



PowerBoot

Rel 3.0

**for PPMC-270/PPMC-270SL/
PPMC-275/PPMC-280**

Instruction Set

P/N 223382 Revision AD
July 2004

Copyright

The information in this publication is subject to change without notice. Force Computers reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance, or design.

Force Computers shall not be liable for technical or editorial errors or omissions contained herein, nor for indirect, special, incidental, or consequential damages resulting from the furnishing, performance, or use of this material. This information is provided “as is” and Force Computers expressly disclaims any and all warranties, express, implied, statutory, or otherwise, including without limitation, any express, statutory, or implied warranty of merchantability, fitness for a particular purpose, or non-infringement.

This publication contains information protected by copyright. This publication shall not be reproduced, transmitted, or stored in a retrieval system, nor its contents used for any purpose, without the prior written consent of Force Computers.

Force Computers assumes no responsibility for the use of any circuitry other than circuitry that is part of a product of Force Computers. Force Computers does not convey to the purchaser of the product described herein any license under the patent rights of Force Computers nor the rights of others.

Copyright © 2004 by Force Computers. All rights reserved.

The Force logo is a trademark of Force Computers.

IEEE is a registered trademark of the Institute for Electrical and Electronics Engineers, Inc.

PICMG, CompactPCI, and the CompactPCI logo are registered trademarks and the PICMG logo is a trademark of the PCI Industrial Computer Manufacturer's Group.

MS-DOS, Windows95, Windows98, Windows2000 and Windows NT are registered trademarks and the logos are a trademark of the Microsoft Corporation.

Intel and Pentium are registered trademarks and the Intel logo is a trademark of the Intel Corporation.

SPARC is a registered trademark and the SPARC logo is a trademark of SPARC International, Inc.

PowerPC is a registered trademark and the PowerPC logo is a trademark of International Business Machines Corporation.

Hard Hat Linux is a trademark of Monta Vista Software, Inc. Linux is a registered trademark of Linus Torvalds. Red Hat is a registered trademark of Red Hat, Inc.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.



World Wide Web: www.forcecomputers.com

24-hour access to on-line manuals, driver updates, and application notes is provided via SMART, our SolutionsPLUS customer support program that provides current technical and services information.

Headquarters

The Americas

Force Computers Inc.
4211 Starboard Drive
Fremont CA 94538

Tel.: +1 (510) 624-5300
Fax: +1 (510) 624-5301
Email: support@fci.com

Europe

Force Computers GmbH
Lilienthalstr. 15
D-85579 Neubiberg/München

Tel.: +49 (89) 608 14-0
Fax: +49 (89) 609 77 93
Email: support-de@fci.com

Asia

Force Computers Japan K.K.
Shibadaimon MF Bldg. 4F
Shiba Daimon 2-1-16
Minato-ku, Tokyo 105-0012

Tel.: +81 (03) 3437-3948
Fax: +81 (03) 3437-*3968
Email: support-de@fci.com

Contents

Using This Guide

Other Sources of Information

1 PowerBoot Instruction Set

PowerBoot Commands Overview	1-3
Specifying Addresses	1-6
Command Description Structure	1-6
BF – Filling Blocks	1-7
BM – Copying Blocks	1-8
BS – Searching Blocks	1-8
BT – Running Diagnostic Tests	1-9
BV – Verifying Blocks	1-11
FERASE – Erasing Flash Memory	1-12
FPROG – Programming Flash Memory	1-13
FREAD – Reading Flash Memory	1-14
FVERIFY – Verifying Flash Memory	1-15
FRCEEPROMWRITE – Writing to EEPROM Device	1-16
FRCEEPROMREAD – Reading the Contents of EEPROM Devices	1-17
GETMAC – Getting the MAC Address	1-17
GO – Starting Binary Images	1-18
GOLINUX – Loading Linux using RAMDISK	1-18

HELP – Displaying Help Message	1-19
M – Modifying the Memory	1-20
MD – Displaying Memory Contents	1-22
NETLOAD – Loading a Binary Image Through Ethernet	1-23
NETSAVE – Saving Data Through Network into File	1-24
SETMAC – Setting the MAC Address	1-25
VERSION – Displaying Version Information	1-26
MEMMAP – Displaying Memory Map of the System	1-27
BATCH – Storing/Executing a Set of Commands	1-27
BAUD – Changing Baudrate	1-28
CHANGE_BOOT – Changing Bootimage File name	1-28
LINUXCMDLINE – Editing Linux Command Line	1-29
FACTORY_DEFAULTS – Restoring Default Values	1-29

2 PowerBoot Commands

PowerBoot Commands	2-3
RESET – Restarting the Board	2-3
SETBOOT – Editing Auto Boot Parameters	2-5
Boot select [0=GO, 1=Net, 2=BOOTP, 3=FLASH, 4=BATCH] (1):	2-6
Select Serial Baud (600 1200 2400 4800 9600 19200 38400 57600 115200)[115200]:	2-6
Auto boot [0=disable, 1=enable], (0):	2-6
Auto boot delay [0..99s], (55):	2-6
Load address (01000000):	2-6
Boot address (01000000):	2-7
Boot USER_FLASH [1..4], (1):	2-7
On-Board Ethernet Port [0/1] (00000000):	2-7
RARP [1] or ARP [2] protocol, (1):	2-7
Server-IP# [aaa.bbb.ccc.ddd]:	2-7
Target-IP# [aaa.bbb.ccc.ddd]:	2-7
TFTP Boot file name:	2-8
Boot Image Type [Linux = 1 Others = 0]: (0):	2-8
Current Linux CMDLINE (“console=ttyS0,115200 root=/dev/ram mtdparts=0:3072k(ker- nel),22528k(Ramdisk),-(JFFS2)”) New Cmd-line:	2-8
TFTP Ramdisk file name : ramdisk.gz:	2-8
Ramdisk Load address (02000000):	2-8
Shared Memory Interface:	2-8
Handle EREADY pin:	2-9
Power On Test POT [1=disable, 0=enable], (0):	2-9
POT Results, Store Address (0x00004000):	2-9
Configurations for SETBOOT Parameters	2-10

Configurations for GO Boot Select Option:	2-10
Configurations for NET Boot Select Option:	2-10
Configurations for BOOTP Boot Select Option:	2-10
Configurations for FLASH Boot Select Option:	2-11
Configurations for BATCH Boot Select Option:	2-11
Persistent Memory	2-12
Enabling Persistent memory feature	2-12
After Hard Reset	2-13
Shared Memory Interface	2-14
Command for Accessing the PCI Bus	2-16
BUSSHOW – Displaying PCI Devices	2-16
PROBEPCI – Displaying detailed PCI Device Information	2-16
CONFIG_RD – Performing a PCI Bus Configuration Read Cycle	2-18
CONFIG_WR – Performing a PCI Bus Configuration Write Cycle	2-19
Commands for Programming and Verifying Flash Memory	2-20
Reprogramming Boot Flash	2-20
Programming User Flash	2-22

3 Getting Started with PowerBoot

Getting Started	3-3
Setting Up the System for PowerBoot	3-4
PowerBoot Features	3-5
PowerBoot Ethernet Support	3-5
Entering PowerBoot	3-6
Exiting PowerBoot	3-7
Getting Online Help	3-8
Command Line Requirements and Restrictions	3-10
Special Keys and Characters	3-11
Command Line Characteristics	3-12

Index

Product Error Report

Tables

PowerBoot Instruction Set

Table 1	Summary of PowerBoot Operations.	1-3
---------	--	-----

PowerBoot Commands

Table 2	Boot Flash Address Range	2-20
---------	------------------------------------	------

Getting Started with PowerBoot

Table 3	PowerBoot Command Line Requirements and Restrictions	3-10
Table 4	Special Keys and Characters for PowerBoot Operations	3-11

Figures

PowerBoot Commands

Figure 1	Shared Memory Interface Setup	2-15
----------	-------------------------------------	------

Using This Guide

This guide explains how to use the PowerBoot command line interface (CLI) on PPMC-2xx modules. The guide also details how to boot Linux on PPMC-2xx.

The guide details instructions for PowerBoot Release Version 3.0 for PPMC-2xx.

This guide provides the following information:



Chapter Number	Name of Chapter	Description
1	“PowerBoot Instruction Set” chapter	Details generic PowerBoot commands
2	“PowerBoot Commands” chapter	Details PowerBoot PPMC-2xx board specific commands
3	“Getting Started with PowerBoot” chapter	Details system set up for PowerBoot, features of PowerBoot, entering and exiting from PowerBoot

This guide is for designers, developers, and system integrators who are designing and building PPMC-2xx modules into application systems. Manufacturing and field technicians and support specialists can also use the information in this manual to help configure systems and diagnose problems.

The guide assumes the user is familiar with board/system design and its architecture.

Conventions

Notation	Description
	All numbers are decimal numbers except when used with the following notations:
0000.0000 ₁₆	Typical notation for hexadecimal numbers (digits are 0 through F), e.g. used for addresses and offsets. Note the dot marking the 4th (to its right) and 5th (to its left) digit.
0000 ₂	Same for binary numbers (digits are 0 and 1)
Bold	Character format used to emphasize a word
Courier	Character format used for on-screen output

Notation	Description
<i>Courier+Bold</i>	Character format used to characterize user input and to separate it from system output
<i>Italics</i>	Character format for references and for table and figure descriptions.
<text>	Typical notation used for variables and keys.
[text]	Typical notation for buttons.
Note:	No danger encountered. Pay attention to important information marked using this layout.
Caution 	Possibly dangerous situation; slight injuries to people or damage to objects possible.
Danger 	Dangerous situation; injuries to people or severe damage to objects possible.

Abbreviations

ARP	Address Resolution Protocol
CLI	Command Line Interface
CPU	Central processing unit
CRC	Cyclic Redundancy Checking
DRAM	Dynamic random access memory
I/O	Input/output
JFFS2	Journalling Flash File System Version 2
MAC	Media Access Control
MMU	Memory Management Unit
NFS	Network File System
NVRAM	Nonvolatile random access memory
PCI	Peripheral Component Interconnect (bus)
PMC	PCI mezzanine card

POT	Power-on test
PPC	PowerPC
RAM	Random access memory
RARP	Reverse Address Resolution Protocol
SDRAM	Synchronous dynamic random access memory
SHM	Shared Memory
TFTP	Trivial File Transfer Protocol

Revision History

Order Number	Revision	Date	Description
223382 410 000	AA	June 2004	Release 3.0
223382 410 000	AB	June 2004	Release 3.0 Updated “PowerBoot Commands”, see “Power On Test POT [1=disable, 0=enable], (0):” on page 2-9
223382 410 000	AC	June 2004	Release 3.0 Included “Shared Memory Interface:” on page 2-8 and “Handle EREADY pin:” on page 2-9. Updated “Power On Test POT [1=dis- able, 0=enable], (0):” on page 2-9 Updated “Specifying Addresses” on page 1-6
223382 410 000	AD	July 2004	Release 3.0 Updated the following chapters: • “PowerBoot Instruction Set” • “PowerBoot Commands” • “Getting Started with PowerBoot”

Other Sources of Information

For further information refer to the following documents:

Company	Web Address	Document
Force Computers	www.forcecomputers.com	The following documents are available via http://splus.forcecomputers.com/cgi-bin/user/account/services
VITA Standards Organization	www.vita.com	Processor PMC Standard for Processor PCI Mezzanine Cards, VITA32

1

PowerBoot Instruction Set

PowerBoot Commands Overview

PowerBoot is firmware that provides basic test and debug commands. It is stored in the on-board boot ROM or Flash.

