in double quotation marks.

The output of this command is printed to the screen via the more command.

The following example searches for 0x3c and 0xd4 from 0xa0020000 to 0xa0030000:

```
PMON> search a0020000 a0030000 3c d4
```

The following example searches for "ABC" from 0xa0020000 to 0xa0030000:

```
PMON> search a0020000 a0030000 -s "ABC"
```

### See Also

*d* command and *more* command.

## 4.2.9 Dump Memory (dump)

The dump command uploads S-records to the host port.

### Format

The format for the dump command is:

```
dump adr siz
```

where:

| | |
|---|---|
| adr | is the base address of the data to be uploaded |
| siz | the number of bytes to be uploaded |

### Functional Description

The *dump* command uploads Motorola S-records to the host port. All uploaded S-records except the terminating S-record are S3-records. The terminating S-record is an S7-record.

The *uleof* and *ulcr* Variables affect dump behavior.

After the *dump* is completed, the string specified in *uleof* will be transmitted. The default value for *uleof* is "%".

If the variable *ulcr* is set to "`off`", the lines will be terminated by a carriage return ("\r") and a linefeed character ("\n").

If *ulcr* is set to "`on`", each line will be terminated by a linefeed character ("\n") only.

The default value for *ulcr* is "`off`".

The following example of the dump command dumps 128 bytes starting at 0xFFF000100.

```
PMON> dump FFF000100 80
S325FF000100480000694800006D436F7079726967687420313938342D313939392057696E640A
S325FF0001202052697665722053797374656D732C20496E632E20202020202020202020202066
S325FF000140436F707972696768742032303032202054455753205445434484E4F4C4F47494529
S325FF0001605320476D624800003BE00002480000097C7F1B787C0002787C1043A67C1143A6CC
S70500000000FA
%PMON>
```

### See Also

*l* command, *d* command, and *m* command.

# 4.3 Execution Control and Breakpoints

## 4.3.1 Start Execution (g)

The *g* command starts program execution.

### Format

The format for this command is:

```
g [-ste] [adr [bptadr]] [-c args]
```

where:

| | |
|---|---|
| -s | is a flag indicating that the stack pointer, sp, should not be set |
| -t | is a flag that causes the execution time of the client program to be displayed in seconds. |
| -e | changes behavior of 'adr' (below) so that the actual execution address is extracted from the image. |
| adr | is the address of the first instruction to be executed. |
| bptadr | is a breakpoint address where program execution is to be stopped. This breakpoint is removed the next time that execution halts. |
| -c args | indicates that the argument or arguments args are to be passed to the client program. No more options to the g command itself can be given after this option. |

By default, the *g* command starts program execution at the address in the *CPC* register, and sets the stack pointer register, to the end of the stack area. Note that the stack grows downwards.

### Functional Description

The *g* command starts program execution. If the user does not specify the starting address adr, execution starts at the current value of the *CPC* register. This command must only be used once after downloading a new program. Use the *c* command to continue execution after a breakpoint.

If the user specifies the -c option, the Monitor passes all arguments after -c to the client program by the following method. If -c is specified, the Monitor places the number of arguments (argc) in Register a0. The Monitor also places the address of an array of pointers to the command argument-strings in Register a1. The first element in the array pointed to by a1 contains a pointer to the string -c (the option on the command line). Note that if the -c is not specified, register a0 will be set to zero. The function declaration of the function called is thus:

```
FUNC(int argc, char **argv, char **envp);
```

If *adr* is specified, a temporary breakpoint (*bptadr*) may also be specified. The temporary breakpoint remains in effect only until the next time that program execution is halted. The character '.' may be used as a placeholder for the *adr* if you wish to specifiy a temporary breakpoint without specifying a start address.

If you use the standard start-up file for executing programs under the control of the PROM Monitor, the function main will receive three incoming arguments, argc, argv and envp, having the correct values to permit a program to read options from the command line, where argv[0] is a pointer to the string -c.

The -e option causes 'go' to examine the embedded execution address of an executable located at 'adr' and use that embedded address as the starting point. Such code must have a front-end that relocates the code before allowing the execution to begin at the new address.

Examples illustrating the use of the g command follow.

```
PMON> g                       Start executing at the current value of
                              the EPC register.

PMON> g a0020000              Start executing at 0xa0020000.

PMON> g a0020000 a0020008     Start executing at 0xa0020000 and
                              break at 0xa0020008.

PMON> g -e ff000100           Examine code at 'adr' (ff000100) and
                              relocate it to the embedded execution
                              address.
```

## See Also

*c* command

### 4.3.2 Display / Set Breakpoints (b)

The *b* command sets and displays breakpoints.

#### Format

The format for this command is:

```
b
b adr..
b adr -s str
```

where:

| | |
|---|---|
| adr | specifies an address for the breakpoint. Up to 32 breakpoints addresses can be set. |
| -s str | executes the command string when the breakpoint is hit. |

Invoking the *b* command with no options causes the Monitor to print a list of the current breakpoints.

#### Functional Description

The b command sets a breakpoint at the specified address or addresses. Multiple addresses may be specified. Specified addresses must be word-aligned.

The Monitor automatically assigns a number to each breakpoint. The Monitor allocates the lowest available breakpoint number from 0 to 31 to any new breakpoint.

The Monitor reports a new breakpoint's number immediately after the breakpoint is set (see the examples at the end of this subsection for illustration of this). The assigned numbers can be used in the db (Delete Breakpoint) command.

When a breakpoint is reached, the command list specified in the environment variable *brkcmd* is executed. The default setting for brkcmd is:

```
brkcmd = "l @pc 1"
```

This command "`l @pc 1`", specifies that when the breakpoint occurs, the Monitor will disassemble one line starting at the address of the program counter.

You can change the breakpoint command variable with the set command. For example, you can include additional monitor commands in the *brkcmd* variable. You must separate additional commands on the command line with a semicolon. For example, entering the following command lists one line after reaching a breakpoint, and then displays all the register values.

```
set brkcmd "l @epc 1;r *"
```

By default, breakpoints are cleared when the load command is executed. See the section on the load command later in this document for details on how to override automatic breakpoint clearing after a download operation.

Some examples illustrating the use of the b command follow.

| | |
|---|---|
| `PMON> b a002000c` | Set a breakpoint at 0xa002000c. |
| `Bpt 1 = a002000c` | |
| `PMON> b`<br>`Bpt 0 = 8002022c`<br>`Bpt 1 = a002000c` | Display all breakpoints. |
| `PMON> b 80021248 -s "r"` | Set a breakpoint at 0x80021248. Display registers when the breakpoint is encountered. |

### See Also

db and load commands.

### 4.3.3 Delete Breakpoint (db)

The *db* command deletes the specified breakpoints.

#### Format

The format for this command is:

```
db [numb|*]
```

where:

| | |
|---|---|
| numb | is the breakpoint number to be deleted. |
| * | deletes all breakpoints. |

Entering *db* without any parameters lists all existing breakpoints. Entering an asterisk ("*") instead of a breakpoint number deletes all the existing breakpoints.

#### Functional Description

The *db* command deletes one or more specified breakpoints.

Examples illustrating the use of the db command follow.

| | |
|---|---|
| `PMON> db 3` | Delete breakpoint 3. |
| `PMON> db 4 6` | Delete breakpoints 4 and 6. |
| `PMON> db *` | Delete all breakpoints. |
| `PMON> db`<br>`Bpt 0 = a002000c` | Display all breakpoints. |

#### See Also

*d* and *load* commands.

## 4.3.4 Single Step (t / to)

The t command performs a trace (single step) operation.

### Format

The format for this command is:

```
t [-vbci] [cnt]

to [-vbci] [cnt]
```

where:

| | |
|---|---|
| -v | lists each step (verbose). |
| -b | captures only branches. |
| -c | captures only function calls. |
| -i | stops on invalid program counter. |
| cnt | traces cnt instructions and stops. |

### Functional Description

The t command executes the instruction addressed by the current value of the *user PC* register.

