SPARC/CPSB-560 OpenBoot Enhancements

Programmer's Guide

P / N 227476 Revision AA
September 2005

Copyright

Notice

While reasonable efforts have been made to assure the accuracy of this document, Motorola GmbH assumes no liability resulting from any ommissions in this document, or from the use of the information obtained herein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the Motorola Embedded Communications Computing Web site. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of Motorola GmbH.

It is possible that this publication may contain reference to or information about Motorola products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

Contents

2 Board Configuration

4 Access to the IPMI Controller

Index

Tables

Introduction

Board Configuration

Flash

Access to the IPMI Controller

Figures

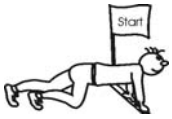
Access to the IPMI Controller

Using This Guide

This Programmer's Guide is intended for software developers writing applications to run on the SPARC / CPSB-560. This manual describes how to use the board-specific OpenBoot commands provided by Motorola. Throughout this manual it is assumed that you are generally familiar with stack-oriented Forth commands, OpenBoot firmware, CompactPCI, and IPMI.

Conventions

Notation	Description
57	All numbers are decimal numbers except when used with the notations described below
00000000_{16}	Typical notation for hexadecimal numbers (digits are 0 through F), e.g. used for addresses and offsets
0000_2	Same for binary numbers (digits are 0 and 1)
x	Generic use of a letter
n	Generic use of numbers
Bold	Character format used to emphasize a word
`Courier`	Character format used for on-screen output
`Courier+Bold`	Character format used to characterize user input
Italics	Character format for references, table, and figure descriptions
[text]	Typical notation for optional parameters
<text>	Typical notation used for variables and keys
{text text}	Typical notation used for parameters which are returned together after successful command execution.
`command (parameter — return)`	Typical notation for the syntax of OpenBoot commands with postfix notation, i.e. commands where parameters have to be entered before the command itself. Return values written on the right side of the "—" are put on the stack.
...	Repeated item

Notation	Description
..	Ranges
|	Logical OR
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
	Start of a procedure
	End of a procedure

Abbreviations

APB	Advanced PCI Bridge
ASM	Advanced System Monitoring
BIB	Board Information Block
BIOS	Basic Input Output System
BMC	Base Board Management Controller
CORE	Common Operations and Reset Environment
DIMM	Dual Inline Memory Module
EPROM	Erasable Programmable Read-Only Memory
FCODE	Forth Code
FPGA	Field-Programmable Gate Array
FRU	Field Replaceable Unit
I^2C	Intelligent I/O Controller
ICMB	Intelligent Chassis Management Bus

IDE	Integrated Drive Electronics
ID PROM	Identification Programmable Read-Only Memory
IPMB	Intelligent Peripheral Management Bus
IPMI	Intelligent Platform Management Interface
IRQ	Interrupt Request
KCS	Keyboard Controller Style
LSB	Least Significant Byte
LUN	Logical Unit Number
MSB	Most Significant Byte
NDA	Non-Disclosure Agreement
NVRAM	Nonvolatile Random Access Memory
OBP	OpenBoot PROM
OEM	Original Equipment Manufacturer
OS	Operating System
PCI	Peripheral Component Interconnect
PM	Peripheral Management Controller
POH	Power-On Hours
POST	Power-On Self-Test
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
ROM	Read-Only Memory
RTB	Rear Transition Board
SBC	Single Board Computer
SCSI	Small Computer System Interface
SDR	Sensor Data Record
SDRAM	Synchronous Dynamic Random Access Memory
SEL	System Event Log
SMART	Software Maintenance and Reference Tools
SMC	System Management Controller

SPD	Serial Presence Detect
SRAM	Static RAM
SROM	Serial Read-Only Memory

Revision History

Order No.	Revision	Date	Description
217648	AA	February 2003	First version of Programmer's Guide
220733	AA	May 2003	Updated IPMI firmware upgrade procedure Added SDR upgrade procedure
223825	AA	July 2004	Replaced IPMI firmware upgrade procedure and SDR upgrade procedure with reference to *IPMI Firmware for SPARC/CPCI-550 and SPARC/CPSB-560 Installation Guide* Updated section Other Sources of Information Editorial changes
227476	AA	September 2005	Changed manual to Motorola-style (logo, copyright, etc.)