The PowerBoot CLI consists of commands for managing the operation of the system, running diagnostics, and verifying the integrity of the system design. Refer to the “PowerBoot Instruction Set” chapter and the the “PowerBoot Commands” chapter of this manual for detailed generic and platform-specific commands.

Table 1 lists the types of operations that can be performed by using the PowerBoot commands.

Table 1: *Summary of PowerBoot Operations*

Operation	Command Index
Displaying Online Help	
Display online help	“HELP – Displaying Help Message” on page 1-19
Configuring Boot Parameters	
Configure boot parameters	“SETBOOT – Editing Auto Boot Parameters” on page 2-5
Monitoring and Managing PCI Devices	
Scan a PCI bus segment and display information about devices found	“BUSSHOW – Displaying PCI Devices” on page 2-16
Display detailed PCI device information	“PROBEPCI – Displaying detailed PCI Device Information” on page 2-16
Perform a PCI Bus configuration read cycle	“CONFIG_RD – Performing a PCI Bus Configuration Read Cycle” on page 2-18
Perform a PCI Bus configuration write cycle	“CONFIG_WR – Performing a PCI Bus Configuration Write Cycle” on page 2-19
Initiating a Soft Reset	
Initiate a soft reset	“RESET – Restarting the Board” on page 2-3
Managing and Displaying the Contents of Main Memory	
Display the contents of an area of memory	“MD – Displaying Memory Contents” on page 1-22
Fill an area of memory with constant values	“BF – Filling Blocks” on page 1-7

Table 1: *Summary of PowerBoot Operations (cont.)*

Operation	Command Index
Copy the contents of an area of memory to another area of memory	“BM – Copying Blocks” on page 1-8
Compare the contents of two memory locations	“BV – Verifying Blocks” on page 1-11
Search an area of memory for constant values	“BS – Searching Blocks” on page 1-8
Modify the contents of memory	“M – Modifying the Memory” on page 1-20
Test memory	“BT – Running Diagnostic Tests” on page 1-9
Managing Flash Memory Devices	
Erase the contents of a Flash memory device or an area of a Flash memory device	“FERASE – Erasing Flash Memory” on page 1-12
Program a Flash memory device	“FPROG – Programming Flash Memory” on page 1-13
Compare byte-wise contents of specified Flash bank with the contents of a specified memory location	“FVERIFY – Verifying Flash Memory” on page 1-15
Read contents of specified Flash device into RAM	“FREAD – Reading Flash Memory” on page 1-14
Setting Boot Parameters	
Change Bootimage file name	“CHANGE_BOOT – Changing Bootimage File name” on page 1-28
Edit Linux command line	“LINUXCMDLINE – Editing Linux Command Line” on page 1-29
Performing Network Operations	
Load a binary image into memory over the network	“NETLOAD – Loading a Binary Image Through Ethernet” on page 1-23
Save a memory area via network into a file:	“NETSAVE – Saving Data Through Network into File” on page 1-24
Starting Binary Images	
Start a binary image that resides in main memory	“GO – Starting Binary Images” on page 1-18

Table 1: *Summary of PowerBoot Operations (cont.)*

Operation	Command Index
Start the binary image from the two memory locations (Kernel image and the Ramdisk image)	“GOLINUX – Loading Linux using RAMDISK” on page 1-18
Setting Serial Port Parameters	
Change baudrate	“BAUD – Changing Baudrate” on page 1-28
Configuring On-board Ethernet Port’s MAC Address	
Set the MAC address for the two on-board Ethernet ports	“SETMAC – Setting the MAC Address” on page 1-25
Dump the MAC addresses for the two on-board Ethernet ports	“GETMAC – Getting the MAC Address” on page 1-17
Managing EEPROM Devices	
Write to EEPROM device	“FRCEEPROMWRITE – Writing to EEPROM Device” on page 1-16
Read the contents of EEPROM devices	“FRCEEPROMREAD – Reading the Contents of EEPROM Devices” on page 1-17
Utility Command	
Store/Execute a set of commands	“BATCH – Storing/Executing a Set of Commands” on page 1-27
Setting Default Parameters	
Restore default values	“FACTORY_DEFAULTS – Restoring Default Values” on page 1-29
Displaying Board Information	
Display the memory map of the system	“MEMMAP – Displaying Memory Map of the System” on page 1-27
Display version information	“VERSION – Displaying Version Information” on page 1-26

Most commands require arguments. Some options enable a finer level of control over the command’s execution. Options appear in syntax as keywords separated by a vertical bar (|).

Specifying Addresses

Hexadecimal addresses, e.g. 0000.4C00₁₆, can be entered in their complete form in the command line, i.e. 00004C00, or the preceding zeros can be left out, i.e. it is also possible to enter only 4C00.

Long-aligned addresses are addresses which are divisible by 4 without a remainder.

Note: The space from 0x000000 - 0x7FFFFFFF is strictly reserved for PowerBoot area. Address 0x200000 will contain PowerBoot specific data (Version/Builddate).

Command Description Structure

This manual provides command description structured as follows:

1. Short description of the command.
2. Syntax description consisting of:

COMMAND <parameter1> <parameter2> [parameter3]

The parameter given in brackets is optional. The following sections explain respective parameters.

3. Detailed description of the command completing the short command description given at the beginning of the respective section.
4. Example

BF – Filling Blocks

BF fills a specified RAM memory area with a byte, a word, or a long constant or with a user defined ASCII string.

SYNTAX: **BF** <begin> <end> <value>

begin specifies the starting address of the memory area to be filled

end specifies the end address of the memory area to be filled (exclusive of **end**)

value specifies the constant used for filling. It can be one of the following constants:

- byte value = a byte constant followed by the character B (e.g. A5 B)
- word value = a word constant followed by the character W (e.g. A5B6 W)
- long value = a long constant followed by the character L (e.g. A5B6C7D8 L)
- string value = an ASCII string followed by the character P, the ASCII string has to be set within double quotes (e.g. "Hello" P)

EXAMPLE: In the following example a 4-KByte memory beginning at 0100.0000₁₆ and ending at 0100.1000₁₆ is filled with the word constant A5A5₁₆. This word constant fills the 4-KByte memory.

```
PowerBoot>BF 1000000 1001000 A5A5 W
PowerBoot>
```

In the second example an 8-KByte memory beginning at 0100.4C00₁₆ and ending at 0100.6C00₁₆ is filled with the ASCII string "Hello":

```
PowerBoot>BF 1004C00 1006C00 "Hello" P
PowerBoot>
```

BM – Copying Blocks

BM copies the contents of a specified source memory area to a memory area starting at a specified destination address. However, if the source and the destination areas overlap, only the destination area will be complete.

SYNTAX: **BM** <begin> <end> <destination>

begin specifies the starting address of the source memory area to be copied

end specifies the end address of the source memory area to be copied (exclusive of **end**)

destination specifies the starting address of the destination memory area to where the first byte of the source memory area is copied

EXAMPLE: In the first example a 4-KByte memory is copied from address 0100.2D00₁₆ to address 0100.5E00₁₆:

```
PowerBoot>BM 1002D00 1003D00 1005E00
PowerBoot>
```

In the second example an 8-KByte memory is copied from address 0100.4C00₁₆ to address 0100.5C00₁₆. In this case, the areas overlap each other:

```
PowerBoot>BM 1004C00 1006C00 1005C00
PowerBoot>
```

BS – Searching Blocks

BS searches a memory area for a byte, a word, or a long constant or for a user defined ASCII string.

The two tasks performed by the **BS** command are:

- To display the locations where the searched byte, word, long constant, or ASCII string is found (use Syntax 1).
- To display the locations where the searched byte, word, long constant, or ASCII string is not found (use Syntax 2).

SYNTAX 1: **BS** <begin> <end> <value>

SYNTAX 2: **BS** <begin> <end> </value>

begin specifies the starting address of the memory area to be searched

end	specifies the end address of the memory area to be searched (exclusive of end)
value	specifies the constant to search for. It can be one of the following constants: <ul style="list-style-type: none"> – <i>byte value</i> = a byte constant followed by the character B (e.g. 3F B) – <i>word value</i> = a word constant followed by the character W (e.g. 3F3F W) – <i>long value</i> = a long constant followed by the character L (e.g. EFEFEFEF L) – <i>string value</i> = an ASCII string followed by the character P, the ASCII string has to be set within double quotes (e.g. "FORCE" P)

EXAMPLE: In this example a 4-KByte memory from 0000.2D00₁₆...0000.3D00₁₆ is searched for the byte constant 3F₁₆. The byte constant is found at addresses 0000.2E01₁₆ and 0000.2E05₁₆.

```
PowerBoot>BS 2D00 3D00 3F B
Search: 00002E01 = 3F
Search: 00002E05 = 3F
PowerBoot>
```

BT – Running Diagnostic Tests

BT executes a set of basic binary tests to locate memory errors.

SYNTAX: **BT** <begin> <end> [e]

begin	specifies the starting address of the memory area to be tested
end	specifies the end address of the memory area to be tested (exclusive of end)
e	executes the test in an endless loop

DESCRIPTION: **BT** executes the following test types:

Test name	Access size		
	8-bit	16-bit	32-bit
Checkerboard test (reverse)	x	x	x
Walking ones test (reverse)	x	x	x
Walking zeros test (reverse)	x	x	x
Increment test	x	x	x

Test name	Access size		
	8-bit	16-bit	32-bit
Prime test	—	—	x
March-b test	—	—	x

All tests, except the prime test, fill the memory area specified by **begin** and **end**. During this action, fill appears on the screen. After the memory area has been filled, the testing starts and is indicated by the following message:

- test forward, or
- test backward

where, test forward means that the memory is tested from **begin** to **end**, whereas test backward indicates that the memory is tested from **end** to **begin**. When the test has been finished, a “done” message appears on the screen.

If an error occurs during the test, the following error message will appear:

```
error at location 0x12345678 value found:0xAB should be:0xBA
```

EXAMPLE:

In the following example the memory area $0100.0000_{16} \dots 0101.0000_{16}$ is tested:

```
PowerBoot> _
PowerBoot> BT 1000000 1010000
Blocktest at single mode, testing 65536 bytes from:
0x1000000...0x100FFFF

Checkerboard test 8bit.....done
Checkerboard test reverse...done

Walking ones test 8bit.....done
Walking ones test reverse...done

Walking zeros test 8bit.....done
Walking zeros test reverse..done

Increment test 8bit.....done

Checkerboard test 16bit.....done
Checkerboard test reverse...done

Walking ones test 16bit.....done
Walking ones test reverse...done

Walking zeros test 16bit.....done
Walking zeros test reverse..done

Increment test 16bit.....done
```

```

Checkerboard test 32bit.....done
Checkerboard test reverse...done

Walking ones test 32bit.....done
Walking ones test reverse...done

Walking zeros test 32bit....done
Walking zeros test reverse..done

Increment test 32bit.....done

Prime test 32bit.....done
Prime test 32bit R/T.....done

March-b test .....done

--- Blocktest runs 1 times - 00000000 errors ---

PowerBoot>_

```

BV – Verifying Blocks

BV compares two memory areas on a byte-organized level.