The *to* command is similar to the t command, except that the to command treats an entire procedure as a single step. The command or commands that are executed on completion of the single step is determined by the value of the environment variable *brkcmd*.

### See Also:

set / unset command.

## 4.3.5 Back Trace (bt)

The *bt* command displays a function call backtrace.

### Format

The format for this command is:

```
bt [-v] [ cnt]
```

where:

| | |
|---|---|
| -v | specifies that each function’s stack frame base address and size should be displayed. |
| cnt | specifies the number of lines to be displayed. |

When invoking this command with no options, the *backtrace* displays the names and up to four arguments for each level of stack frame.

### Functional Description

The *bt* command displays a list of function calls, starting with the function in which the User Stack register currently lies, and finishing when a return address becomes ‘‘invalid’’. An address is deemed invalid if it does not lie within one of the ranges specified by the valid pc environment variable.

Each line of output gives the current position in a function, and up to four of its arguments. The arguments can only be retrieved if they are saved within the function prologue, and this is unlikely to be the case for assembler functions and optimized C code. If you want to be able to see the arguments to C functions, then compile your program with optimization disabled. If the -v option is given, then the command additionally displays the stack-frame base address and size for each function. It will also indicate the amount of dynamic stack space allocated using C’s alloca() function, or equivalent.

The output of this command is passed to the more command, letting the user view one screen full of output at a time. Optionally, the user can specify *cnt*, which limits the number of lines to that number. An example illustrating the use of the *bt* command follows.

Example illustrating the use of the bt command follows. (MIPS code)

```
PMON> c write+10
     write+0x0010  3c09a07f  lui       t1,0xa07f
PMON> bt
     write+0x0010  (0x00000001,0xa0030300,0x0000001c)
     flsbuf+0x0234 (0xa0030300,0xa0029030)
     printf+0x045c (0xa0025490,0xa0020000,0x000000001,0x00000010)
     main+0x0138   (0x00000001,0xa07ffffe0)
     _start+0x0040 ()
```

### See Also

See also the *more* command

## 4.3.6 Continue Execution (c)

The *c* command makes program execution continue after a breakpoint has stopped program execution.

### Format

The format for this command is:

```
c [bptadr]
```

where:

| | |
|---|---|
| bptadr | specifies a single breakpoint. The breakpoint is removed when execution halts at this specified address. |

Invoking the *c* command with no arguments causes the program execution to continue from the address specified in the *current pc, CPC* register.

### Functional Description

When the user enters the *c* command, program execution starts at the address pointed to by the *CPC* register's current value. Use the *g* command to start program execution from an address specified on the command line.

As an option, a single temporary breakpoint may be specified. The temporary breakpoint is removed when execution stops. The temporary breakpoint is removed if another breakpoint stops program execution first.

Examples of the c command follow.

| | |
|---|---|
| `PMON> c` | Continue execution until exit or a regular breakpoint is encountered. |
| `PMON> c a0020104` | Continue execution until 0xa0020104 or a regular breakpoint is encountered. |

### See Also

*g* command

## 4.3.7 Execute Subroutine (call)

The *call* command executes a subroutine.

### Format

The format for this command is:

```
call adr [-s str|val]..
```

where:

| | |
|---|---|
| adr | is the address of the subroutine to be executed. |
| -s *str* | is a string argument. |
| *val* | is a value to be passed. |

The *call* command calls a function using the standard C calling convention. The "-s *str*" and *val* options permit arguments to be passed to the subroutine.

### Functional Description

The *call* command executes a downloaded subroutine, using a normal function call instruction to pass control to the specified address. This does not affect the existing value of the saved registers. Instead the subroutine is called directly from within PMON code without restoring the saved registers. Control returns to PMON via the usual subroutine return mechanism.

If the user specifies arguments, these are passed using the standard C calling convention. If the "-s" option is specified, the following argument is assumed to be a string. In this case the address of the string is passed to the subroutine. If a numerical value is specified in place of the "-s", it will be evaluated according to the existing rules and passed to the function. Up to ten arguments may be passed.

This command is usually used to provide a method of displaying application-specific data structures. For example, if your application has a complex, linked-list data structure, you might find it helpful to add a function to your program that can display the structure. The *call* command can then be used to invoke this function from the PMON prompt at any time in the execution, even between two single-step operations.

Examples illustrating the use of the call command follow.

```
PMON> call prstat                      Call the function whos name is
                                       prstat.

PMON> call prrec a0020008              Call the function 'prrec' and
                                       pass it the value 0xa0020008 as
                                       the first argument.

PMON> call printf -s "hello world"     Call the function printf and
                                       pass it theaddress of the
                                       string "hello world".
```

# 4.4 Miscellaneous and Environment Control

## 4.4.1 Help (h)

The h command provides on-line help.

### Format

The format for this command is:

```
h [*|cmd-]
```

where:

| | |
|---|---|
| * | provides detailed help on all the commands. |
| cmd | is a command. The Monitor then provides help on the stated command. |

If the command is executed without any parameters, then the Monitor lists all the available commands.

### Functional Description

The *h* command provides on-line help. If issued without arguments, all commands are listed. If issued with one or more command names as an option, it produces more detailed help on those commands.

The "*" option produces detailed help on all the commands, using the more command to control output on the screen.

Examples illustrating the use of the h command follow.

```
PMON> h
       h  on-line help                   hi  display command history
       m  modify memory                   r  display/set register
       d  display memory                  l  list (disassemble) memory
    copy  copy memory                  fill  fill memory
  search  search memory                  tr  transparent mode
       g  start execution (go)            c  continue execution
       t  trace (single step)            to  trace (step over)
       b  set break point(s)             db  delete break point(s)
    load  load memory from hostport    dump  dump memory to hostport
     set  display/set variable         eset  edit variable(s)
   unset  unset variable(s)            date  get/set date and time
   flash  program flash memory         stty  set terminal options
   about  about PMON                    sym  define symbol
      ls  list symbols                flush  flush caches
      mt  memory test                  call  call function
      bt  stack backtrace                sh  command shell
    more  paginator                  reboot  reboot PMON
    boot  boot wrapper              netboot  load network file
    ping  ping remote host         scsiboot  boot from scsi
```

```
     pci  view PCIbus devices            setup  setup TVMExxxx hardware
      ip  view IP configuration           info  view board information
     i2c  I2C EEPROM access              bconf  BIST configuration
    ibit  perform build-in tests            ab  configure autoboot
      cs  calculate the checksum …       sleep  sleep for a given number of seconds
vmeslave  open a VME slave window    vmemaster  open a VME master window
      cm  pci config modify

PMON>

PMON> h stty
    stty  [device] [opts]                                  set terminal options
                       -v    list possible baud rates and terminal types
                       -a    list all settings
                   <baud>    set baud rate
                   <term>    set terminal type
                     sane    set sane settings
                    ixany    allow any char to restart output
                   -ixany    allow only <start> to restart output
                    ixoff    enable tandem mode
                   -ixoff    disable tandem mode
PMON>
```

### 4.4.2 About PMON (about)

The about command displays information of the TVME8240A-PMON project, how it came to be ported to the PowerPC platform and who contributed.

#### Format

The format for this command is:

```
PMON> about

PMON/2000 is a derivative work under the BSD Copyright. It is freely
redistributable under this generous copyright as long as all pre-
existing copyrights are retained.

This implementation was ported to the TVMExxxx PowerPC Board by Rainer Harland
at TEWS TECHNOLOGIES, Germany.
```

## 4.4.3 History (hi)

The *hi* command lists the command history.

### Format

The format for this command is:

```
hi [cnt]
```

where:

| | |
|---|---|
| cnt | is the number of commands to list. |

Entering the command with no parameters lists the last 200 executed command lines to the screen.

### Functional Description

The *hi* command shows the command history, together with the history number for each command, in reverse order (the last command entered is listed first; the first command entered is listed last). The command numbers are reset to zero each time the system is reset.