Other Sources of Information

For further information refer to the following documents:

Company	URL	Document
Motorola	www.motorola.com/computers	SPARC/CPSB-560 Reference Guide
		Intelligent Board Management Unit Reference Guide
		IPMI Firmware for SPARC/CPCI-550 and SPARC/CPSB-560 Installation Guide
Intel	www.intel.com/design/servers/ipmi/spec_old.htm	IPMI Specification v1.0 Rev. 1.1
Sun	www.sun.com/oem/products/manuals/index.html	SPARCengine ASM Reference Manual
	www.sun.com	OpenBoot 3.x Quick Reference
Vitesse	www.vitesse.com	VSC215 IPMI Baseboard Management Controller, Software Developer's Kit document

1

Introduction

About this Manual

This reference Guide describes in detail the OpenBoot enhancements available for the SPARC / CPSB-560 board. It provides a detailled reference of all supported OpenBoot commands.

Organization of this Manual

The Reference Guide is organized as follows.

Chapter	Description
Using This Guide	Lists all conventions and abbreviations used in this manual and outlines the revision history
Other Sources of Information	Lists related documentation and specifications
Introduction	Provides a basic overview of the features of the OpenBoot
Board Configuration	Describes OpenBoot commands that allow to modify the board configuration
Flash	Describes OpenBoot commands that are related to the on-board flash
Access to the IPMI Controller	Describes OpenBoot commands that are related to the on-board IPMI controller

Feedback

Motorola welcomes and appreciates your comments on its documentation. We want to know what you think about our manuals and how we can make them better. Mail comments to:

- Motorola GmbH
  ECC Embedded Communication Computing
  Lilienthalstr. 15
  85579 Neubiberg-Munich / Germany
- reader-comments@mcg.mot.com

Overview

The OpenBoot on the SPARC / CPSB-560 is based upon OpenBoot V4.0 obtained from Sun Microsystems. Apart from the commands already provided by the standard OpenBoot firmware, the SPARC / CPSB-560 OpenBoot firmware additionally provides:

- Commands to:
  - Display and change settings of board components and functions (LEDs, watchdog, timer, etc.)
  - Display settings of hardware switches
  - Display CPU and board temperature
  - Access and program I^2C-based memory
  - Access and program the flash
  - Access the IPMI controller
- Variables to
  - Configure probe lists
  - Configure the IPMI watchdog timer
  - Select the keyboard / mouse type
  - Display messages while erasing or programming the flash

Note: OpenBoot is subject to changes. Some features are only available with specific versions of the software. For information on the latest OpenBoot version and how to upgrade it, refer to the former Force Computers S.M.A.R.T. service or the Motorola website.

For a description of standard OpenBoot 4.x firmware features, see the OpenBoot 4.x Manual Set.

Entering OpenBoot Commands

The OpenBoot commands can either be entered at the ok prompt or into the nvramrc via the NVRAM editor. If used in nvramrc, the commands are executed during power up.

OpenBoot provides commands with different notations:

- Prefix notation
- Postfix notation

Prefix Notation Commands

The presentation of these commands in this document is:
command <parameter1> <parameter2>.

The command and the parameters are entered in exactly the sequence shown in the syntax.

Postfix Notation Commands

The presentation of these commands is as follows:
command (parameter1 parameter2 — return).

If a command is presented as shown above you cannot enter the command in the given sequence, you have to enter the parameters before the command itself. If several parameters are given, they have to be entered in the given sequence.

The command led!, for example, has the following syntax:
led! (color freq led# —).

To make user LED 0 flash in red and with moderate speed, enter
`red moderate 0 led!`

Some commands need the parameter <false> or <true>. The values to be entered for <false> / <true> depend on the mode you are using.