SYNTAX: **BV** <begin> <end> <destination>

begin	specifies the starting address of the source memory area to be compared
end	specifies the end address of the source memory area to be compared (exclusive of end)
destination	specifies the starting address of the destination memory area, i.e. the first byte of the destination memory area which is compared with the source memory

DESCRIPTION: If the compared memory areas are not equal, the different values and the memory location will be displayed. Overlapping of memory areas is not allowed.

EXAMPLE: In the first example a 4-KByte memory from 0000.2D00₁₆...0000.3D00₁₆ is compared with a destination memory starting at address 0000.5E00₁₆:

```

PowerBoot>BV 2D00 3D00 5E00
PowerBoot>

```

In the second example an 8-KByte memory from 0000.4C00₁₆...0000.6C00₁₆ is compared with a destination memory starting at address 0000.7C00₁₆. The two

compared memory areas are not identical at the locations $0000.4C10_{16}$ and $0000.7C10_{16}$:

```
PowerBoot>BV 4C00 6C00 7C00
Verify: 00004C10 = EE 00007C10 = 3F
PowerBoot>
```

FERASE – Erasing Flash Memory

FERASE erases a whole Flash memory device or a specified area of the Flash memory device. The specified area must exactly match the page boundaries of the Flash memory device. If a User Flash consists, for example, of a 28F008 device (1 Mbit * 8 bit), the minimum size of the erasable area is 64 KByte, i.e. the size of one sector.

The two tasks performed by the **FERASE** command are:

- To erase a whole Flash memory device (use Syntax 1).
- To erase a specified area of a Flash memory device (use Syntax 2).

SYNTAX 1: **FERASE** <flashDevice>

SYNTAX 2: **FERASE** <flashDevice> <flashOffset> <length>

flashDevice specifies the name of the Flash memory device to be erased. If only **flashDevice** is defined, the whole Flash memory device will be erased. The following names are currently supported:

Name of Flash Memory Device	Description
BOOT_FLASH	Boot Flash
USER_FLASH1	First User Flash
USER_FLASH2	Second User Flash

flashOffset specifies the destination offset within the Flash memory device area to be erased. The offset must be sector aligned. Specify the offset at proper boundaries of 256 Kbytes in case of User Flash and 64 Kbytes for Boot Flash.

length specifies the number of bytes within the Flash memory device area to be erased. The length must be sector aligned. The Flash device can be erased only in valid block size of 256 Kbytes in case of User Flash and 64 Kbytes for Boot Flash.

Note: `USER_FLASH` is equivalent to `USER_FLASH1`.

Caution

Do not erase Flash memory and re-program it, unless certain. Faulty re-programming of Boot Flash will result in a damaged and inoperative board.

EXAMPLE:

When **FERASE** checks for Flash write protection, the following messages appear:

```
PowerBoot> FERASE user_flash1 0 40000
```

```
Are you sure you wish to erase the Flash:[yes = 1,no = 0] : 1
Erasing flash memory ... done.
```

```
/* Common Errors */
```

```
PowerBoot> FERASE user_flash1 0 7000
```

```
Are you sure you wish to erase the Flash:[yes = 1,no = 0] : 1
Offset and size parameters are not sector aligned!
```

```
PowerBoot> FERASE user_flash1 0 3000000
```

```
Flash Size is too large
```

```
PowerBoot>
```

```
PowerBoot> FVERIFY boot_flash 1000000 0 22AC0
```

```
Cannot verify for BOOT_FLASH.
```

FPROG – Programming Flash Memory

FPROG programs Flash memory devices.

The three tasks performed by the **FPROG** command are:

- To program a whole Flash memory device (use Syntax 1).
- To program the space between a specified offset and the end of the Flash memory device (use Syntax 2).
- To program a specified number of bytes starting at a specified destination offset (use Syntax 3).

SYNTAX 1: **FPROG** <flashDevice> <source>

SYNTAX 2: **FPROG** <flashDevice> <source> <flashOffset>

SYNTAX 3: `FPROG <flashDevice> <source> <flashOffset> <length>`

flashDevice specifies the name of the Flash memory device to be programmed. The following names are currently supported:

Name of Flash Memory Device	Description
BOOT_FLASH	Boot Flash
USER_FLASH1	First User Flash
USER_FLASH2	Second User Flash

source specifies the source address containing the data with which the Flash memory device is programmed

flashOffset specifies the destination offset within the Flash memory device

length specifies the number of bytes to be programmed

Caution



Do not erase Flash memory and re-program it, unless certain. Faulty re-programming of Boot Flash will result in a damaged and inoperative board.

EXAMPLE: In this example, the user reprograms the User Flash with the data stored at source address 0100.0000₁₆:

```
PowerBoot> FPROG user_flash1 1000000 0 40000
Programming flash memory
```

```
0 |#####| 100%
```

```
Done.
```

FREAD – Reading Flash Memory

FREAD reads the contents of Flash Memory into RAM.

SYNTAX: `FREAD <flashDevice> <destination> <flashOffset> <length>`

flashDevice specifies the name of the Flash memory device to be verified. The following names are currently supported:

Name of Flash Memory Device	Description
USER_FLASH1	First User Flash
USER_FLASH2	Second User Flash

destination specifies the memory address where the read bytes from Flash are to be stored

flashOffset specifies the destination offset within the Flash memory device

length specifies the number of byte to be verified

EXAMPLE: In this example the user reads the contents of the User Flash 1 memory into RAM:

```
PowerBoot> FREAD user_flash1 1000000 0 200
```

Here, **FREAD** reads contents of User Flash1

where,

user_flash1: User Flash bank 1

1000000 : Memory address where the read bytes of Flash are to be stored

0 : destination offset within the Flash memory device

200 : Length of image in User Flash to be read

FVERIFY – Verifying Flash Memory

FVERIFY compares byte-wise the contents of a specified Flash bank with the contents of a specified memory location.

SYNTAX: **FVERIFY** <flashDevice> <source> <flashOffset> <length>

flashDevice specifies the name of the Flash memory device to be verified. The following names are currently supported:

Name of Flash Memory Device	Description
USER_FLASH1	First User Flash
USER_FLASH2	Second User Flash

source specifies the source address containing the data with which the Flash memory device is verified

flashOffset specifies the destination offset within the Flash memory device

length specifies the number of bytes to be verified

EXAMPLE: In this example, the user wants to verify the User Flash 1 with the data stored at source address 0100.0000₁₆:

```
PowerBoot> FVERIFY user_flash1 1000000 0 1D3A9C
Verifying flash contents
```

```
0 |#####| 100%
```

Done.

```
PowerBoot>
```

Here, **FVERIFY** reads contents of User Flash1, offset 0 and compares it with data at 0x1000000, upto length 0x1D3A9C.

where,

```
user_flash1 : User Flash bank 1
1000000      : Start address in memory of image
0            : Offset into User Flash
1D3A9C       : Length of image in User Flash to be verified
```

FRCEEPROMWRITE – Writing to EEPROM Device

This command writes to the EEPROM Device.

SYNTAX: **FRCEEPROMWRITE** <deviceAddress> <offset> <numberOfBytes> <ramAddress>

deviceAddress specifies the address of EEPROM to be written

offset specifies the offset within the EEPROM in hexadecimal notation

numberOfBytes number of bytes to be written

ramAddress specifies the source address containing data with which the EEPROM device is to be programmed

EXAMPLE:

```
PowerBoot>FRCEEPROMWRITE a6 20 100 2000000
```

where,

```
a6          : The device address of EEPROM to be written
20          : Offset within the EERPOM in hexadecimal
100         : Number of bytes to be written in hexadecimal
```

2000000 : The source address containing data that is to be programmed into the EEPROM device

Note: In the device a8, the memory area containing the MAC address is protected from 1ff0 to 1fff.

FRCEEPROMREAD – Reading the Contents of EEPROM Devices

This command reads the contents of EEPROM devices.

SYNTAX: `FRCEEPROMREAD <deviceAddress> <offset> <numberOfBytes> <ramAddress>`

`deviceAddress` specifies the address of EEPROM to be read

`offset` specifies the offset within the EEPROM device (hexadecimal)

`numberOfBytes` Number of bytes to be read (hexadecimal)

`ramAddress` location in memory where the contents are to be placed

EXAMPLE: `PowerBoot>FRCEEPROMREAD a6 10 10 1000000`

where,

`a6` : The device address of EEPROM to be read

`10` : Offset within EEPROM device

`10` : Number of bytes to be read

`1000000` : Location in RAM memory where the read contents are to be placed

GETMAC – Getting the MAC Address

GETMAC gets the MAC address for the two on-board Ethernet ports.

SYNTAX: `GETMAC <ethInterface>`

EXAMPLE: `PowerBoot> GETMAC 0`

Current MAC for Eth0 : 10:20:0C:B9:1C:30

`PowerBoot> GETMAC 1`

Current MAC for Eth1 : 40:50:60:70:73:93

In this example, the **GETMAC** command gets the MAC address for the two on-board Ethernet ports where the addresses are:
10:20:0C:B9:1C:30 and 40:50:60:70:73:93

GO – Starting Binary Images

GO executes a binary image located in the DRAM/User Flash memory. After executing the **GO** command, PowerBoot relinquishes control. Cache and interrupts are disabled before executing the binary image.

SYNTAX: **GO** <address>

address specifies the starting address of the binary image

EXAMPLE: The following example starts a binary image at address 0100.0000₁₆:

```
PowerBoot> GO 1000000
```

GOLINUX – Loading Linux using RAMDISK

GOLINUX starts the binary image from the two memory locations, namely the Kernel Image, and the Ramdisk Image. After executing the **GOLINUX** command, PowerBoot relinquishes control. Cache and interrupts are disabled before executing the Linux image.

SYNTAX: **GOLINUX** <kernelStartAddress> <ramdiskStartAddress> <ramdiskEndAddress>

kernelStartAddress specifies the starting address of the downloaded kernel image in memory

ramdiskStartAddress specifies the starting address of the ramdisk image in memory

ramdiskEndAddress specifies the end address of the ramdisk image in memory

EXAMPLE: The following example starts a kernel binary image at the address 0100.0000₁₆. The RAMDISK start address is 0200.0000₁₆ and the RAMDISK end address is 0333.F394.

```
PowerBoot> GOLINUX 1000000 2000000 333F394
```

HELP – Displaying Help Message

HELP displays all commands provided by PowerBoot.

SYNTAX 1: **HELP**

DESCRIPTION: **HELP** command is used to display the help messages for the entire set of commands.