Entering the *hi* command with no arguments lists the last 200 commands. This option is useful for determining the history number for a particular command.

The user can page through the output of the *hi* command, one screen at a time.

The optional cnt parameter selects a set number of lines to be output. The history list is intentionally in the reverse order to that used in a C shell, so that the latest entry is displayed first. If a command line is identical to the previous command, it is not added to the command history.

Examples illustrating the use of the *hi* command follow.

```
PMON> hi 3    Display the three last commands.

14 hi 3
13 hi
12 l
```

### See Also

*sh* command, which maintains a command history.

## 4.4.4 Display / Set Environment Variable (set)

The set command sets and displays environment variables.

### Format

The format for this command is:

```
set [name [value]]
```

where:

| | |
|---|---|
| *name* | is the name of the environment variable to set. |
| *value* | is the string to which the environment variable is set. |

Entering the set command with no arguments displays all the current environment variables.

### Functional Description

The set command is used to set or display environment variable values, to copy the settings of environment variables and terminal options to NVRAM, and to specify a list of commands to be executed by the TVME8240A-PMON Monitor following reset.

In some cases, when the Monitor displays a variable's current value, the Monitor prints a list of allowed values enclosed in square brackets; in other cases, no list is shown. In general, when the value is a numeric value, or when the value has an unlimited range of possible values, no list is shown.

The set command does not evaluate the specified value or check the specified value against a list of allowed values. Value checking is only performed when a command uses a variable.

To set a variable to a multiple-word value, enclose the value in single or double quotation marks.

Examples illustrating the use of the set command follow.

```
PMON> set               Display all current values.
    ipaddr = 10.0.29.234
   dlproto = XonXoff     [none XonXoff EtxAck]
  hostport = tty0
      ulcr = crlf        [cr lf crlf]
   memsize = 64
  cpuclock = 300000000
  busclock = 100000000
   systype = TVME8240A
    brkcmd = "l -r @cpc 1"
    datasz = -b          [-b -h -w]
    dlecho = off         [off on lfeed]
     bootp = no          [no sec pri save]
   inalpha = hex         [hex symbol]
    inbase = 16          [auto 8 10 16]
    moresz = 10
    prompt = "PMON> "
  regstyle = sw          [hw sw]
    rptcmd = trace       [off on trace]
   trabort = ^K
     uleof = %
   validpc = "_ftext etext"
   heaptop = 00100000
   showsym = yes         [no yes]
     fpfmt = both        [both double single none]
     fpdis = yes         [no yes]

     PMON> set moresz         Display current moresz.

     moresz = 10


     PMON> set moresz 20      Set moresz to 20 decimal.
```

Display instruction at current pc and display all general-purpose registers:

```
PMON> set brkcmd "l @cpc 1;r"
```

Environment Variables and Default Values

| Environment Variable | Default Value | Options |
|---|---|---|
| bootp | no | [no\|sec\|pri\|save] |
| brkcmd | "l @cpc 1" | command list |
| busclock | 100000000 | external/bus clock |
| cpuclock | 300000000 | cpu pipeline clock |
| datasz | -b | [-b\|-h\|-w] |
| dlecho | off | [off\|on\|lfeed] |
| dlproto | EtxAck | [none\|XonXoff\|EtxAck] |
| ethaddr | target dependent | string |
| ipaddr | none | string |
| heaptop | target dependent | string |
| hostport | tty1 | tty0-1 |
| inalpha | hex | hex symbol |

| | | |
|---|---|---|
| inbase | 16 | [auto\|8\|10\|16] |
| memsize | target dependent | memory size in MBytes |
| moresz | 10 | 0-n |
| prompt | "PMON> " | string |
| regstyle | sw | [hw\|sw] |
| rptcmd | trace | [off\|on\|trace] |
| showsym | yes | [yes\|no] |
| trabort | ^K | char |
| ulcr | off | [off\|on] |
| uleof | off | string |
| gateway | NULL | string |
| vxWorks | NULL | string |
| validpc | "_ftext etext" | string |
| autoboot | NULL | string |
| bootdelay | NULL | string |

Environment variables can be set and displayed using the set command.

Brief descriptions of each of the variables follow, together with references to their complete descriptions.

| Variable | Description |
|---|---|
| brkcmd | This variable specifies a sequence of Monitor commands that are executed when a breakpoint halts program execution. See the b command. |
| busclock | Set to the value of the bus/timing clock frequency. Useful for programs that needs to know this for timing purposes. Not applicable to all targets. |
| cpuclock | Set to the pipleine clock frequency of the target processor. |
| datasz | This variable controls whether data is displayed in byte, half-word, or word groups. See the d command. |
| dlecho | This variable controls whether the target board echoes on downloads. An entire line can be echoed, a single line-feed character can be echoed, or there can be no echo at all. See the load command and the section on flow control. |
| dlproto | This variable selects the download protocol for transfers via RS-232C. The Monitor supports Xon/Xoff and EtxAck download protocols. See the load command and the section on flow control. |
| bootp | Enable bootp protocol for setting IP address and download/boot. |
| ethaddr | unused |

| | |
|---|---|
| ipaddr | This variable specifies the Internet Protocol address. See the section on downloading via Ethernet. |
| heaptop | This variable specifies the highest allowable address in the heap maintained by the PROM Monitor. See the load command. |
| hostport | This variable selects whether tty0, tty1, or ethernet is the host port. See the load command and the section on flow control. |
| inalpha | This variable selects whether strings starting with the ASCII characters a, b, c, d, e, and f are interpreted as symbols or hexadecimal numbers.<br>See the sh command. |
| inbase | This variable selects the default input base for numeric values. Users can input octal, decimal, or hexadecimal numbers by changing this variable.<br>See the sh command. |
| memsize | The probed total RAM memory size in MBytes. |
| moresz | This variable specifies how many lines to display during screen-at-a-time display. See the more command. |
| prompt | This variable defines the Monitor prompt. An example of using this command is when you need to set the prompt to "PMON> " for compatibility with a source-level debugger. To do this use the following command:<br>*PMON2000> set prompt "PMON> "*<br>This will set the prompt to "PMON> " (note the space) and save this new value in the non-volatile memory (if supported). |
| regstyle | This variable defines whether hardware or software names are displayed in the l command. See the l command. |
| rptcmd | When this variable is set to "on," the previous command is executed again when the user enters an empty line. See the sh command. |
| showsym | Show or don't show symbols when doing trace and disassembly. |
| trabort | This variable selects the character that terminates transparent mode and returns the Monitor to command mode. See the tr command. |
| ulcr | This variable defines whether there is a carriage return or both a carriage return and a linefeed character at the end of the line during dumps. See the dump command. |
| uleof | This variable specifies a string that is sent to the host after a dump to the target has completed. See the dump command. |
| validpc | This variable specifies the range of valid PC values during program tracing. See the trace command. |
| gateway | This variable specifies an IP address that is to be used for indirect access to a Network. The gateway must know how to forward packets from the local Network to the destination. |
| vxWorks | This string variable is passed to a VxWorks kernel at boot time. It contains host and target IP address, user name and other required parameters. See vendor's BSP / Wind River documentation for details. NOTE: Most PowerPC BSP's copy this string to LOW_RAM for use by the VxWorks kernel. |
| autoboot | This string is parsed by the command shell after initialization (if present) and each properly formed command is executed in turn. Commands are seperated by a semicolon (";"). When entering a complex string value, use quotemarks (" ") to enclose the entry. This allows embedded SPACE characters. |

| | |
|---|---|
| bootdelay | This string value must be a number, greater than zero (0), and less than some unreasonable value. It is used to provide a time delay, in seconds, before any 'autoboot' string is executed. Users may enter any character during the bootdelay time to abort the autoboot sequence. |
| sysfail | This variable controls the SYSFAIL VMEbus signal. If sysfail is set to “on”, the SYSFAIL signal is asserted. If sysfail is set to “off” the SYSFAIL signal is not asserted.<br>If this variable is omitted the SYSFAIL signal is asserted by default.<br>The current state of SYSFAIL is printed in the startup message of PMON. |
| ifconfig | This variable selects the Ethernet interface to be used. If ifconfig is set to ‘fxp0’ the front panel interface is selected. Setting ifconfig to ‘fxp1’ selects the P2 back I/O interface.<br>If this variable is omitted then the front panel interface ‘fxp0’ will be used. |
| port2 | This variable controls the mode of the programmable transceiver for serial 2. If port2 is set to ‘RS232’ the port operates as RS232 interface. If port2 is set to ‘RS422’ the port operates as RS422 interface.<br>If this variable is omitted serial 2 operates as RS232 interface. |

## 4.4.5 Set Terminal Parameters (stty)

The *stty* command displays and sets terminal options.