Table 1: *True and False Parameters in Binary and Hexadecimal Mode*

Parameter	Binary Mode	Hexadecimal Mode
True	-1	FFFFFFFFFFFFFFFF
False	0	0

To disable the watchdog timer interrupt, for example, enter the command ?ie-wdt (false | true —) as follows: **`0 ?ie-wdt`**

Using NVRAM Editor

To enter into the NVRAM editor and save your inputs, do the following:

1. Open editor by entering **`nvedit`** at OpenBoot ok prompt
2. Enter OpenBoot commands
   Make sure to enter the commands and their parameters in the right sequence (see description on previous pages).
3. To quit editor, press <Ctrl+c>
4. To save your modifications, enter **`nvstore`**
5. To enable script, enter **`setenv use-nvramrc? true`**

Showing Return Values

If a command returns values (shown after the "—" in the syntax), these values are not directly shown on the screen but are put in stacks. The command smc-get-role-info, for example, has the following syntax:
smc-get-role-info (— {host-role smc-role false} | true)

If executed successfully, the command smc-get-role-info returns three values: <host-role>, <smc-role>, and <false>. <host-role> is located in the bottom stack, <smc-role> in the middle stack, and <false> is located in the top stack. OpenBoot provides three commands to show the values in the stacks.

Note: If you enter a further value or command which returns a value, this value will be in the top stack. The return values of the command executed before will move one stack down.

Table 2: *Commands to Show Stack Contents*

Command	Description
.	Displays value located in top stack and deletes it
.s	Displays values of all stacks without deleting them
showstack	Displays values of all stacks automatically after command execution. The stack values are not deleted.

2

Board Configuration

LEDs

The commands in this section can be used to:

- Turn on / off a LED
- Set the color of a LED
- Set the frequency a LED is flashing
- Return LED register settings
- Set LED registers
- Return whether a LED is turned on or off

The following table shows the possible tasks and the corresponding commands in more detail.

Table 3: *LED-Specific Command Overview*

Task		Command	Page
Return	Content of LED 0 Control register	led0-ctrl@	2-5
	Content of LED 1 Control register	led1-ctrl@	2-6
	Content of LED 2 Control register	led2-ctrl@	2-6
	Content of LED 3 Control register	led3-ctrl@	2-7
	Whether a LED is turned on or off	led?	2-4
Set	Color of a LED	led!	2-4
	Flashing frequency of a LED	led!	2-4
	LED 0 Control register	led0-ctrl!	2-5
	LED 1 Control register	led1-ctrl!	2-6
	LED 2 Control register	led2-ctrl!	2-7
	LED 3 Control register	led3-ctrl!	2-7
Turn on / off	Turn a LED off	led-off	2-8
	Turn a LED on	led-on	2-8
	Turn a LED on / off	toggle-led	2-8

led!

DESCRIPTION Makes one of the user LEDs flash in a specified color and with a specified frequency.

SYNTAX `led! (color freq led# — )`

PARAMETERS

Parameter	Description
color	LED color Possible values are: black: LED is turned off green: LED shines green red: LED shines red
led#	User LED number Possible values are: 0: User LED 0 1: User LED 1 2: User LED 2 3: User LED 3
freq	Frequency the LED is flashing Possible values are: no-blinking: LED does not flash slow: LED flashes at 0.5 Hz moderate: LED flashes at 1 Hz fast: LED flashes at 2 Hz

EXAMPLE To make the user LED 0 flash at a moderate frequency in red, enter:

```
red moderate 0 led!
```

led?

DESCRIPTION Returns whether a LED is turned on or off.

SYNTAX `led? (led# — true | false)`

PARAMETERS

Parameter	Description
led#	User LED number Possible values are: 0: User LED 0 1: User LED 1 2: User LED 2 3: User LED 3

RETURNS

Return	Description
true	LED is turned on.
false	LED is turned off.

led0-ctrl@

DESCRIPTION Returns the contents, an 8-bit data, of the LED 0 Control register.

SYNTAX `led0-ctrl@ ( — byte)`

PARAMETERS None

RETURNS <byte>: 8-bit content of the LED 0 Control register. For a description of the return value, refer to the "LED 0 Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

led0-ctrl!

DESCRIPTION Writes an 8-bit data into the LED 0 Control register.

SYNTAX `led0-ctrl! (byte — )`

PARAMETERS <byte>: 8-bit data to be written into the LED 0 Control register. For a register description, refer to the "LED 0 Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

led1-ctrl@

DESCRIPTION Returns the contents, an 8-bit data, of the LED 1 Control register.