EXAMPLE: PowerBoot> **HELP**

--- Command words ---

BATCH - Stores/Executes a set of commands
BAUD - Changes Console Baudrate
BF - Fills Block of Memory Region
BM - Copy Block of Memory Region
BS - Searches a Pattern in a Block of Memory
BT - Tests the Specified Memory Region
BUSSHOW - Displays PCI Devices (PCI Scan)
BV - Verify Block of Memory Region
CHANGE_BOOT - Changes Bootimage File name in NVRAM
CONFIG_RD - Performs a PCI Bus Configuration Read Cycle
CONFIG_WR - Performs a PCI Bus Configuration Write Cycle
FACTORY_DEFAULTS - Sets Setboot parameters to factory defaults

Page 1 : Enter CR for more Commands

FERASE - Erases Boot/User Flash Memory
FPROG - Programs Boot/User Flash Memory
FRCEEPROMREAD - Reads data from EEPROM Device
FRCEEPROMWRITE - Writes data to EEPROM Device
FREAD - Reads User Flash Memory
FVERIFY - Verify Boot/User Flash Memory
GETMAC - Get the MAC Address
GO - Executes a Binary Images located in the Memory
GOLINUX - Loading Linux using RAMDISK
HELP - Displays Help Messages
LINUXCMDLINE - Edits Linux command line
M - Modifies the Memory Contents

Page 2 : Enter CR for more Commands

MD - Displays Memory Contents
MEMMAP - Displays Memory Map of the System
NETLOAD - Loads a Binary Image into memory through Network
NETSAVE - Saving Data through Network into File
PROBEPCI - Displays PCI Devices
RESET - Does a warm reboot

SETBOOT - Edits Auto Boot Parameters
SETMAC - Set the MAC Address
VERSION - Displays version Information

"HELP <command>": Displays Syntax and Usage of the individual commands

PowerBoot>

SYNTAX 2: **HELP <commandName>**
 command can be used to display the help for a single command.

EXAMPLE: PowerBoot> **HELP BF**
 Command : **BF**
 Usage : **BF** - Filling Blocks

BF fills a specified RAM memory area with a byte, a word, or a long constant or with a user defined ASCII string.

SYNTAX: **BF <begin> <end> <value> [B|W|L|P]**

begin	: The first byte of the memory area used for filling
end	: The first byte of the memory area not used for filling
value	: Specifies the constant used for filling

M – Modifying the Memory

m displays and modifies the memory contents of an address.

The three tasks performed by the **m** command are:

- To display and to modify the memory contents of an address (use Syntax 1). Syntax 1 includes the size option L (default).
- To specify the memory access type and to display and modify the memory contents of an address (use Syntax 2).

SYNTAX 1: **M <address>**

SYNTAX 2: **M <address> [size & type]**

address is the first byte which is read for displaying and modifying the memory contents

size The **size** options O and E override the options B, W, and L. With the exception of access type O, the other access options B, W, L, N, and E check whether the write access has been successful by performing a read access after the write access. If the written and the read data do not match, the command is terminated and an error message displayed.

size can be one of the following:

- B = the memory access is byte-sized (8-bit)
- W = the memory access is word-sized (16-bit)
- L = the memory access is long-word-sized (32-bit)

type The **type** option specifies the memory access type. **type** can be one of the following:

- N = the memory access is limited to writing, the current contents are not displayed
- E = the memory access is byte-sized and refers only to even addresses
- O = the memory access is byte-sized and refers only to odd addresses

DESCRIPTION: **m** supports a number of commands which can be entered instead of a new memory address. These commands do not change the access option in the command line. The following commands are supported:

Command	New memory address =
cr (key)	current address + 1
=	same as current address
-	current address – 1 x size
– <i>count</i>	current address – <i>count</i> x size
+	current address + 1 x size
+ <i>count</i>	current address + <i>count</i> x size
# <i>address</i>	<i>address</i>
.	exit to the PowerBoot command line

EXAMPLE: In the following example, the memory contents of address 0000.1000₁₆ are modified:

```
PowerBoot>M 1000 b
00001000 3f : 4f
00001001 3f : =
```

```
00001001 3f : =
00001001 3f : -
00001000 3f : #2000
00002000 4f : .
PowerBoot>
```

In the following example, even addresses from 0100.0000₁₆ are modified. The N option limits access to writing at locations. No read is performed.

EXAMPLE:

```
PowerBoot>M 1000000, NE
01000000 ?? : 12
01000002 ?? : 34
01000004 ?? : ,
PowerBoot>
```

MD – Displaying Memory Contents

md displays the memory contents byte-wise and in ASCII code representation.

SYNTAX: **MD** <address> [size]

address is the first byte which is read in order to display the memory contents. The access to the address space is byte-sized.

size specifies the memory access size.

DESCRIPTION: The data is displayed in hexadecimal notation and ASCII code. If the data cannot be displayed in ASCII code, a period is displayed instead.

md displays 16 lines, each representing 16 bytes. The user now has the option to continue the display or to return to the command line by using the period “.” character.

EXAMPLE: In the following example the memory contents starting at FF70.0400₁₆ are displayed:

```
PowerBoot> MD ff700400
FF700400: 46 FC 27 00 41 FB 81 70 00 01 2F AE 4E 7B 88 01 F.'A..p../.N...
FF700410: 70 07 4E 7B 00 00 4E 7B 00 01 0E B9 00 00 00 03 p.N...N.....
FF700420: FF 00 2E 3B 81 70 00 01 D7 82 0E B9 78 00 00 03 ...i.p.....x...
FF700430: FF 00 02 87 FF FF E0 00 28 47 DE B9 FF 71 C8 6C .....(G...q.l
FF700440: 2E 47 26 7B 81 70 00 01 C4 2C 22 2C 10 40 14 2C .G&..p...,".@.,
FF700450: 10 09 02 02 00 F4 67 12 2E 3B 81 70 00 01 D7 70 .....g...i.p...p
FF700460: 02 87 00 1F FB FF 29 47 10 40 30 6C 10 16 39 7B .....)G.@01..9.
FF700470: 81 70 00 01 D7 58 10 16 22 6C 10 00 29 7B 81 70 .p...X.."l..).p
FF700480: 00 01 D7 30 10 00 24 2C 10 50 26 2C 10 54 29 7B ...0...$.P&,.T).
FF700490: 81 70 00 01 D7 22 10 50 29 7B 81 70 00 01 D7 1C .p..."P)..p....
```



```

FF7004A0: 10 54 28 2C 10 60 2A 2C 10 64 2C 2C 10 70 2E 2C .T(,.`*,.d,,.p.,
FF7004B0: 10 74 29 7B 81 70 00 01 D7 0A 10 74 29 7B 81 70 .t)..p.....t)..p
FF7004C0: 00 01 D6 FC 10 70 B7 FC FF FF FF FF 67 46 48 D3 .....p.....gFH.
FF7004D0: 00 FF D7 FC 00 00 00 20 26 EC 10 80 26 EC 10 84 .....&...&...
FF7004E0: 26 EC 10 90 26 EC 10 94 26 EC 10 A0 26 EC 10 A4 &...&...&...&...
FF7004F0: 26 EC 10 B0 26 EC 10 B4 26 EC 10 C0 26 EC 10 C4 &...&...&...&...
More (cr) ? .
PowerBoot>

```

NETLOAD – Loading a Binary Image Through Ethernet

NETLOAD loads a binary image through TFTP (Trivial File Transfer Protocol) from a host acting as server for the specified Ethernet address. For creating a connection to the server, the following two procedures are supported:

- RARP (Reverse Address Resolution Protocol)
- ARP (Address Resolution Protocol).

These protocols refer to the CPU board as target.

Note: This release supports only Ethernet port 0.

SYNTAX: **NETLOAD** [-c port] <filename> <address> [targetIP#] [serverIP#]

Port option is used to specify the on-board ethernet port to be used for **NETLOAD**/**NETSAVE**. The valid values are 0 & 1.

filename is the absolute file name (including the path name) to access the file to be loaded. The length of the file name is limited to 128 characters. The file has to be a binary image in big endian notation.

address is the first byte of the memory used for the binary image. The size of the loaded file is the only parameter which determines the number of bytes written to the memory. The binary image must be adapted to the address, because after the loading no relocation will be done. The **address** must be long-aligned.

targetIP# defines the Internet IP address of the CPU board (target), for example: gg.hh.ii.kk

serverIP# defines the Internet IP address of the server accessed for downloading the TFTP file

DESCRIPTION: The memory used for the binary image will be made available to the processor if necessary. To receive the image from the server, **NETLOAD** broadcasts a RARP or an ARP request via Ethernet. If RARP is supported, **NETLOAD** waits for the Ethernet

frame of the server that responds first. If ARP is supported, the specified server will be connected, after which the image is downloaded octet-wise via TFTP.

EXAMPLE: In the following, example the specified file is downloaded to address 0100.0000 on the CPU board:

```
PowerBoot>NETLOAD PowerBoot.bin 1000000 10.208.33.57 10.208.32.56
    Eth1 Link is Up
    MAC Address : 00:C0:8B:06:F8:64
    GMII/MII : Port works in half-duplex mode
    Port works at 10 Mbps

Transmitting ARP Request ... Reception of ARP Reply
Filename : PowerBoot.bin
Load Address : 0x01000000

Transmitting TFTP Request to Server 10.208.32.56
Options negotiated with the Server
And Connected to TFTP Server 00:07:E9:F6:4E:86
Packet : 00129 - Loaded From 0x01000000 to 0x0102CD4C (183628 bytes)
```

NETSAVE – Saving Data Through Network into File

NETSAVE saves a specified memory area via TFTP into a file located in a host system. The file cannot be created, i.e. it must already exist on the host with correct write permissions. **NETSAVE** overrides the content of the file.

In order to create a connection to the server, the following 2 procedures are supported:

- RARP (Reverse Address Resolution Protocol), and
- ARP (Address Resolution Protocol).

These protocols refer to the CPU board as target.

SYNTAX: **NETSAVE** [-c port] <filename> <startAddr> <endAddr> [targetIP#]
[serverIP#]

Port option is used to specify the on-board ethernet port to be used for **NETLOAD/NETSAVE**. The valid values are 0 & 1.

filename	specifies the name of the file in the host system
startAddr	specifies the starting address of the memory area to be saved into the file on the host
endAddr	specifies the end address of the memory area to be saved into the file on the host

targetIP#	defines the Internet IP address of the CPU board (target), for example: gg:hh:ii:kk
serverIP#	defines the Internet IP address of the server accessed for uploading the TFTP file
EXAMPLE:	In the following example the memory area 0100.0000 ₁₆ ...0119.ce38 ₁₆ is saved into the vmlinux file:

```
PowerBoot> NETSAVE vmlinux 1000000 119ce38 192.168.1.228 192.168.1.18
Eth0 Link is Up
GMII/MII : Port works in full-duplex mode
Port works at 100 Mbps

Transmitting ARP Request ... Reception of ARP Reply
Filename : vmlinux
Load Address : 0x01000000

Transmitting TFTP Request to Server 192.168.1.18
Connected to TFTP Server
- Saved From 0x01000000 to 0x0119CE38 (1691192 bytes)

/* Common Errors */
PowerBoot> NETSAVE
Parameter mismatch
NETSAVE    [] [Trgt-IP# Server-IP#]
```

SETMAC – Setting the MAC Address

SETMAC sets the MAC address for the two on-board Ethernet ports.

SYNTAX: **SETMAC** <ethInterface>

EXAMPLE: In this example, the MAC address of the on-board Ethernet Port 1 is changed:

```
PowerBoot> SETMAC 1

Current MAC for Eth1 : 04:05:60:70:73:93
New MAC for Eth1      : 40:50:60:70:73:93
MAC Address Updated Sucessfully
```

If no new value is entered when prompted for a new MAC address, the original MAC address is retained.

```
PowerBoot> SETMAC 1

Current MAC for Eth1 : 40:50:60:70:73:93
New MAC for Eth1      :
Retaining previous MAC Address
```

VERSION – Displaying Version Information

This command displays the version number and build details of PowerBoot. The screen capture is provided in the following section.