### Format

The format for this command is:

```
stty  [device][-av]  [baud]  [sane]  [term][ixany|-ixany]
[ixoff|-ixoff]
```

where:

| | |
|---|---|
| device | is either tty0 or tty1. The default is tty0. |
| -a | gives a long listing showing all current settings. |
| -v | displays the possible choices for baud rate and terminal type. |
| baud | sets the baud rate. |
| sane | resets terminal settings to the default. |
| term | sets the terminal emulation type. |
| ixany | allows any character to restart the output. |
| -ixany | allows only START to restart the output. |
| ixoff | enables the tandem mode. |
| -ixoff | disables the tandem mode. |

When invoking the *stty* command with no parameters, the Monitor displays the terminal type and baud rate for the tty0 port.

### Functional Description

The *stty* command displays and sets the terminal options, such as terminal emulation type, baud rate, and ioctl settings. First, to display the current terminal type, baud rate, and ioctl settings for tty0, enter:

```
PMON> stty –a
```

To display the same information for tty1, enter:

```
PMON> stty tty1 –a
```

To change the baud rate or terminal type for tty0, simply enter the new setting after stty. Precede the new setting with tty1 to change the settings for tty1.

Examples illustrating the use of this command follow.

```
PMON> stty tty1 115200               Set tty1 to 115200 baud.
```

## 4.4.6 Display / Set Date (date)

The *date* command displays or sets the date and time.

### Format

The format of this command is:

```
date [-x][ yyyymmddHHMM.SS]
```

where:

| | |
|---|---|
| yyyymmddHHMM.SS | is the new date and time. |
| -x | stop real time clock |

### Functional Description

The *date* command with no arguments displays the current date and time as stored in the board's battery-backed clock/calendar device.

If an argument is given, then this sets the current date and time.

The optional argument is a string of pairs of digits, with the following meaning:

| | |
|---|---|
| yyyy | year |
| mm | month (January = 01) |
| dd | day of month |
| HH | hour (24 hour clock) |
| MM | minute |
| .SS | seconds |

When setting the date and time, you only need to enter as much as needs changing, starting with the minutes, then hours, then day, etc. Any value which is omitted is unchanged, except for seconds, which will be set to zero if omitted.

Some examples of the date command follow.

```
PMON> date
Mon Oct 28 20:36:14 2002

PMON> date 200210281939.00
Mon Oct 28 19:39:00 2002
```

## 4.4.7 Write / Erase Flash Memory (flash)

The flash command provides for programming, erasing and copying flash memory areas.

### Format

The format for the command is:

```
flash [-q] [-e addr size] [addr size from_addr]
```

where:

| | |
|---|---|
| <none> | gives a list of available flash areas and types |
| -q | gives a list of available flash areas and types |
| -e addr size | erases the flash area specified by addr and size |
| addr size from_addr | writes the flash *size* bytes from memory specified by *from_addr* to the flash beginning at addr. |

Each area in the flash devices will be identified, and is dependant on the system implementation.

These areas are useable for storage or for bootable images. The -q (Query) option will list each system dependent flash ROM area. The base address and size is listed, along with each flash area configuration, sector size (if supported) and the actual Flash device Manufacturing ID value (if reported). Example:

```
PMON> flash -q
Available FLASH memory
  Start     Size   Width Sectorsize  Type
70000000 00800000  8*8   00004000   SST39VF1601
ffe00000 00200000  1*8   00010000   S29GL016A-R1
```

Program a new PMON image into the boot FLASH:

```
PMON> netboot -o 1000000 10.0.0.1:pmon
PMON> flash -e fff00000 80000
PMON> flash fff00000 80000 1010000
Programming FLASH. Done.
```

The -e (Erase) option code calculates the sector from the flash address you enter, and the number of sectors is the size divided by sectorsize (rounded up).

### See Also:

The boot and netboot command.

## 4.4.8 Transparent Mode (tr)

The *tr* command selects transparent mode.

### Format

The format for this command is:

```
tr
```

### Functional Description

The *tr* command selects transparent mode.

In transparent mode, the Monitor copies any characters typed on the keyboard to the selected port and then copies characters arriving at the selected port to the screen.

The environment variable *trabort* selects the character that terminates the transparent mode and returns the Monitor to the default command mode.

The environment variable *hostport* determines the default port for the *tr* command.

```
PMON> set hostport tty1

PMON> tr
Entering transparent mode, ^K to abort

PMON>
```

## 4.4.9 Flush the Caches (flush)

The flush command flushes the data and/or instruction cache.

### Format

The format for the flush command is:

```
flush [-di]
```

where:

| | |
|---|---|
| -d | flushes the data cache only |
| -i | flushes the instruction cache only |

Entering flush without any parameters flushes both caches.

### Functional Description

The flush command performs a hard flush of the data and/or instruction cache. All entries will be flushed, even those that had been locked.

## 4.4.10 The Command Shell (sh)

The *sh* command is an embedded command that executes the Monitor command typed following the prompt.

### Functional Description

The following syntactic rules apply to all command lines entered at the Monitor prompt.

- Multiple commands can appear on one line if each command is separated by a semicolon (;).
- Register names are replaced by their contents if the register name is prefixed with an "at" symbol (@).
- Symbol names are replaced by their value if the symbol name is prefixed with a dollar sign symbol ($).
- Control-S pauses the output stream.
- Control-Q restarts the output stream.
- Control-C aborts the current command.

The shell also maintains a command history. Previous command lines are recalled either with Emacs-like commands or with C Shell "!" notation. The following table lists the commands that are supported by the Monitor.

| Command | Action |
|---|---|
| ^P | Recall previous command |
| ^N | Recall next command |
| ^F | Move cursor once character to the right (forward) |
| ^B | Move the cursor one character to the left (back) |
| ^A | Move the cursor to the beginning of the line |
| ^E | Move the cursor to the end of the line |
| ^D | Delete character at cursor position |
| ^H | Delete character to the left of the cursor |
| !*str* | Recall and execute the last command that started with the string *str* |
| !*num* | Recall and execute command number *num* |
| !! | Recall and execute last command |
| +-/() | Algebraic operators |
| ^*addr* | Substitute with contents of address *addr* |
| *@name* | Substitute with contents of named register |
| $*name* | Substitute with value of symbol *name* |

| | |
|---|---|
| 0x*num* | Treat *num* as a hexadecimal number |
| 0o*num* | Treat *num* as an octal number |
| 0t*num* | Treat *num* as an decimal number |

## The inbase, inalpha, prompt, and rptcmd Variables

The following paragraphs describe the *inbase*, *inalpha*, *prompt*, and *rptcmd* environment variables:

The *inbase* variable selects the default input base for numeric values. A value of 8, 10, or 16 selects that base as the assumed default. If "auto" is specified, the base is determined according to the usual C language rules (0x = hex, leading 0 = octal, otherwise decimal).

If *inbase* is set to 8, 10, or 16, then values starting with zero through nine are assumed to be values in the specified base. If *inbase* is set to "auto", then values starting with zero are assumed to be octal, and numbers starting with one through nine are assumed to be decimal.