SYNTAX `led1-ctrl@ ( — byte)`

PARAMETERS None

RETURNS <byte>: 8-bit content of the LED 1 Control register. For a description of the return value, refer to the "LED 1 Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

led1-ctrl!

DESCRIPTION Writes an 8-bit data into the LED 1 Control register.

SYNTAX `led1-ctrl! (byte — )`

PARAMETERS <byte>: 8-bit data to be written into the LED 1 Control register. For a register description, refer to the "LED 1 Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

led2-ctrl@

DESCRIPTION Returns the contents, an 8-bit data, of the LED 2 Control register.

SYNTAX `led2-ctrl@ ( — byte)`

PARAMETERS None

RETURNS <byte>: 8-bit content of the LED 2 Control register. For a description of the return value, refer to the "LED 2 Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

led2-ctrl!

DESCRIPTION Writes an 8-bit data into the LED 2 Control register.

SYNTAX `led2-ctrl! (byte — )`

PARAMETERS <byte>: 8-bit data to be written into the LED 2 Control register. For a register description, refer to the “LED 2 Control Register” section in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

led3-ctrl@

DESCRIPTION Returns the contents, an 8-bit data, of the LED 3 Control register.

SYNTAX `led3-ctrl@ ( — byte)`

PARAMETERS None

RETURNS <byte>: 8-bit content of the LED 3 Control register. For a description of the return value, refer to the “LED 3 Control Register” section in the *SPARC/CPSB-560 Reference Guide.*

led3-ctrl!

DESCRIPTION Writes an 8-bit data into the LED 3 Control register.

SYNTAX `led3-ctrl! (byte — )`

PARAMETERS <byte>: 8-bit data to be written into the LED 3 Control register. For a register description, refer to the “LED 3 Control Register” section in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

led-off

DESCRIPTION Turns off the specified LED.

SYNTAX `led-off (led# — )`

PARAMETERS

Parameter	Description
led#	User LED number. Possible values: 0: User LED 0 1: User LED 1 2: User LED 2 3: User LED 3

RETURNS None

led-on

DESCRIPTION Turns on the specified user LED.

SYNTAX `led-on (led# — )`

PARAMETERS

Parameter	Description
led#	User LED number. Possible values: 0: User LED 0 1: User LED 1 2: User LED 2 3: User LED 3

RETURNS None

toggle-led

DESCRIPTION Turns the specified user LED on or off. The LED is turned on when it was turned off before, and vice versa. Regardless of the set color, the LED shines green after this command has been used.

SYNTAX `toggle-led (led# — )`

PARAMETERS

Parameter	Description
led#	User LED number. Possible values: 0: User LED 0 1: User LED 1 2: User LED 2 3: User LED 3

RETURNS None

Switch Status

The commands described in the following are used to display the setting of the following hardware switches.

Table 4: *Switch-Specific Command Overview*

Switch No.	Switch Description	Command	Page
All	All	.cpu-switch-stat	2-10
SW 2-1	Boot device selection	plcc-boot?	2-12
SW 2-3	Flash boot area (first MByte) write protection	boot-flash-write-prot?	2-10
SW 2-2	Flash write protection (whole device)	flash-write prot?	2-11
SW 2-4	Reset / abort key	reset-abort-key-ena?	2-11
SW 3-1	Watchdog enable / disable	wd-ena?	2-12

boot-flash-write-prot?

DESCRIPTION Returns the setting of the flash boot area write protection switch (SW2-3) i.e. returns whether the first MByte of the flash (boot area) is write protected.

SYNTAX `boot-flash-write-prot? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	The flash boot area (first MByte of flash) is write protected.
false	The flash boot area (first MByte of flash) is not write protected.

.cpu-switch-stat

DESCRIPTION Displays the current settings of all switches on the SPARC / CPSB-560.

SYNTAX `.cpu-switch-stat ( — )`

PARAMETERS None

RETURN None

EXAMPLE

```
ok .cpu-switch-stat
SW1-1: CPCI Reset ENABLED
SW1-2: Force IPMI PM OFF
SW1-3: Force IPMI SYSEN ACTIVE
SW1-4: Phy selection FRONT
SW2-1: Boot from PLCC
SW2-2: Flash memory write protection OFF
SW2-3: Flash memory write protection for the TSOP boot space OFF
SW2-4: Reset/Abort key ENABLED
SW3-1: Disable watchdog DISABLED
SW3-2: Reserved
SW3-3: Reserved
SW3-4: Reserved
```

flash-write-prot?