Note: The dump given below is specific to PowerBoot 3.0 on PPMC-280. The following parameters vary with the type of board:

- **Processor Name**
- **On- Board Memory**
- **MAC address**
- **Board Name**
- **Board Version**
- **TLA Number**

```
PowerBoot> VERSION
```

[illegible]

```
PowerBoot Version:    3.0
PPMC280  Version:    H/B.0
Build Date       :    Jun  7 2004
Build Time       :    15:05:52

TLA Number      :    120673
```

PowerBoot>

MEMMAP – Displaying Memory Map of the System

This command displays the memory map details. The screen capture is provided below:

```
PowerBoot> MEMMAP
System/PowerBoot Area:      0x00000000 - 0x007FFFFF
PowerBoot Code Area:       0x00200000 - 0x0022841C
Stack and Device Buffers:   0x0022841C - 0x007FFFFF

SDRAM Mapping:
Bank0 : Base 0x00000000 Size 512 MB

GT Register space mapped at 0xF1000000 - 0xF100FFFF
Internal SRAM is mapped at  0xF2000000 - 0xF203FFFF
User Flash1 mapped at      0xA0000000 - 0xA1FFFFFF
User Flash2 mapped at      0xA2000000 - 0xA3FFFFFF

PCI Memory mapped at       0x80000000 - 0x807FFFFF
PCI I/O mapped at         0x88000000 - 0x8801FFFF

PowerBoot>
```

BATCH – Storing/Executing a Set of Commands

Stores/Executes a set of upto 10 commands in Boot Flash.

SYNTAX: - **BATCH** [-c[reate] | -d[isplay]]

DESCRIPTION: The **BATCH** command can be used to create a batch of PowerBoot commands and execute the batch of commands. The currently stored batch can be displayed using the option **-d**. The batch created is stored in the NVRAM. Typing “end” terminates and saves the command list to the NVRAM.

EXAMPLE: Create a batch of commands:

```
PowerBoot>BATCH -c
Enter Batch Command 1: MEMMAP
Enter Batch Command 2: END
```

Execute the batch of commands:

```
PowerBoot> BATCH
MEMMAP

System/PowerBoot Area:      0x00000000 - 0x007FFFFF
PowerBoot Code Area:       0x00200000 - 0x0022CD48
```

```
Stack and Device Buffers:      0x0022CD48 - 0x007FFFFFFF

SDRAM Mapping:
Bank0 : Base 0x00000000 Size 512 MB

GT Register space mapped at    0xF1000000 - 0xF100FFFF
Internal SRAM is mapped at    0xF2000000 - 0xF203FFFF
User Flash1 mapped at         0xA0000000 - 0xA1FFFFFF
User Flash2 mapped at         0xA2000000 - 0xA3FFFFFF

PCI Memory mapped at          0x80000000 - 0x807FFFFF
PCI I/O mapped at             0x88000000 - 0x8801FFFF

PowerBoot>
```

BAUD – Changing Baudrate

Changes the serial port baudrate for the session.

SYNTAX: **BAUD** <baudRate>
Supported Baudrates are:
600|1200|2400|4800|9600|19200|38400|57600|115200

Note: The baud is updated only for the current session and is not updated in Boot parameters. To specify a new baudrate for the next boot, use **SETBOOT** command.

CHANGE_BOOT – Changing Bootimage File name

Changes the **NETLOAD/BOOTP** boot file name. This command also updates the boot parameters stored in Boot Flash.

SYNTAX: **CHANGE_BOOT** <bootFilename>

EXAMPLE: PowerBoot> **CHANGE_BOOT**
Syntax : **CHANGE_BOOT** <bootFilename>

PowerBoot> **CHANGE_BOOT** PowerBoot.bin
Updating Boot Filename

LINUXCMDLINE – Editing Linux Command Line

Prints or sets command line to be passed to Linux kernel, when linux is booted using **GOLINUX**.

SYNTAX: **LINUXCMDLINE -s "commandLine"**

EXAMPLE: `PowerBoot> LINUXCMDLINE -s "console=ttyS0,115200 root=/dev/ram
mtdparts=0:3072k(kernel),22528k(Ramdisk),-(JFFS2)"`

```
PowerBoot> LINUXCMDLINE  
console=ttyS0,115200 root=/dev/ram mtdparts=0:3072k(kernel),22528k(Ramdisk),-(JF  
FS2)
```

FACTORY_DEFAULTS – Restoring Default Values

This command is used to restore **SETBOOT** parameters in NVRAM to factory default values.

SYNTAX: **FACTORY_DEFAULTS**

EXAMPLE: `Powerboot> FACTORY_DEFAULTS
Powerboot>`

2

PowerBoot Commands

PowerBoot Commands

The PPMC-2xx implementation of the PowerBoot Command Line Interface (CLI) supports the generic set of commands described in the “PowerBoot Instruction Set” chapter, plus an additional set of board-specific commands. The PPMC-2xx implementation of board-specific PowerBoot commands is detailed in this chapter.

RESET – Restarting the Board

Restarts the PPMC-2xx module.

Note: RESET initiates only a jump to the /HRESET exception vector.

SYNTAX: **RESET**

DESCRIPTION: RESET restarts the PPMC-2xx module. This command forces a hard reset of the processor. The initialization of all the components on the board have to be re-done as the execution of this command is similar to a manual power OFF and power ON.

EXAMPLE: **Note:** The dump given below is specific to PowerBoot 3.0 on PPMC-280.
The following parameters vary with the type of board:

- **Processor Name**
- **On- Board Memory**
- **MAC address**
- **Board Name**
- **Board Version**
- **TLA Number**

```
POWERBOOT> RESET
INIT SERIAL MPSC0 PORT DONE

FORCE POWERPMC-280
COPYRIGHT FORCE COMPUTERS, LTD., 2003 - 2004

TOTAL MEMORY DETECTED : 0x40000000
ERROR CHECKING AND CORRECTION IS ENABLED ON ONBOARD SDRAM
MEMORY DETECTION AND CONFIGURATION COMPLETE

USING DEFAULT BOOT PARAMETERS
```

[illegible]

SETBOOT – Editing Auto Boot Parameters

Prompts the user to enter parameter values for automatic booting. The defined parameters become valid after the next power-on or after **RESET** is entered. The parameters are stored in the Boot Flash memory. **SETBOOT** calculates a checksum to protect the Boot Flash contents containing the edited auto boot parameters.

SYNTAX: **SETBOOT**

DESCRIPTION: **SETBOOT** prompts the user to assign a value to each auto boot parameter. The prompt provides the possible values of each parameter and its current setting, when appropriate.

The parameters provide the following booting information:

- Define the boot method – **GO**, **NET**, **BOOTP**, **FLASH**, **BATCH** (Boot select).
- Determine the type of booting and the location at which the binary image is to start (Auto boot, Boot address, Load address).
- Select a delay for auto booting after power-on: (Auto boot delay).
- Define the name and the path of the file loaded during auto boot (TFTP Boot file name).
- Select the Internet protocol for connecting a server to the board (RARP or ARP protocol).
- Select the protocol numbers for selecting the Trivial File Transfer Protocol (TFTP) file server and for identifying the board (Server-IP#, Target-IP#).
- Define the Baud rate to be set (Select Serial Baud).
- Select shared memory interface (Shared Memory Interface).
- Define whether or not the power on test (POT) should be executed after the next power-on or reset (Power On Test POT).
- Define the address where the POT results should be stored (POT results, Store address).

PARAMETERS

Boot select [0=GO, 1=Net, 2=BOOTP, 3=FLASH, 4=BATCH] (1):

This option defines the location of the automatically loaded or pre-programmed binary image:

- 0 = Executes a binary image located at the Boot address.
- 1 = Downloads the image from TFTP server specified and boots the image.
- 2 = Boots the device using BOOTP to load the image.
- 3 = Copy the User Flash memory contents to the Load address and start executing at the Boot address. The complete Flash bank containing the image is always copied regardless of the type and size of the devices present and the amount of already programmed or used User Flash storage capacity.
- 4 = Executes the commands stored using **BATCH** command when autoboot is enabled.

Select Serial Baud (600|1200|2400|4800|9600|19200|38400|57600|115200)[115200]:

This option specifies the baudrate to be used on next boot. Only one of the above values can be specified.

Auto boot [0=disable, 1=enable], (0):

This option enables or disables auto booting:

- 0 = Disables auto booting. All auto boot parameters are ignored.
- 1 = Enables auto booting. Auto boot will take place after the next power-on or RESET.

Auto boot delay [0..99s], (55):

This option enables a delay of auto booting for a preset period of time, ranging from 0 to 99 seconds. During the auto boot delay, the auto booting may be halted by pressing any user key on the serial console line.

Load address (01000000):

This option specifies the location in the CPU addressing space at which the opcode is downloaded. This parameter does not depend on other auto boot parameters. The binary image is always downloaded to the Load address on specifying Boot Select parameters 0 or 1.

Boot address (01000000):

This option specifies the location in the CPU addressing space at which the opcode is started. This parameter does not depend on other auto boot parameters. The binary image always starts at Boot address, regardless of whether it is downloaded during power-on to Load address, stored in the user-programmable region of Boot Flash, or stored in the User Flash.

Boot USER_FLASH [1..4], (1):

This option is used to select one of the User Flashes. There are two valid options currently available: User Flash 1 and 2; selecting 3 or 4 would result in undefined behavior.

On-Board Ethernet Port [0/1] (00000000):

This option specifies the on-board Ethernet port. (Port 0, Port 1 or Port 2).

- 0 = Specify the on-board Ethernet port as Port 0
- 1 = Specify the on-board Ethernet port as Port 1

RARP [1] or ARP [2] protocol, (1):

This option selects the Internet protocol used for connecting a server to the board.

- 1 = RARP (Reverse Address Resolution Protocol). If Internet protocol RARP is selected, Target-IP# is ignored.
- 2 = ARP (Address Resolution Protocol). If Internet protocol ARP is selected, the values entered for Server-IP# and Target-IP# are used.

For a value other than 1 or 2, the default setting, 1, is assigned to the parameter.

Server-IP# [aaa.bbb.ccc.ddd]:

This option specifies an Internet protocol number that selects a TFTP file server, for example, 123.3.255.255. If the Internet protocol RARP is selected, this parameter is ignored. The value is stored as a string.

Target-IP# [aaa.bbb.ccc.ddd]:

This option specifies an Internet protocol number that identifies the board to the Internet layer, for example, 3.255.37.67. If the Internet protocol RARP is selected, this parameter is ignored. The value is stored as a string.

TFTP Boot file name:

This option defines the name and path of a boot file to be loaded during auto booting (if Boot select is 1 or 2). The file name and path must not exceed 128 characters. The name and path must be specified completely along with the correct upper and lower case.

In order to use TFTP for booting, the host must be set up as the TFTP server. The host must provide the desired file via Ethernet (TFTP and front panel connector).

Boot Image Type [Linux = 1 | Others = 0]: (0):

This option is used along with **BOOTP** to identify the image type. This enables **BOOTP** to pass Linux command line parameters if image type is Linux.

Image type is set to 1 if the loaded image is Linux else, the image type is set to 0.