The following lists the rules that hold in setting the default numeric base.

| Inbase | Base |
|---|---|
| 0x | Hexadecimal |
| 0t | Decimal |
| 0o | Octal |
| [g-zG-Z@_.] | Symbol |
| $ | Symbol |
| @ | Register |

The *inalpha* variable selects whether arguments starting with a, b, c, d, e, or f are interpreted as symbols or as hexadecimal numbers.

Setting *inalpha* to "hex" causes the Monitor interpret the argument as a hexadecimal value, if possible. If the argument cannot be interpreted as a hexadecimal value, then the Monitor checks the symbol table to see if the argument is a known symbol.

Setting *inalpha* to "symbol" causes the Monitor to check the symbol table first.

It is also possible to specify values using simple expressions using the arithmetic operators +, -, *, and /. Expressions do not take spaces between the numerals and operators. For example,

```
PMON> b printf+4
```

Sets a breakpoint at (printf+4). Any combination of register names, symbols, and values may be used. The precedence order of operators is the same as that defined by the C language. Two examples showing the use of simple arithmetic operators follow:

```
PMON> ls -v start+0x240                    Show the actual address.
PMON> d map+0t10*4                         Dump memory at (map+(10*4)).
PMON> d @a0+0t56                           Dump memory at 56(a0)
PMON> d ^tcbchn                            Dump memory at contents of tcbchn
```

**prompt -** This variable specifies the command prompt string.

The meta character "!" is replaced by the current history number. For example,

```
PMON> set prompt "!> "

23> _
```

It is not possible to display system variables in the prompt.

**rptcmd -** When this environment variable is set to "on", the previous command is repeated when the user enters a blank line. When set to "trace", only trace commands (t or to) are repeated.

## See Also

hi (command history) and set (setup and display environment variables) commands.

## 4.4.11 Paginator (more)

The *more* command provides screen-at-a-time control for user input.

### Format

The more command is an embedded command and is not accessible to the user on the command line.

### Functional Description

The more command is not specified by the user on the command line, but is implicitly used by certain commands. After displaying the number of lines according to the value of the *moresz* environment variable, the more command displays the prompt "more-" Commands that use the more command include h, hi, d, l, search, and ls.

The user can enter the following commands at the "more-" prompt:

| Command | Action |
|---|---|
| Space | Print one more page |
| /str | Search forward for string str |
| n | Repeat last executed search |
| <CR> | Show next line |
| q | Quit from the more prompt and return to the monitor prompt |

The Variable *moresz* sets how many lines are displayed on one screen during screen-at-a-time output. If *moresz* is set to zero, the screen scrolls continuously. The ^S or ^Q control sequence must be used to pause the output, and the ^C control sequence must be used to terminate output.

For example, to set the default number of lines output by the more command to 12, enter:

```
PMON> set moresz 12
```

### See Also

set command for the setup of the environment variables.

### 4.4.12 Reboot PMON (reboot)

The reboot command attempts to restart the TVME8240A-PMON monitor (and any other code which runs before TVME8240A-PMON at bootstrap time) by jumping to `0xfff00100` - the PowerPC restart location.

If your system initialization depends on some device receiving a hardware reset, this may not work.

#### Format

The format for the reboot command is:

```
PMON> reboot
```

## 4.4.13 Calculate Checksum (cs)

Calculate a simple checksum of a given address range. The resulting checksum is displayed in the following format: Checksum = <hex value>, where the placeholder <hex value> is substituted by the calculated checksum in “0x…” representation (e.g. 0x12, 0x1234 or 0x12345678 depending on the checksum size).

The checksum is calculated by adding each element of the given address range. If a carry is generated, a one is added to the checksum variable. The type size of each element and the checksum variable depends on the option flags [-bhw]. If not specified, half words (16 bit) are used by default.

The “*to*” address value is aligned to the next higher 16-bit boundary (MVME162 Bug compatibility).

The checksum and address range calculation must be compatible to the MVME162 Bug Monitor cs command.

### Format

```
cs  [-bhw] from to
```

where:

| | |
|---|---|
| -b | size is byte (8 bit) |
| -h | size is half words (16 bit) |
| -w | size is word (32 bit) |
| from | start address of the address range |
| to | end address of the address range |
| -b | size is byte (8 bit) |

## 4.4.14 Delay Execution (sleep)

The sleep command delays the execution of following commands of <*value*> seconds.

### Format

```
sleep  value
```

## 4.4.15 BIST Configuration (bconf)

The *bconf* command configures the build-in self test (BIST) facility.

### Format

The format for this command is:

```
bconf [-iedvnq]
```

where:

| | |
|---|---|
| -i | Initialize the BIST feature with default values. All tests will be enabled for startup and initiated build in test. Display output is set to verbose. |
| -e | Enable BIST feature |
| -d | Disable BIST feature |
| -v | Set verbose display output |
| -n | Set normal display output |
| -q | Disable any display output (quiet mode) |

### Functional Description

The *bconf* command configures the BIST facility either by command options or by an interactive dialog if the command is called without options. The BIST configuration is stored in the NVRAM

The interactive dialog is divided into two sections. The first section let you configure the startup build-in test (SBIT). The second section is intended for the initiated build-in test (IBIT) configuration. Each question can be answered by “Y” or “N” (not case sensitive) or simply by pressing the return key if the current setting (default value) is suitable. The dialog can be aborted by pressing the “.” key.

### Example

Enable BIST feature.

```
PMON> bconf -e
NVRAM updated...
```

Disable BIST feature

```
PMON> bconf -d
NVRAM updated...
```

Initialize and configure BIST feature

```
PMON> bconf -i
NVRAM updated...

PMON> bconf
```

```
SBIT (startup BIT) configuration:
Enable NVRAM test (Y/N) = Y ?
Enable DRAM test (Y/N) = Y ?
Full DRAM test (Y/N) = Y ?
Enable EEPROM test (Y/N) = Y ?
Enable FLASH test (Y/N) = Y ?
Enable IPAC test (Y/N) = Y ?
Enable UART test (Y/N) = Y ?
Enable Network test (Y/N) = Y ?
Network test timeout (sec) [20] ?
Enable SCSI test (Y/N) = Y ? n
Enable PCI test (Y/N) = Y ?
Enable RTC test (Y/N) = Y ? n

IBIT (initiated BIT) configuration:
Enable NVRAM test (Y/N) = Y ?
Enable DRAM test (Y/N) = Y ?
Full DRAM test (Y/N) = Y ?
Enable EEPROM test (Y/N) = Y ?
Enable FLASH test (Y/N) = Y ?
Enable IPAC test (Y/N) = Y ?
Enable UART test (Y/N) = Y ?
Enable Network test (Y/N) = Y ?
Network test timeout (sec) [20] ?
Enable SCSI test (Y/N) = Y ? n
Enable PCI test (Y/N) = Y ?
Enable RTC test (Y/N) = Y ? n
Display messages during execution (Y/N) = Y ?
Verbose messages (Y/N) = Y ? n
Enable BIST (Y/N) = Y ?
Save configuration in NVRAM (Y/N) = Y ?
NVRAM updated...
PMON>
```

The parameter “Network test timeout” controls the Ethernet link test. If this parameter is 0 the link test will be omitted and the test result is SUCCESS even if there is no cable connected. If this parameter is not 0 the full test will be executed.

### 4.4.16 Initiated Build-in Test (ibit)

The initiated or command build-in test (IBIT) command executes build-in tests specified by the optional command line arguments. If IBIT is entered without [args] the configured (*bconf*) IBIT tests will be executed.