DESCRIPTION Returns the setting of the flash write protection switch (SW2-2) i.e. returns whether the complete flash is write protected.

SYNTAX `flash-write-prot? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	The complete flash is write protected.
false	The complete flash is not write protected.

reset-abort-key-ena?

DESCRIPTION Returns the setting of the reset and abort key switch SW2-4, i.e. returns whether the reset and abort keys are enabled.

SYNTAX `reset-abort-key-ena? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Reset / abort key is enabled i.e. switch SW2-4 is set to OFF.
false	Reset / abort key is disabled i.e. switch SW2-4 is set to ON.

plcc-boot?

DESCRIPTION Returns the setting of the boot device selection switch SW2-1, i.e. returns whether the board boots from boot PROM or from flash.

SYNTAX `plcc-boot? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Board boots from boot PROM i.e. SW 2-1 is OFF.
false	Board boots from flash i.e. SW 2-1 is ON.

wd-ena?

DESCRIPTION Returns the setting of the watchdog enable / disable switch SW3-1, i.e. returns whether the watchdog is enabled.

SYNTAX `wd-ena? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Watchdog timer is disabled i.e. SW 3-1 is OFF.
false	Watchdog timer is enabled i.e. SW 3-1 is ON.

Watchdog

The table below shows which tasks can be performed with the watchdog-specific commands.

Table 5: *Watchdog-Specific Command Overview*

Task		Command	Page
Enable / Disable	Watchdog timer interrupt	?ie-wdt	2-14
Return	Whether the watchdog timer interrupt is enabled / disabled	ie-wdt?	2-13
	Whether the watchdog timer has expired	ip-wdt?	2-13
Set	Watchdog timer time-out value	set-wdi!	2-14
Trigger	Watchdog timer	?wdi	2-15

ie-wdt?

DESCRIPTION Returns whether the watchdog timer interrupt is enabled or disabled.

SYNTAX `ie-wdt? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Watchdog timer interrupt is enabled.
false	Watchdog timer interrupt is disabled.

ip-wdt?

DESCRIPTION Returns whether the watchdog timer has expired or not.

SYNTAX `ip-wdt? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Watchdog timer has not expired.
false	Watchdog timer has expired.

set-wdi!

DESCRIPTION Sets the watchdog timer time-out value in the Watchdog Timer Control register.

SYNTAX `set-wdi! (byte — )`

PARAMETERS <byte>: You can choose a time-out value between 125 ms and 60 minutes. For further information on possible time-out values and the necessary hexadecimal values, refer to the "Watchdog Timer Control Register" section in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

?ie-wdt

DESCRIPTION Enables / Disables the watchdog timer interrupt via the Watchdog Timer Interrupt register. After reset, the watchdog timer interrupt is disabled.

SYNTAX `?ie-wdt (true | false — )`

PARAMETERS

Return	Description
true	Enables the watchdog timer interrupt.
false	Disables the watchdog timer interrupt.

RETURNS None

?wdi

DESCRIPTION Triggers the watchdog timer via the Watchdog Timer Trigger Input register.

SYNTAX `?wdi (true | false — )`

PARAMETERS

Return	Description
true	Triggers the watchdog to prevent it from generating an interrupt
false	Does not trigger the watchdog

RETURNS None

Timer

The following table shows the tasks which can be performed with the timer-specific commands.