Current Linux CMDLINE (“console=ttyS0,115200 root=/dev/ram mtdparts=0:3072k(kernel),22528k(Ramdisk),-(JFFS2)”) New Cmd-line:

This option specifies the command line to be passed to Linux after next reboot of PowerBoot when **BOOTP** or **NET** option is selected in Boot Select.

TFTP Ramdisk file name : ramdisk.gz:

This option specifies a ramdisk file to be loaded when **BOOTP**, **NET** option is selected.

If this file is specified, the **BOOT** file is booted using **GOLINUX** command during auto boot and the specified ramdisk is passed to the command.

If the file name is empty, the behavior of **BOOTP**, **NET** remains unchanged and it uses the **GO** command.

Ramdisk Load address (02000000):

This option specifies the location where ramdisk image has to be downloaded.

Shared Memory Interface:

This option disables or enables the shared memory interface. Please refer to the section “Shared Memory Interface” on page 2-14 for further details.

- 0 = Disables the shared memory interface (default)
- 1 = Enables the shared memory interface

Handle EREADY pin:

This option disables or enables the EREADY pin. If this option is enabled and the Board is in Non-Monarch mode, the EREADY pin is asserted after on-board initialization.

- 0 = Disable the EREADY pin (default)
- 1 = Enable

Power On Test POT [1=disable, 0=enable], (0):

This option disables or enables the Power On Test (POT) that is executed at boot time:

- 1 = Disable the Power On Test.
- 0 = Enable the Power On Test (default)

POT Results, Store Address (0x00004000):

This option specifies the location where the POT results will be stored.

Note: The expected result for POT success is 00's. If there is any error the result will be non-zeros (i.e. not 00's) as given below:

- **POT_DISABLED** **0x0001**
- **POT_NVRAM** **0x0002**
- **POT_RAM** **0x0004**
- **POT_BOOTFLASH** **0x0008**
- **POT_ISA** **0x0010**
- **POT_PCI** **0x0020**
- **POT_CACHE** **0x0040**

The default address of POT results is: 00004000

Configurations for SETBOOT Parameters

The following are the relevant options that have to be configured for each BOOT SELECT option.

Note: Common parameters for all BOOT SELECT options are:

- Baud rate setting
- Power-On Test
- POT results
- Shared memory interface
- Handle EREADY pin

The other options can be ignored.

Configurations for GO Boot Select Option:

- Auto boot
- Auto boot delay
- Boot address

Configurations for NET Boot Select Option:

- Auto boot
- Auto boot delay
- Load address
- RARP or ARP protocol
- Ethernet interface number
- Server IP
- Target IP
- Current Linux CMDLINE
- TFTP/Disk boot file name
- TFTP Ramdisk file name (when auto boot option is enabled)
- Ramdisk load address (when auto boot option is enabled)

Configurations for BOOTP Boot Select Option:

- Auto boot
 - Auto boot delay
-

- Load address
- RARP or ARP protocol
- Ethernet interface number
- Server IP
- Target IP
- Current Linux CMDLINE
- TFTP/Disk boot file name
- Boot image type
- TFTP Ramdisk file name (when auto boot option is enabled and Boot image type is Linux)
- Ramdisk load address (when auto boot option is enabled and Boot image type is Linux)

Configurations for FLASH Boot Select Option:

- Auto boot
- Auto boot delay
- Load address
- Boot User Flash

Configurations for BATCH Boot Select Option:

- Auto boot
- Auto boot delay

EXAMPLE: For GO Boot Select Option:

```
PowerBoot> SETBOOT
This will modify Boot Flash Contents.Do you want to continue(Y/[N])? y

-General Boot Parameters-

Boot select [0=GO, 1=Net, 2=BOOTP, 3=FLASH, 4=BATCH] (0) : 0
Select Serial Baud
(600|1200|2400|4800|9600|19200|38400|57600|115200)[115200]:
Auto boot [0=disable, 1=enable], (0) : 1
Auto boot delay [0..99s], (5) : 10
Load address (01000000) :
```

```

Boot address (01000000) : fff00100

-TFTP Ethernet, Boot file parameters-

RARP [1] or ARP [2] protocol : (2) :
Server-IP# [aaa.bbb.ccc.ddd] : 0.0.0.0 :
Target-IP# [aaa.bbb.ccc.ddd] : 0.0.0.0 :
TFTP/Disk Boot file name :
vmlinux.bin :
Current Linux CMDLINE ()
New Cmd-line :
Shared Memory Interface [0=disable, 1=enable], (0) : 1
Handle EREADY pin (For Monarch/Non-Monarch) [0=disable, 1=enable], (0) :

-Power On Test (POT) parameters-

Power ON Test POT [1=disable, 0=enable], (0) :
POT Results, Store Address (00004000) :
CSUM : 0x747
PowerBoot>

```

Note:

- **All other parameters are for internal use only. Do not change their default settings.**
 - **Shared memory interface is disabled by default; it may be enabled using SETBOOT command.**
-

Persistent Memory

Provides a mechanism to retain the contents of memory over a specified range, even after reset, using either software or hardware.

Enabling Persistent memory feature

MAGIC VALUE	Magic value is used enable this feature. Magic value used is 0x26A1BD0F. m command is used to set this value at location 0x1F0 in main memory.
START ADDRESS	Here 8 MB aligned start address should be specified at location 0x1E0 in main memory using m command.
END ADDRESS	Here 8 MB aligned end address should be specified at location 0x1E8 in main memory using m command.

EXAMPLE:

```
PowerBoot>
PowerBoot> M 1f0
000001f0 00000000 : 26a1bd0f
000001f4 00000000 : .
PowerBoot> M 1e0
000001e0 00000000 : 1000000
000001e4 00000000 :
000001e8 00000000 : 2000000
000001ec 00000000 : .
PowerBoot>
```

After Hard Reset

Note: The dump given below is specific to PowerBoot 3.0 on PPMC-280.
The following parameters vary with the type of board:

- **Processor Name**
 - **On- Board Memory**
 - **MAC address**
 - **Board Name**
 - **Board Version**
 - **TLA Number**
-

Init serial MPSC0 port done

Force PowerPMC-280
Copyright Force Computers, Ltd., 2003 - 2004

Total Memory detected : 0x40000000
Error Checking and Correction is enabled on onboard SDRAM
Memory detection and configuration complete

Using default Boot parameters
Persistent Memory is Disabled

Executing in Monarch mode
Init DTLB/ITLB for block translation, enable MMU
Instruction Cache is enabled
Init exception vectors starting at address: 0x00000100

Found CPU MPC7447 at 1000MHz PVR=80020101.

Onboard Memory DRAM: 1024MB 0x00000000 - 0x3FFFFFFF
GT Register space mapped at 0xF1000000 - 0xF100FFFF

Initializing on-board ethernet
Initializing Eth0 ... done

[illegible]

Shared Memory Interface

The Shared Memory Interface feature provides a mechanism for redirecting the PowerBoot console to other hosts via the PCI interface. In order to use this feature, the other host through which the PowerBoot console is to be viewed must support the SHM driver.

Please refer the host documentation for details on the SHM driver and the configuration procedure.

The following Figure 1 “Shared Memory Interface Setup” shows the shared memory interface setup.

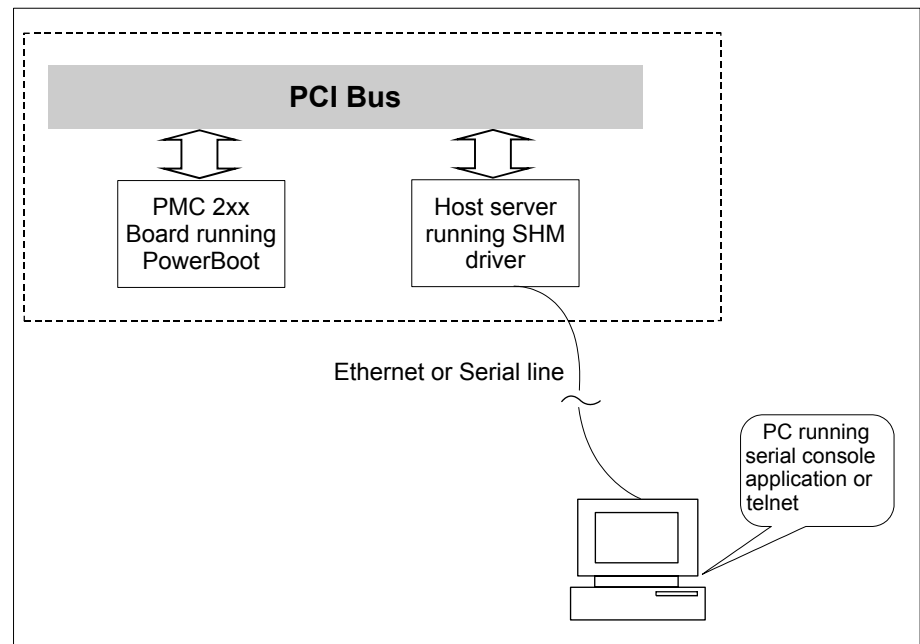


Figure 1: *Shared Memory Interface Setup*

Command for Accessing the PCI Bus

The PPMC-2xx implementation of PowerBoot provides a command for accessing the PCI bus: **BUSSHOW**.

Caution

This command is valid only if the PPMC-2xx is running in Monarch mode and the baseboard is a CompactPCI slot 1 board. If the module is running in Non-Monarch/PCI-agent mode, these commands may lock the system.

BUSSHOW – Displaying PCI Devices

Scans all PCI bus segments and for each PCI agent found, prints the vendor/device ID, bus number, device number, and the device function number.

SYNTAX: **BUSSHOW**

EXAMPLE:

```
PowerBoot> BUSSHOW
Scanning PCI 0 interface
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 0
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 1
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 2
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 3
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 4
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 5
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 6
Found PCI device: Device/VendorID: 0x646011AB at bus 0, device 0, function 7
Found PCI device: Device/VendorID: 0xB1548086 at bus 0, device 8, function 0
Found a Bridge
PowerBoot>
```

PROBEPCI – Displaying detailed PCI Device Information

The PROBEPCI command displays the PCI devices similar to **BUSSHOW** command, providing additional details like Interrupt configuration, Latency settings, base addresses, device classes and the Vendor name.