#### Format

The format for this command is:

```
ibit [-v] [args]...
```

where:

| | |
|---|---|
| -v | Enable verbose mode |
| args | Optional argument to select one or more tests for execution. Keywords for possible tests are listed below |

| | |
|---|---|
| DRAM | Check DRAM; data, address lines and cells |
| NVRAM | Check entire NVRAM (see DRAM). Check battery status. |
| EEPROM | CRC test |
| FLASH | Read and display FLASH ID (read/write sequence) |
| IPAC | Check PLD status. If slot occupied, read IDPROM and check CRC |
| UART | Implicitly checked by the serial device driver. |
| Ethernet | Check for link and determine speed |
| SCSI | No function |
| PCI | Check availability of necessary PCI devices |
| RTC | Check clock status |

Argument passing is not case sensitive. Arguments must be separated by spaces.

#### Functional Description

The *ibit* command executes build-in tests specified by command line arguments. If the optional *[args]* are omitted the complete IBIT test, configured by *bconf*, is executed.

Depending on the configured display mode the result output contains one or more lines with the test result and additional information.

The test result can be:

| | |
|---|---|
| PASSED | Test successful completed. |
| SKIPPED | Test was skipped. No result is available. |
| FAILED | Test failed. |

## Example

Execute selected build-in tests:

```
PMON> ibit DRAM
Test DRAM     >>>  ...............................
Test DRAM     >>>  PASSED

PMON> ibit -v IPAC Ethernet PCI
Test IPAC     >>>  PASSED
Test IP-A     >>>  SKIPPED
Test IP-B     >>>  PASSED
Test IP-B     >>>  <Vendor=0xb3, Model=0x1c, CRC=OKAY>
Test IP-C     >>>  SKIPPED
Test IP-D     >>>  SKIPPED
Test Ethernet >>>  PASSED
Test Ethernet >>>  <Link okay @ 100 Mbps>
Test PCI      >>>  PASSED
```

Execute configured IBIT test:

```
PMON> ibit -v
===========================================================
Test NVRAM    >>>  PASSED
Test DRAM     >>>  ...............................
Test DRAM     >>>  PASSED
Test EEPROM   >>>  PASSED
Test FLASH    >>>  PASSED
Test IPAC     >>>  PASSED
Test IP-B     >>>  PASSED
Test UART     >>>  PASSED
Test Ethernet >>>  PASSED
Test PCI      >>>  PASSED
-----------------------------------------------------------
Test TOTAL    >>>  PASSED
===========================================================
PMON>
```

# 4.5 Diagnostics

## 4.5.1 Memory Test (mt)

The *mt* command executes the memory test.

### Format

The format for this command is:

```
mt [-c] [[ addr] size]
```

where:

| | |
|---|---|
| -c | implements a continuous memory test. |
| addr | is the base address from which to perform the memory test. |
| size | is the number of bytes, in hexadecimal, on which to execute the memory test. |

Entering this command with no parameters tests all available (non PMON-resident) memory.

### Functional Description

The *mt* command tests the available memory. By default, this command tests the memory at the first location after TVME8240A-PMON text to the last location of free RAM below the TVME8240A-PMON data segment.

If size is specified, then only that number of bytes is tested. If *addr* is also specified, then testing starts at the specified address. Both *addr* and *size* are rounded down to the nearest word address. If the user specifies a size of 0 (zero), the test executes on the entire available memory and does not terminate.

The mt memory test is not an exhaustive test. In the *mt* test, a single ‘‘walking one’’ is written to each word and cleared in turn. Then, to test other bits in the word, each word is loaded with its own address and then read back. Because this test writes an exclusive value to every word, it is sufficient to find most stuck-at faults and shorts. However, this test is not adequate to find pattern sensitivity and DRAM decay/leakage faults.

Examples illustrating the use of the *mt* command follow.

```
PMON> mt                        Test from just above TVME8240A-PMON text
                                to data-1.
PMON> mt 00020000               Test 8 Kbytes starting at 0x00020000
PMON> mt 00030000 4000          Test 16 Kbytes starting at 0x00030000
```

## 4.5.2 Network PING (ping)

The ping command ‘‘bounces’’ a packet to and from a specified network host.

### Format

The format for this command is:

```
ping [-nqv] [-i wait]] [-s size] [-l preload] host
```

where:

| | |
|---|---|
| -i wait | Wait wait in seconds between sending each packet. The default is to wait for one second between each packet. |
| -l preload | If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior. |
| -n | No attempt will be made to lookup symbolic names for host addresses. |
| -q | Nothing is displayed except the summary lines at startup time and when finished. |
| -s size | ICMP packets other than ECHO_RESPONSE that are received are listed. ‘‘Echo Replies’’ are displayed symbolically. |
| -v | ICMP packets other than ECHO_RESPONSE that are received are listed. ‘‘Echo Replies’’ are displayed symbolically. |

### Functional Description

The *ping* command is used to verify ether net network connections and setup.

It makes use of a feature of the ICMP protocol, which is used by hosts and gateways for low-level administrative chores. Each ICMP host is required to respond to an ECHO_REQUEST datagram with an ECHO_RESPONSE. ECHO_REQUEST datagrams (‘‘pings’’) have an IP and ICMP header, followed by a time and then an arbitrary number of ‘‘pad’’ bytes used to fill out the packet. The command continues pinging until interrupted by a Control-C.

When using ping for fault isolation, start by pinging `127.0.0.1` (a universal self-address, by internet convention.) This verifies that at least the onboard setup is workable. Then, hosts and gateways further and further away should be ‘‘pinged’’. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the program is terminated by a Control-C a brief summary is displayed.

Ping will report duplicate and damaged packets. Duplicate packets ‘‘should never happen’’: they’d have to be gateway problems. Tell your network manager.

Damaged packets (data doesn’t look like it should) are serious cause for alarm and often indicate broken hardware somewhere in the ping packet’s path (in the network or in the hosts).

## 4.5.3 View PCI Devices (pci)

The *pci* command displays information of PCI devices on the screen.

### Format

The format for this command is:

```
pci [-bcv]
```

where:

| | |
|---|---|
| -b | displays PCI devices plugged at a PMCSPAN (PCI-PCI bridge). |
| -c | displays a dump of the configuration space of the specified device (interactive) on the screen. |
| -v | displays additional PCI header information |

Entering this command with no parameters displays brief PCI device information on the screen.

```
PMON> pci
==>  bus = 0   device = 13   function = 0
Newbridge Universe VME (bridge, miscellaneous)

==>  bus = 0   device = 14   function = 0
Intel 82559ER (network, ethernet)

==>  bus = 0   device = 16   function = 0
PLX Technology, product: 0x9030 (bridge, miscellaneous)

==>  bus = 0   device = 17   function = 0
Intel 82559ER (network, ethernet)
```

Verbose output looks like this:

```
PMON> pci -v
==>  bus = 0   device = 13   function = 0
Newbridge Universe VME (bridge, miscellaneous)

Vendor ID      = 10e3h
Device ID      = 0h
Class Code     = 6800002h
Revision ID    = 2h
Status Reg     = 200h
Command Reg    = 147h
Header type    = 0h Single-function
BIST           = 0h Build-in-self-test not supported
```

```
Latency Timer  = 0x88
Cache Line Size= 0x0
Base Address   =  IO@ 00BFF000h length 4096 bytes
               = MEM@ 8FFBF000h,32bit length 4096 bytes
Max Lat        = 0ns
Min Gnt        = 3ns
PCI Int Pin    = 1
PCI Int Pin    = INT A
Interrupt line = 0

PCI Header:
 00 : 00 00 10 E3 02 00 01 47 06 80 00 02 00 00 88 00
 10 : 00 BF F0 01 8F FB F0 00 00 00 00 00 00 00 00 00
 20 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 30 : 00 00 00 00 00 00 00 00 00 00 00 00 00 03 01 00

==>  bus = 0   device = 16   function = 0
PLX Technology, product: 0x9030 (bridge, miscellaneous)

Vendor ID      = 10b5h
Device ID      = 9030h
Class Code     = 6800000h
Revision ID    = 0h
Status Reg     = 280h
Command Reg    = 142h
Header type    = 0h Single-function
BIST           = 0h Build-in-self-test not supported
Latency Timer  = 0x0
Cache Line Size= 0x0
Base Address   = MEM@ F5002000h,32bit length 128 bytes
               =  IO@ 00000000h length 128 bytes
               = MEM@ F5001000h,32bit length 256 bytes
               = MEM@ F5000000h,32bit length 1024 bytes
               = MEM@ F2000000h,32bit length 33554432 bytes
               = MEM@ F4000000h,32bit length 16777216 bytes
Max Lat        = 0ns
Min Gnt        = 0ns
PCI Int Pin    = 1
PCI Int Pin    = INT A
Interrupt line = 0
```

## 4.5.4 Setup TVME8240A hardware (setup)

The *setup* command is used to configure PCI devices after manufacturing of the TVME8240A. This command was implemented for internal use by TEWS TECHNOLOGIES only.