Table 6: *Timer-Specific Command Overview*

Task		Command	Page
Delete	Status bits of timer 1	clear-timer1	2-17
	Status bits of timer 2	clear-timer2	2-17
Enable / Disable	Timer 1 interrupt	?ie-tim1	2-23
	Timer 2 interrupt	?ie-tim2	2-24
Return	Lower byte of the timer 1 value from the Timer Counter Status register	ltimer1@	2-19
	Lower byte of the timer 2 value from the Timer Counter Status register	ltimer2@	2-20
	Timer mode	en-mod32?	2-17
	Upper byte of the timer 1 value from the Timer Counter Status register	utimer1@	2-22
	Upper byte of the timer 2 value from the Timer Counter Status register	utimer2@	2-22
	Whether timer 1 interrupt is enabled	ie-tim1?	2-18
	Whether timer 2 interrupt is enabled	ie-tim2?	2-18
	Whether timer 1 interrupt is pending	ip-tim1?	2-19
	Whether timer 2 interrupt is pending	ip-tim2?	2-19
	Whether timer 1 is enabled / disabled	timer1?	2-21
	Whether timer 2 is enabled / disabled	timer2?	2-21
Set	Lower byte of the timer 1 initial value in the Timer Initial Control register	ltimer1!	2-20
	Lower byte of the timer 2 initial value in the Timer Initial Control register	ltimer2!	2-20
	Timer mode to 32-bit or 16-bit mode	?en-mod32	2-23
	Upper byte of the timer 1 initial value in the Timer Initial Control register	utimer1!	2-22

Table 6: *Timer-Specific Command Overview (cont.)*

Task		Command	Page
	Upper byte of the timer 2 initial value in the Timer Initial Control register	utimer2!	2-23
Start / Stop	Timer 1	?timer1	2-24
	Timer 2	?timer2	2-25

clear-timer1

DESCRIPTION Clears the status bits of timer 1 in the Timer Status register.

SYNTAX `clear-timer1 ( — )`

PARAMETERS None

RETURNS None

clear-timer2

DESCRIPTION Clears the status bits of timer 2 in the Timer Status register.

SYNTAX `clear-timer2 ( — )`

PARAMETERS None

RETURNS None

en-mod32?

DESCRIPTION Returns whether the timer operates in 32-bit or 16-bit mode.

SYNTAX `en-mod32? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Timer operates in 32-bit mode.
false	Timer operates in 16-bit mode.

ie-tim1?

DESCRIPTION Returns whether the timer 1 interrupt is enabled or disabled. After reset the timer interrupt is disabled.

SYNTAX `ie-tim1? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Timer 1 interrupt is enabled.
false	Timer 1 interrupt is disabled.

ie-tim2?

DESCRIPTION Returns whether the timer 2 interrupt is enabled or disabled. After reset the timer interrupt is disabled.

SYNTAX `ie-tim2? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Timer 2 interrupt is enabled.
false	Timer 2 interrupt is disabled.

ip-tim1?

DESCRIPTION Returns whether an interrupt from timer 1 is pending. After reset, the timer 1 interrupt pending bit in the Interrupt Pending Status register is cleared.

SYNTAX `ip-tim1? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Interrupt from timer 1 is pending.
false	Interrupt from timer 1 is not pending.

ip-tim2?

DESCRIPTION Returns whether an interrupt from timer 2 is pending. After reset, the timer 2 interrupt pending bit in the Interrupt Pending Status register is cleared.

SYNTAX `ip-tim2? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Interrupt from timer 2 is pending.
false	Interrupt from timer 2 is not pending.

ltimer1@

DESCRIPTION Returns the lower byte of the timer 1 value from the Timer Counter Status register indicating how much time remains until an interrupt occurs. After reset this byte is cleared.

SYNTAX `ltimer1@ ( — byte)`

PARAMETERS None

RETURNS <byte>: Lower byte of the timer 1 value from the Timer Counter Status register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

ltimer1!

DESCRIPTION Sets the lower byte of the timer 1 initial value in the Timer Initial Control register, i.e. sets the lower byte of the value from which timer 1 counts to zero. After reset this byte is cleared.

SYNTAX `ltimer1! (byte — )`

PARAMETERS <byte>: Lower byte of the timer 1 initial value in the Timer Initial Control register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

ltimer2@

DESCRIPTION Returns the lower byte of the timer 2 value from the Timer Counter Status register indicating how much time remains until an interrupt occurs. After reset this byte is cleared.

SYNTAX `ltimer2@ ( — byte)`

PARAMETERS None

RETURNS <byte>: Lower byte of the timer 2 value from the Timer Counter Status register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

ltimer2!