SYNTAX: **PROBEPCI**

EXAMPLE:

```
PowerBoot> PROBEPCI
Probing PCIbus at 0x80000000
Device ID = 0x6460; Vendor ID = 0x11AB;
Status    = 0x02B0; Command  = 0x0007;
```



```

Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x0000000C, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000100
Device ID = 0x6460; Vendor ID = 0x11AB;
Status = 0x02B0; Command = 0x0007;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x4000000C, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000200
Device ID = 0x6460; Vendor ID = 0x11AB;
Status = 0x02B0; Command = 0x0007;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0xA0000004, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000300
Device ID = 0x6460; Vendor ID = 0x11AB;
Status = 0x02B0; Command = 0x0007;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0xFF000004, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000400
Device ID = 0x6460; Vendor ID = 0x11AB;
Status = 0x02B0; Command = 0x0007;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x9000000C, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000500
Device ID = 0x6460; Vendor ID = 0x11AB;
Status = 0x0000; Command = 0x0000;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID = 0x03;
BIST = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x00000000, base addr1= 0x00000000;
Max Lat = 0x00; Min Gnt = 0x00; IRQ Pin = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIbus at 0x80000600

```

```

Device ID = 0x6460; Vendor ID = 0x11AB;
Status    = 0x0000; Command  = 0x0000;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID  = 0x03;
BIST      = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x00000000, base addr1= 0x00000000;
Max Lat   = 0x00; Min Gnt   = 0x00; IRQ Pin   = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIBus at 0x80000700
Device ID = 0x6460; Vendor ID = 0x11AB;
Status    = 0x0000; Command  = 0x0000;
Base Class= 0x05; Sub Class = 0x80; Prg. Inter= 0x00; Rev. ID  = 0x03;
BIST      = 0x80; Header Typ= 0x80; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x00000000, base addr1= 0x00000000;
Max Lat   = 0x00; Min Gnt   = 0x00; IRQ Pin   = 0x01; IRQ Line = 0x00;
Found PCI device: Discovery Chipset, PCI interface

```

```

Probing PCIBus at 0x80004000
Device ID = 0xB154; Vendor ID = 0x8086;
Status    = 0x02B0; Command  = 0x0000;
Base Class= 0x06; Sub Class = 0x04; Prg. Inter= 0x00; Rev. ID  = 0x00;
BIST      = 0x00; Header Typ= 0x01; Latency Ti= 0x00; Cache Line= 0x00;
base addr0= 0x00000000, base addr1= 0x00000000;
Max Lat   = 0x00; Min Gnt   = 0x00; IRQ Pin   = 0x00; IRQ Line = 0x00;
Found PCI device: Intel

```

```

Probing PCIBus at 0x8000F800
PowerBoot>

```

CONFIG_RD – Performing a PCI Bus Configuration Read Cycle

Performs a PCI bus configuration read cycle on a specified address. Specify the PCI bus device, the address/register offset of the location to be read, and the size of the data (8, 16, or 32 bits).

SYNTAX: **CONFIG_RD** <busNr> <devNr> <funNr> <regNr> [B|W|L]

busNr	specifies a PCI bus segment number
devNr	specifies a PCI bus device number
funNr	specifies a PCI bus device function number
regNr	specifies the address/register offset of the location to be read

[B/W/L] specifies the size of the data to be read

where,

B = byte (8 bits)

W = word (16 bits)

L = longword (32 bits)

EXAMPLE: The following command performs a PCI bus configuration read cycle for 32 bits of data:

```
PowerBoot> CONFIG_RD 0 18 0 4 1
address: 0x8000C004
0xFFFFFFFF
PowerBoot>
```

CONFIG_WR – Performing a PCI Bus Configuration Write Cycle

Performs a PCI bus configuration write cycle to a specified address. Specify the PCI bus device, the address/register offset of the location to be written, the data to be written, and the size of the data (8, 16, or 32 bits).

SYNTAX: `CONFIG_WR <busNr> <devNr> <funNr> <regNr> <data> [B|W|L]`

busNr specifies a PCI bus segment number

devNr specifies a PCI bus device number

funNr specifies a PCI bus device function number

regNr specifies the address/register offset of the location to be read

[B/W/L] specifies the size of the data to be read

where,

B = byte (8 bits)

W = word (16 bits)

L = longword (32 bits)

EXAMPLE: The following command performs a PCI bus configuration write cycle for 8 bits of data:

```
PowerBoot> CONFIG_WR 0 18 0 19 1 b
address: 0x8000C019
```

Commands for Programming and Verifying Flash Memory

PowerBoot provides the **FERASE**, **FPROG** and **FVERIFY** commands for programming and verifying Flash memory devices. These commands are described in detail in the the “PowerBoot Instruction Set” chapter. This section describes the use of these commands in connection with the PPMC-2xx on-board Flash memory.

Caution



Ensure not to erase Flash memory and re-program it, unless certain. Faulty re-programming will result in a damaged and inoperative board.

Reprogramming Boot Flash

The PPMC-2xx Boot Flash device (**BOOT_FLASH**) provides a total of 1 MB of Flash memory with PowerBoot firmware pre-installed. One MB of Boot Flash is mirrored eight times in the 8 MB Boot Flash region.

For example, use PowerBoot commands to place an operating system loader image, such as a VxWorks boot ROM image, into the user-programmable 8 MB region of Boot Flash.

Note: When reprogramming the PPMC-2xx Boot Flash device using **FPROG**, an **FERASE** is performed automatically to erase the target region. Other platforms may require a manual issue of **FERASE** command before the **FPROG**.

Table 2: *Boot Flash Address Range*

Component	Starting Address	Ending Address	Size
BOOT_FLASH	FF80 0000	FFFF FFFF	8 MB
NVRAM AREA	FFF4 0000	FFF4 FFFF	64 KB

Program the Boot Flash device as follows:

1. Start PowerBoot by powering on the PPMC-2xx module.

PowerBoot> _

2. In order to reprogram the Boot Flash device, load the new program image into DRAM memory. For example, use the PowerBoot **NETLOAD** command.
(The **NETLOAD** command loads a binary image via TFTP from a host acting as a server; see the the “PowerBoot Instruction Set” chapter for more information.)

```
PowerBoot> NETLOAD power.bin 1000000 10.208.1.208 10.208.1.158
PHY-Device at 100MB/s negotiated
WANCOM MAC ADDRESS : FE:FF:FF:00:43:00
Transmitting ARP-REQUEST... Reception of ARP-REPLY
Transmitting TFTP-REQUEST to server 00:D0:09:F9:FF:5B, IP
10.208.1.158
PACKET:1 PACKET:50 PACKET:100 PACKET:150 PACKET:200 PACKET:250
PACKET:300 PACKET:350 PACKET:400 PACKET:414 - loaded
$01000000..$01033A5A (211547 bytes)
PowerBoot> _
```

3. Verify the DRAM memory contents by using the PowerBoot **MD** command:

```
PowerBoot> MD 1000000
01000000: 46 6F 72 63 65 20 43 6F 6D 70 75 74 65 72 73 2F Force Computers/
01000010: 42 61 6E 67 61 6C 6F 72 65 2F 73 77 65 6E 67 2F Bangalore/sweng/
01000020: 50 6F 77 65 72 42 6F 6F 74 20 33 2E 30 20 20 20 PowerBoot 3.0
01000030: 46 6F 72 20 50 50 4D 43 32 38 30 20 20 20 20 20 For PPMC280
01000040: 44 61 74 65 3A 4A 75 6E 20 32 38 20 32 30 30 34 Date:Jun 28 2004
01000050: 54 69 6D 65 3A 31 39 3A 33 38 3A 30 37 20 20 20 Time:19:38:07
01000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
More (cr) ?
PowerBoot>
```

4. Program the image residing in DRAM memory into the Boot Flash device by using the PowerBoot **FPROG** command. The Boot Flash device, the source DRAM address, and the destination Boot Flash offset need to be specified.

Note: The Boot Flash offset specified in the **FPROG** command is relative to the base address of the Boot Flash, **FFF00000**.

For example, the following command programs the DRAM memory contents beginning at 1000000_{16} into the Boot Flash beginning at offset 0000.0000_{16} , which corresponds to address **FFF00000**:

```
PowerBoot> FPROG boot_flash 1000000 0 33a5b
Warning, error during programming could render board un-operational!
Confirm Boot Flash programming [0=no, 1=yes] (0) : 1
Programming PowerBoot Flash memory
Will need to power cycle after completion
PowerBoot>
```

5. Use the PowerBoot **MD** command, to view the contents of the programmed Boot Flash device:

```
PowerBoot> MD fff00000
FFF00000: 46 6F 72 63 65 20 43 6F 6D 70 75 74 65 72 73 2F Force Computers/
FFF00010: 42 61 6E 67 61 6C 6F 72 65 2F 73 77 65 6E 67 2F Bangalore/sweng/
FFF00020: 50 6F 77 65 72 42 6F 6F 74 20 33 2E 30 20 20 20 PowerBoot 3.0
FFF00030: 46 6F 72 20 50 50 4D 43 32 38 30 20 20 20 20 20 For PPMC280
FFF00040: 44 61 74 65 3A 4A 75 6E 20 31 30 20 32 30 30 34 Date:Jun 10 2004
FFF00050: 54 69 6D 65 3A 31 38 3A 31 33 3A 33 38 20 20 20 Time:18:13:38
FFF00060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF00070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF00080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF00090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
FFF000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
More (cr) ?
PowerBoot>
```

6. In order to execute the programmed image, reset the board.

Programming User Flash

The User Flash device may be programmed as follows:

1. Start PowerBoot by powering on the PPMC-2xx module.

```
PowerBoot> _
```

2. Erase the User Flash using the **FERASE** command. For example, use the following command to erase User Flash:

```
PowerBoot> FERASE user_flash1
```

```
Are you sure you wish to erase the Flash:[yes = 1,no = 0] : 1
```

Note:

- The User Flash device must be erased prior to reprogramming.
 - The **FERASE** command shown erases the entire 32 MB or 64 MB memory of the User Flash device. Erasing the entire User Flash is a lengthy operation. In order to erase smaller parts of the User Flash device memory, the offset and length parameters need to be specified. For a list of these parameters, see the the “PowerBoot Instruction Set” chapter.
-

3. In order to reprogram the User Flash device just erased, load the new program image into DRAM memory. For example, use the PowerBoot **NETLOAD** command. (The **NETLOAD** command loads a binary image via TFTP from a host acting as a server; see the the “PowerBoot Instruction Set” chapter for more information.)

```
PowerBoot> NETLOAD linux_exp 1000000 10.208.33.57 10.208.32.56
Eth1 Link is Up
MAC Address : 00:C0:8B:06:F8:64
GMII/MII : Port works in half-duplex mode
Port works at 10 Mbps

Transmitting ARP Request ... Reception of ARP Reply
Filename : PowerBoot.bin
Load Address : 0x01000000

Transmitting TFTP Request to Server 10.208.32.56
Options negotiated with the Server
And Connected to TFTP Server 00:07:E9:F6:4E:86
Packet : 00129 - Loaded From 0x01000000 to 0x0102CD4C (183628 bytes)
```

4. Verify the DRAM memory contents by using the PowerBoot **MD** command:

```
PowerBoot> MD 1000000
01000000: 60 00 00 00 60 00 00 00 60 00 00 7C 7F 1B 78 `...`...`.....x
01000010: 7C 9E 23 78 7C BD 2B 78 7C DC 33 78 7C FB 3B 78 ..#x...+x...3x..ix
01000020: 3B 00 00 00 48 19 5A D1 48 00 38 19 48 00 37 A5 ;...H.Z.H.8.H.7.
01000030: 48 00 37 F9 48 00 38 31 48 00 5C BD 7C 7A 1B 78 H.7.H.81H.\..z.x
01000040: 3C 83 C0 00 2C 04 00 00 40 82 33 78 7C 00 00 A6 <...,...@.3x....
01000050: 60 00 00 30 7C 1B 03 A6 3C 00 C0 00 60 00 36 E8 `..0....<...`.6.
01000060: 7C 1A 03 A6 4C 00 00 64 00 00 00 00 00 00 00 00 ....L..d.....
01000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000C0: 38 60 00 01 90 60 00 04 48 00 00 00 00 00 00 00 8`...`.H.....
010000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