### Format

The format for this command is:

```
setup [-deprx]
```

where:

| | |
|---|---|
| -d | program Intel 82551ER SROM (select the interface with -0 or -1) |
| -0 | selects interface fxp0 (front panel) |
| -1 | selects interface fxp1 (P2 back I/O) |
| -e | erases the EEPROM of the PLX9030 PCI interface controller of the IP bridge |
| -p | program the EEPROM of the PLX9030 PCI interface controller |
| -r | reads and display the contents of the PLX9030 PCI interface controller |
| -x | Read contents of Intel 82551ER SROM (select the interface with -0 or -1) |

```
PMON> setup -r

PLX9030 EEPROM contents...

 00 : 9030 10B5 0280 0000 0680 0000 2030 1498
 10 : 0000 0040 0000 0100 4801 4801 0000 0000
 20 : 0000 4C06 0000 0003 0FFF FF00 0FFF F800
 30 : 0C00 0000 0F00 0000 0000 0000 0800 0001
 40 : 0400 0001 0000 0001 0C00 0001 0000 0000
 50 : D541 60A0 1581 20A2 1581 20A2 1501 20A2
 60 : 0000 0000 0800 0081 0400 0401 0200 0001
 70 : 0C80 0001 0030 0049 007A 4000 0224 9252
 80 : 0000 0000 0000 0000 FFFF FFFF FFFF FFFF
 90 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 A0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 B0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 C0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 D0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 E0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
 F0 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF

PMON>
```

## 4.5.5 View IPAC Configuration (ip)

The ip command can be used to display information of plugged IPAC modules at the four IPAC slots. In addition to IPAC ID PROM information this command displays base addresses of supported IO and MEM spaces. If no valid IPAC module was found at a certain slot “+++ EMPTY +++” appears in the display section for that slot.

### Format

The format for this command is:

```
ip [-v]
```

where:

| | |
|---|---|
| -v | scan also VMEbus A16/D16 address space for IPAC modules. |

```
PMON> ip
=====================     Local IPAC Carrier     =====================
 IP Slot A :   ++++  EMPTY  ++++
======================================================================
 IP Slot B :   ++++  EMPTY  ++++
======================================================================
 IP Slot C :   Manufacturer ID : 0xB3 (TEWS TECHNOLOGIES)
               Model Number    : 0x1C
               No IPAC IO Space recognized
               IPAC Memory @ F5000000h
               ID-PROM CRC OKAY

 F3000280 : 01 49 01 50 01 41 01 43 01 B3 01 1C 01 10 01 00
 F3000290 : 01 00 01 00 01 0D 01 D4 01 0A 01 00 01 00 01 00
======================================================================
 IP Slot D :   ++++  EMPTY  ++++
======================================================================

PMON>
```

## 4.5.6 View Board Mappings (info)

The info command displays mapping information of all relevant PCIbus, VMEbus and local spaces on the screen. This is only a help facility with informational character.

The output of the info command depends on the board version of the TVME8240A.

### Format

The format for this command is:

```
PMON> info
Table of TVME8240A Address Spaces
00000000 - 03FFFFFF    : SDRAM Memory
80000000 - FCEFFFFF    : PCI Memory Space
FCF00000 - FCFFFFFF    : Embedded Utilities Memory Block (EUMB)
FE000000 - FEBFFFFF    : PCI I/O Space
70000000 - 707FFFFF    : 64-Bit Memory FLASH
FF800000 - FF807F00    : NVRAM
FFE00000 - FFFFFFFF    : 8-Bit Boot FLASH

PCI Address Translation
CPU Address View          PCI Bus Address View
80000000 - FCEFFFFF  <=>  80000000 - FCEFFFFF    : PCI Memory Space
FE000000 - FEBFFFFF  <=>  00000000 - 00BFFFFF    : PCI I/O Space

F3001000 - F30010FF    : IPAC Interface Control Register
F3000000 - F30007FF    : IPAC ID/IO Spaces
F4000000 - F7FFFFFF    : IPAC MEM Spaces
F2000000 - F2FFFFFF    : IPAC MEM Spaces (8-Bit)

CPU Address View          VME Bus Address View
F1FF0000 - F1FFFFFF  <=>      0000 -     FFFF    : VMEbus A16/D16
F0000000 - F0FFFFFF  <=>    000000 -   FFFFFF    : VMEbus A24/D16
A0000000 - AFFFFFFF  <=>  00000000 - 0FFFFFFF    : VMEbus A32/D32
```

## 4.5.7 Access I2C EERPOM (I2C)

The *i2c* command manipulates and displays the contents of the I2C EEPROM.

### Format

The format for this command is:

```
i2c [-edprw][offs value]
```

where:

| | |
|---|---|
| -e | erases the entire I2C EEPROM (requires password) |
| -d | displays current board information data |
| -p | program new board information data in interactive mode (requires password) |
| -r | reads and displays the contents (256 byte) of the I2C EEPROM on the screen |
| -w offs value | writes a new byte value at the specified location (offs) into the EEPROM. |

The following example illustrates the usage of the i2c command:

```
PMON> i2c -d
Board Information Data in EEPROM...

TEWS TECHNOLOGIES TVME8240A-12 V1.0

PMON> i2c -w f0 cd
Enter password :

PMON> i2c -r
EEPROM contents...
 00 : 9D 06 20 30 00 0C 01 00 FF FF FF FF FF FF FF FF
 10 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 20 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 30 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 40 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 50 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 60 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 70 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 80 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 90 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 A0 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 B0 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 C0 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 D0 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
E0 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
F0 : CD FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

### 4.5.8 Open VME Slave Window (vmeslave)

This command opens a Universe II VME bus slave window to map local resources to the VME bus. The Address Modifier (AM) attributes can be set by several command options (see below).

If this command is entered only with the command option "*-w win*" then the specified window is disabled.

The *vmeslave* command doesn't check if the window is already opened by a previous call, rather it overwrites the current mapping.

The address modifier A16 is only allowed for window 0 and 4!

**Format+**

```
vmeslave  -w win [[-snpd] -a space [-l Kbyte] phyaddr vmeaddr]
```

where:

| | |
|---|---|
| -s | The window is set to supervisor AM code. |
| -n | The window is set to non-privileged AM code (default). |
| -p | The window is set to program AM code. |
| -d | The window is set to data AM code (default). |
| -w win | Selects the Universe slave window to be used. Possible values for *win* are 0 to 7. All windows are available at PMON startup. |
| - a space | Is the VME bus address space. Possible values are 32, 24 and 16 (VME bus A32, A24 and A16). |
| -l Kbyte | Is the length of the area to map in Kbytes (1024 Byte). If omitted the minimum size (resolution) of the window is used (window 0, 5 have 4 Kbyte resolution, the other have 64 Kbyte resolution). |
| phyaddr | Is the physical base address of the CPU address area that shall be mapped to the VMEbus. Possible areas are: DRAM, PCI MEM and IO and ROM/FLASH areas.<br>Every local address (00000000…FFFFFFFF) can be mapped to the VMEbus. |
| vmeaddr | Is the VME bus base address where the local address area shall appear. For VME bus A24 spaces enter 24-bits values (e.g. D00000), for A16 spaces enter 16-bit values (e.g. 6000 for FFFF6000). |

The following example maps the entire DRAM space (64 MB) to the VME A32 address 0xC0000000:

```
PMON> vmeslave -w 0 -a 32 -l 65536 0 c0000000
Local 00000000 successfully mapped to A32 VME C0000000 via window 0
```

The following example maps the ID/IO space of the local IPAC carrier to VME A16 address 0x8000

```
PMON> vmeslave -w 4 -a 16 -l 4 f3000000 8000
Local F3000000 successfully mapped to A16 VME 00008000 via window 4
```

### 4.5.9 Open VME Master Window (vmemaster)

This command opens a Universe II VMEbus master window to map a part of the VMEbus address space to local addresses. The Address Modifier (AM) attributes can be set by several command options (see below).

If this command is entered only with the command option "*-w win*" then the specified window is disabled.

The *vmemaster* command doesn't check if the window is already opened by a previous call, rather it overwrites the current mapping.

#### Format

```
vmemaster  -w win [[-sp] -a space -d width[-l Kbyte]
phyaddr vmeaddr]
```

where:

| | |
|---|---|
| -s | The window is set to supervisor AM code. If omitted non-privileged (user) AM code is used (default) |
| -p | The window is set to program AM code. If omitted data AM code is used (default) |
| -b | Enable posted writes (disabled by default). |
| -w win | Selects the Universe master window to be used. Possible values for *win* are 0 to 7. The master windows 0, 1 and 2 are used for the standard VME mapping of PMON. |
| -a space | Is the VME bus address space. Possible values are 32, 24 and 16 (VME bus A32, A24 and A16). |
| -d width | Is the VME bus maximum data width. Possible value are 32, 16 and 8 (VME bus D32, D16 and D8) |
| -l Kbyte | Is the length of the area to map in Kbytes (1024 Byte). If omitted the minimum size (resolution) of the window is used (window 0, 5 have 4 Kbyte resolution, the other have 64 Kbyte resolution). |
| phyaddr | Is the PCI bus address (same as physical address, 1:1 mapping) where the mapped VME bus address area shall appear.<br>The PCI bus address range from 0xB0000000 to 0xEFFFFFFF is free for use. |
| vmeaddr | Is the VME bus base address of the area that shall be mapped to the PCI bus. For VME bus A24 spaces enter 24-bits values (e.g. D00000), for A16 spaces enter 16-bit values (e.g. 6000 for FFFF6000). |

The following example maps a 64 MB A32/D32 VME address space from 0xE0000000 to local address beginning at 0xC0000000.

```
PMON> vmemaster -w 3 -a 32 -d 32 -l 65536 c0000000 e0000000
Local C0000000 successfully mapped to A32/D32 VME E0000000 via window 3
```

Close VME master window 7.

```
PMON> vmemaster -w 7
VME master window 7 successfully closed
```

# 5 PMON Upgrade

The TVME8240A-PMON firmware can be upgraded either via TFTP download or download via a serial connection. The upgrade procedure requires a completed setup of the TVME8240A board as described in chapter 2.

## 5.1 Download Firmware via TFTP

Copy the PMON ELF image (pmon) into the TFTP directory of your TFTP server. Please adapt IP address and file path as necessary.

```
PMON> netboot -o 1000000 10.0.0.1:pmon
Loading file: 10.0.0.1:pmon (elf)
0x1010074/515600 + 0x108de84/43780(z) + 1251 syms|
Entry address is 00010074
```

After successful download the boot FLASH memory must erased and programmed as follows.

```
PMON> flash -e FFF00000 80000
PMON> flash FFF00000 80000 1010000
```

After successful programming the PMON must be restarted.

## 5.2 Download Firmware via Serial Connection

If no TFTP server is available, downloading can be done via a serial connection. If not already done, connect serial port 1 of the TVME8240A board to your host system and start an appropriate terminal program at the host side.

At the PMON command prompt enter the following command to enable XON/XOFF software handshake:

```
PMON> set dlproto xonxoff
PMON> set dlecho off
PMON> set hostport tty0
```

After invoking the load command, PMON is waiting for data at serial port 1 (tty0). Now start the text file transfer of pmon.s19.

```
PMON> load
Downloading from tty0, ^C to abort
```

After successful download the boot FLASH memory must erased and programmed as follows.

```
PMON> flash -e FFF00000 80000
PMON> flash FFF00000 80000 1000000
```

After successful programming the PMON must be restarted.

# 6 Initial PMON Firmware Programming

Programming the PMON firmware after manufacturing or after an unintentionally erasing of the PMON requires the following steps.

1. On the TVME8240A board set all DIP switch positions of the “Factory Switch” S3 to ON. The Factory Switch S3 is right beside the Reset Switch.

2. Put a helper board (TVME8240A, TVME8300 or TVME8400) on the left side into a VME rack (system controller). Put the TVME8240A board right beside the helper board into the VME rack. Be sure that the TVME8240A board can access a VME slave window (A32/D32) on the helper board.

3. At the PMON prompt of the helper board enter the following commands (adapt IP address and host path):
   ```
   PMON> vmeslave -w 1 -a 32 -l 512 2000000 fff00000
   PMON> netboot -o 1000000 10.0.0.1:boot/pmon/tvme8240a/pmon
   PMON> copy 1010000 2000000 100000
   ```

4. Press the reset switch on the TVME8240A. Now the board should boot over the VME bus with the PMON firmware provided by the helper board via a VME slave window.

5. At the PMON prompt of the TVME8240A enter the following commands to program a new PMON firmware (adapt IP address and host path):
   ```
   PMON> netboot -o 1000000 10.0.0.1:boot/pmon/tvme8240a/pmon
   PMON> flash -e fff00000 80000
   PMON> flash fff00000 80000 1010000
   ```

6. Shut down the system, remove the TVME8240A board and set all DIP switch positions of the “Factory Switch” S3 to OFF.

7. Completed!
   Now the TVME8240A should boot the PMON Firmware from FLASH.

Command example helper board:

```
* PMON 2000 Professional *
Version: 2.2.0. Build date: Mar 28 2008 10:54:55
This software may be redistributed under the BSD copyright.
TVME8240A-12 BSP Copyright 2002-2008, TEWS TECHNOLOGIES GmbH
CPU PowerPC MPC8245/603e @  300 MHz/100 MHz.
Memory size  64 MB.



PMON> vmeslave -w 1 -a 32 -l 512 2000000 fff00000
Local 02000000 successfully mapped to A32 VME FFF00000 via window 1
PMON> netboot -o 1000000 10.0.0.1:boot/pmon/tvme8240a/pmon
Loading file: 10.0.0.1:boot/pmon/tvme8240a/pmon (elf)
0x1010074/464344 + 0x108164c/44060(z) + 1293 syms|
Entry address is 00010074
PMON> copy 1010000 2000000 100000
PMON>
```

Command example TVME8240A:

```
* PMON 2000 Professional *
Version: 2.2.0. Build date: Mar 28 2008 10:54:55
This software may be redistributed under the BSD copyright.
TVME8240A-51 BSP Copyright 2002-2008, TEWS TECHNOLOGIES GmbH
CPU PowerPC MPC8245/603e @  300 MHz/100 MHz.
Memory size  64 MB.


PMON> netboot -o 1000000 10.0.0.1:boot/pmon/tvme8240a/pmon
Loading file: 10.0.0.1:boot/pmon/tvme8240a/pmon (elf)
0x1010074/464344 + 0x108164c/44060(z) + 1293 syms\
Entry address is 00010074
PMON> flash -e fff00000 80000
Erasing FLASH block 16 Done.
Erasing FLASH block 17 Done.
Erasing FLASH block 18 Done.
Erasing FLASH block 19 Done.
Erasing FLASH block 20 Done.
Erasing FLASH block 21 Done.
Erasing FLASH block 22 Done.
Erasing FLASH block 23 Done.
PMON> flash fff00000 80000 1010000
Programming FLASH. Done.
PMON>
```