DESCRIPTION Sets the lower byte of the timer 2 initial value in the Timer Initial Control register, i.e. sets the lower byte of the value from which timer 2 counts to zero. After reset this byte is cleared.

SYNTAX `ltimer2! (byte — )`

PARAMETERS <byte>: Lower byte of the timer 2 initial value in the Timer Initial Control register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide*.

RETURNS None

timer1?

DESCRIPTION Returns whether timer 1 is enabled or disabled.

SYNTAX `timer1? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Timer 1 is enabled.
false	Timer 1 is disabled.

timer2?

DESCRIPTION Returns whether timer 2 is enabled or disabled.

SYNTAX `timer2? ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Timer 2 is enabled.
false	Timer 2 is disabled.

utimer1@

DESCRIPTION Returns the upper byte of the timer 2 value from the Timer Counter Status register indicating how much time remains until an interrupt occurs. After reset this byte is cleared.

SYNTAX `utimer1@ ( — byte)`

PARAMETERS None

RETURNS <byte>: Upper byte of the timer 1 value from the Timer Counter Status register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

utimer1!

DESCRIPTION Sets the upper byte of the timer 1 initial value in the Timer Initial Control register, i.e. sets the upper byte of the value from which timer 1 counts to zero. After reset this byte is cleared.

SYNTAX `utimer1! (byte — )`

PARAMETERS <byte>: Upper byte of the timer 1 initial value in the Timer Initial Control register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

utimer2@

DESCRIPTION Returns the upper byte of the timer 2 value from the Timer Counter Status register indicating how much time remains until an interrupt occurs. After reset this byte is cleared.

SYNTAX `utimer2@ ( — byte)`

PARAMETERS None

RETURNS <byte>: Upper byte of the timer 2 value from the Timer Counter Status register. For a timer and value description, refer to section “Timer Registers” in the *SPARC/CPSB-560 Reference Guide.*

utimer2!

DESCRIPTION Sets the upper byte of the timer 2 initial value in the Timer Initial Control register, i.e. sets the upper byte of the value from which timer 2 counts to zero. After reset this byte is cleared.

SYNTAX `utimer2! (byte — )`

PARAMETERS <byte>: Upper byte of the timer 2 initial value to be written into the Timer Initial Control register. For a timer and value description, refer to section "Timer Registers" in the *SPARC/CPSB-560 Reference Guide*.

RETURNS None

?en-mod32

DESCRIPTION Sets the timer either in 32-bit or in 16-bit mode. After reset, the timer operates in 16-bit mode.

SYNTAX `en-mod32 (true | false — )`

PARAMETERS

Parameter	Description
true	Timer operates in 32-bit mode.
false	Timer operates in 16-bit mode.

RETURNS None

?ie-tim1

DESCRIPTION Enables / Disables the timer 1 interrupt. After reset, the timer interrupt is disabled.

SYNTAX `?ie-tim1 (true | false — )`

PARAMETERS

Parameter	Description
true	Enables the timer 1 interrupt.
false	Disables the timer 1 interrupt.

RETURNS None

?ie-tim2

DESCRIPTION Enables / Disables the timer 2 interrupt. After reset, the timer interrupt is disabled.

SYNTAX `?ie-tim2 (true | false — )`

PARAMETERS

Parameter	Description
true	Enables the timer 2 interrupt.
false	Disables the timer 2 interrupt.

RETURNS None

?timer1

DESCRIPTION Starts and stops timer 1. After reset, the timer is disabled.

SYNTAX `?timer1 (true | false — )`

PARAMETERS

Parameter	Description
true	Timer 1 is started.
false	Timer 1 is stopped

RETURNS None

?timer2

DESCRIPTION Starts and stops timer 2. After reset, the timer is disabled.

SYNTAX `?timer2 (true | false — )`

PARAMETERS

Parameter	Description
true	Timer 2 is started.
false	Timer 2 is stopped

RETURNS None

Temperatures

OpenBoot provides commands to display the CPU and board temperature and additionally configuration variables to monitor the CPU temperature.

Displaying CPU and Board Temperature

.environment

DESCRIPTION Displays the CPU (junction) and board (ambient) temperature in °C.