```
010000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
More (cr) ?
```

PowerBoot>

5. Program the image residing in DRAM memory into the User Flash device by using the PowerBoot **FPROG** command. The user_flash1 device, the source DRAM address, and the destination User Flash offset need to be specified.

Note: The User Flash offset specified in the **FPROG** command is relative to the base address of the User Flash, A0000000 for User_Flash1 and A2000000 for User_Flash2.

For example, the following command programs the DRAM memory contents beginning at 0100.0000₁₆, for a length of 32 MB or 64 MB (based on the PPMC-2xx User Flash size), into the User Flash beginning at offset 0000.0000₁₆, which corresponds to address A0000000:

```
PowerBoot> FPROG user_flash1 1000000 0
```

Programming flash memory

0 ... 100%

Done.

PowerBoot>

6. In order to view the contents of the programmed User Flash device, use the PowerBoot **MD** command:

```
PowerBoot> MD a0000000
```

```
A0000000: 60 00 00 00 60 00 00 00 60 00 00 00 7C 7F 1B 78 `...`...`.....x
A0000010: 7C 9E 23 78 7C BD 2B 78 7C DC 33 78 7C FB 3B 78 ..#x..+x..3x../x
A0000020: 3B 00 00 00 48 19 5A D1 48 00 38 19 48 00 37 A5 ;...H.Z.H.8.H.7.
A0000030: 48 00 37 F9 48 00 38 31 48 00 5C BD 7C 7A 1B 78 H.7.H.81H.\..z.x
A0000040: 3C 83 C0 00 2C 04 00 00 40 82 33 78 7C 00 00 A6 <...@.3x....
A0000050: 60 00 00 30 7C 1B 03 A6 3C 00 C0 00 60 00 36 E8 `..0....<...`.6.
A0000060: 7C 1A 03 A6 4C 00 00 64 00 00 00 00 00 00 00 00 ....L..d.....
A0000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A0000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A0000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A00000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A00000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A00000C0: 38 60 00 01 90 60 00 04 48 00 00 00 00 00 00 00 8`...`.H.....
A00000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A00000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A00000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
More (cr) ?
PowerBoot>
```


3

Getting Started with PowerBoot

Getting Started

The PowerBoot Command Line Interface (CLI) provides an interface to the firmware of a system. Commands to load, test, and debug software in system memory may be issued from the serial console.

The PPMC-2xx implementation of the PowerBoot CLI supports:

- Generic PowerBoot commands
(BF, BM, BS, BT, BV, FERAISE, FVERIFY, FPROG, FRCEEPROMREAD, FRCEEPROMWRITE, GO, GOLINUX, HELP, FREAD, M, MD, NETLOAD, NETSAVE, VERSION, MEMMAP, and FACTORY_DEFAULTS)
- Board-specific commands described in Chapter 2 of this manual
(BUSSHOW, PROBEPCI, RESET, SETBOOT, SETMAC, GETMAC, CONFIG_RD, and CONFIG_WR)

This chapter details how to get started with PowerBoot.

Setting Up the System for PowerBoot

In order to use the PowerBoot CLI, the system is required to be connected to a console device. The console device can be a video terminal connected with a serial line port or a PC or workstation connected to the system through the serial port.

For information on installing serial-line or network cables, see the appropriate Hardware Installation Guide.

The system connected to the console device should use the following parameters:

- Send/receive 115200 baud
- Eight (8) bit data word
- No parity
- One (1) stop bit
- No flow control

PowerBoot Features

PowerBoot firmware is stored in the module's on-board Boot Flash memory and features the following:

- Power-on tests (POTs) that can be disabled
- Simple CLI and command set
- Utilities for configuring boot parameters and system components, such as PCI devices
- Set of basic binary tests for testing main memory
- Networking capabilities for loading binary images into memory
- Support for scanning of PCI bus

PowerBoot Ethernet Support

PowerBoot supports two on-board MGI Ethernet ports for download of:

- Linux Kernel/Ramdisk
- VxWorks image
- Firmware for upgrade purposes

Entering PowerBoot

The PowerBoot CLI comes up automatically when the power-on tests (POTs) are completed during system startup. The system displays the following prompt on entering the CLI:

```
PowerBoot>
```

The system also enters PowerBoot when a soft reset is generated with the PowerBoot RESET command.

Exiting PowerBoot

The PowerBoot CLI exits when:

- The `GO/GOLINUX` command is issued to start a binary image that has been loaded into main memory
- Auto booting is enabled and the system automatically jumps to the user-specified boot address

Getting Online Help

To display online help, enter the **HELP** command. This command displays each PowerBoot command with a brief description, as shown below:

```
PowerBoot> HELP
```

```
--- Command words ---
```

```
BATCH - Stores/Executes a set of commands
BAUD - Changes Console Baudrate
BF - Fills Block of Memory Region
BM - Copy Block of Memory Region
BS - Searches a Pattern in a Block of Memory
BT - Tests the Specified Memory Region
BUSSHOW - Displays PCI Devices (PCI Scan)
BV - Verify Block of Memory Region
CHANGE_BOOT - Changes Bootimage File name in NVRAM
CONFIG_RD - Performs a PCI Bus Configuration Read Cycle
CONFIG_WR - Performs a PCI Bus Configuration Write Cycle
FACTORY_DEFAULTS - Sets Setboot parameters to factory defaults
```

```
Page 1 : Enter CR for more Commands
```

```
FERASE - Erases Boot/User Flash Memory
FPROG - Programs Boot/User Flash Memory
FRCEEPROMREAD - Reads data from EEPROM Device
FRCEEPROMWRITE - Writes data to EEPROM Device
FREAD - Reads User Flash Memory
FVERIFY - Verify Boot/User Flash Memory
GETMAC - Get the MAC Address
GO - Executes a Binary Images located in the Memory
GOLINUX - Loading Linux using RAMDISK
HELP - Displays Help Messages
LINUXCMDLINE - Edits Linux command line
M - Modifies the Memory Contents
```

```
Page 2 : Enter CR for more Commands
```

```
MD - Displays Memory Contents
MEMMAP - Displays Memory Map of the System
NETLOAD - Loads a Binary Image into memory through Network
NETSAVE - Saving Data through Network into File
PROBEPCI - Displays PCI Devices
RESET - Does a warm reboot
SETBOOT - Edits Auto Boot Parameters
SETMAC - Set the MAC Address
VERSION - Displays version Information
```


"**HELP** <command>": Displays Syntax and Usage of the individual commands

PowerBoot>

Command Line Requirements and Restrictions

The following Table 3 lists PowerBoot command line requirements and restrictions.

Table 3: *PowerBoot Command Line Requirements and Restrictions*

Requirement or Restriction	Explanation
Length	A command line consists of a fixed number of characters, not including the terminating carriage return or any characters that are deleted on entering the command line. The maximum command line length differs among the various commands.
Case	A command line can consist of upper or lower case characters. Characters are displayed in the entered case
Options	The behavior of some commands may be altered by specifying them along with their options. See individual command descriptions for examples.
Numbers	Numbers specified as command parameters are interpreted as hexadecimal values.
ASCII strings	ASCII strings are to be defined within double quotes (" ") followed by the option P.
File names	File names must include absolute pathnames and cannot exceed 128 characters.
Ethernet addresses	Specify Ethernet addresses with the notation <i>xx:xx:xx:xx:xx:xx</i> .
Internet addresses	Specify Internet Protocol (IP) addresses with the standard Internet 4-decimal position "." notation — <i>nnn.nnn.nnn.nnn</i> .
No characters	A command line with no characters is a null command. PowerBoot takes no action and does not issue an error message; it returns the PowerBoot prompt.
Spaces or tabs	PowerBoot compresses and treats multiple adjacent spaces and tabs as a single space and ignores leading and trailing spaces.

Special Keys and Characters

Table 4 lists special keyboard keys and characters that perform specific PowerBoot command operations.

Table 4: *Special Keys and Characters for PowerBoot Operations*

Key or Character	Operation
" "	Denote an ASCII character string.
=	When used with the PowerBoot M command to access a memory location, provides access to the current address.
–	When used with the PowerBoot M command to access a memory location, provides access to the current address minus one times the specified size (<i>current-address – 1 x size</i>).
–count	When used with the PowerBoot M command to access a memory location, provides access to the current address minus <i>count</i> times the specified size (<i>current-address – count x size</i>).
+	When used with the PowerBoot M command to access a memory location, provides access to the current address plus one times the specified size (<i>current-address + 1 x size</i>).
+count	When used with the PowerBoot M command to access a memory location, provides access to the current address plus <i>count</i> times the specified size (<i>current-address + count x size</i>).
#address	When used with the PowerBoot M command to access a memory location, provides access to the specified address.
.	Exits the current command and returns to the PowerBoot prompt.
Backspace	Deletes one character.
Enter	Terminates a command line. No action is taken on a command until terminated. If no characters are keyed in before pressing <Enter>, the PowerBoot re-displays the prompt. When used with the PowerBoot M command to access a memory location, provides access to the current address plus 1 (<i>current-address + 1</i>). When used with the PowerBoot MD command, displays the next screen of memory contents.
Return	Terminates a command line. No action is taken on a command until terminated. If no characters are keyed in before pressing <Return>, the PowerBoot re-displays the prompt.

Command Line Characteristics

PowerBoot command line display the following characteristics:

- The command interpreter is not case-sensitive. Lowercase ASCII characters *a* to *z* are treated as uppercase characters.
- The console does not provide type-ahead buffer support.

Index

A

Arguments
for PowerBoot console commands 1-5

B

Backspace 3-11
Baud rates 3-4

C

Command line, PowerBoot console
characteristics of 3-12
terminating 3-11
Commands
PowerBoot console 1-3
case sensitivity of 3-12
casing of 3-10
length of 3-10
numbers specified with 3-10
options for 3-10
specifying arguments with 1-5
specifying options with 1-5
specifying spaces with 3-10
specifying tabs with 3-10
terminating 3-11
type-ahead buffer support for 3-12
Console
prompt 3-6
Console device 3-4

E

Enter key 3-11

N

Null command 3-10

O

Options, PowerBoot command 1-5

P

Parameters, console device 3-4
Parity 3-4
Ports
serial ports
setting parameters for 3-4
POT (power-on test) 3-6
PowerBoot console
command line 3-12
PowerBoot console commands 1-3
case sensitivity of 3-12
casing of 3-10
length of 3-10
numbers specified with 3-10
options for 1-5, 3-10
specifying arguments with 1-5
specifying spaces with 3-10
specifying tabs with 3-10
terminating 3-11
type-ahead buffer support for 3-12
Power-on test (POT) 3-6

R

Return key 3-11

S

Serial ports
setting parameters for 3-4
Spaces 3-10
Stop bit 3-4

T

Tabs 3-10

Product Error Report

Product:	Serial No.:
Date Of Purchase:	Originator:
Company:	Point Of Contact:
Tel.:	Ext.:
Address: _____ _____ _____	
Present Date:	
Affected Product: o Hardware o Software o Systems	Affected Documentation: o Hardware o Software o Systems
Error Description: _____ _____ _____ _____ _____ _____ _____	
<p>This Area to Be Completed by Force Computers:</p> <p>Date:</p> <p>PR#:</p> <p>Responsible Dept.: o Marketing o Production Engineering ‡ o Board o Systems</p>	

+ Send this report to the nearest Force Computers headquarters, as listed on the address page.