SYNTAX `.environment ( — )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok .environment
CPU Ambient Temp at Sensor 0 : 32 degree C
CPU Junction Temp at Sensor 1 : 54 degree C
ok
```

.temp-monitor

DESCRIPTION Displays the CPU and board temperature in °C and gives additional information on the initialization of the temperature sensor MAX 1617.

SYNTAX `.temp-monitor ( — )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok .temp-monitor
Current temp: cpu temp: 54 ambient temp: 32
Status: none
Configuration: operating, alert enabled
Conversion rate: 8 conversions per second
Ambient temp limits: hi: 127 ho: -65
Cpu temp limits: hi: 127 lo: -65
Manufacturer id: 4d Die revision code: 1
```

CPU Temperature Monitoring with Advanced System Monitoring

Advanced System Monitoring (ASM) allows you to monitor the SPARC engine temperature in OpenBoot and Solaris.

Enabling the ASM Function

1. Enter **`setenv env-monitor enabled`**
2. Enter **`reset`**

Setting Temperature Threshold Values

You can set temperature threshold values for two overheating levels:

- Warning level
  In OpenBoot, the firmware informs you via terminal message that the temperature has reached the warning level. In Solaris, the IPMI controller of the SPARC / CPSB-560 informs you via an event message that the upper non-critical threshold has been reached.
- Shutdown level
  In OpenBoot, the firmware informs you via terminal message that the temperature exceeds the shutdown level. In this case you have to shut down the board. In Solaris, the IPMI controller of the SPARC / CPSB-560 informs you via an event message that the upper critical threshold has been reached.

You can change the factory settings of the threshold values in the configuration variables warning-temp and shutdown-temp.

Note:

- **If you change the temperature settings of warning-temp, you automatically change the upper non-critical threshold value in the CPU temperature sensor SDR.**
- **If you change the temperature settings of shutdown-temp, you automatically change the upper critical threshold value in the CPU temperature sensor SDR.**

To set 60°C as warning level, for example, enter
`setenv warning-temp 60`.

To set 80°C as shutdown level, for example, enter
`setenv shutdown-temp 80`.

I²C Based Memory

The following I²C-based memory devices are connected to the SPARC / CPSB-560 via several I²C buses:

- Four ID PROMs with BIB information of:
  - CPU board
  - Memory module
  - IPMI module
  - Rear transition board
- Four SPD PROMs (one for each DIMM)
- Three SROMS (one for each Ethernet controller)

Each of these I²C-based memory devices is accessible via the commands described on the following pages. For a command overview, see table below.

Table 7: *I²C Memory Device-Specific Command Overview*

Task		Command	Page
Copy	Data from the ID PROM to on-board memory	idprom>mem	2-34
	Data from on-board memory to the ID PROM	mem>idprom	2-35
Display	All I²C memory devices	eeprom-info	2-30
	Content of an ID PROM	.bib	2-29
	Device specifier of all I²C devices	eeprom-info	2-30
Read	Content of an I²C device	eeprom-read	2-31
	One byte from an SPD PROM	i2c@	2-33
Select	I²C device	select-i2c	2-35
Write	One byte to an SPD PROM	i2c!	2-34
	S-record data into an I²C device	eeprom-write	2-32

.bib

DESCRIPTION Displays the BIB contents stored in the respective ID PROM.

SYNTAX `.bib (device-specifier — )`

PARAMETERS

Parameter	Description	
device-specifier	Device-specifier of the respective ID PROM. Possible values are:	
	i2c-rom1	CPU board ID PROM
	i2c-rom2	RTB ID PROM
	i2c-rom3	IPMI controller ID PROM
	i2c-rom4	Memory module ID PROM

RETURNS None

EXAMPLE To display the BIB content of the IPMI controller's ID PROM, enter **`i2c-rom3 .bib`**

```
ok i2c-rom3 .bib
IPMI i2c (8bit) control selected
Product Name                              IBMU
Serial                                    211059300109
BIB file ID                               501618
Revision Information
       BIB definition date:               04/10/2002
       Test date:                         08/20/2002
       Last update:                       -
       PCB revision:                      0.1
       Board revision:                    D
Processor
       CPU speed:                         49 MHz
       Attributes:                        -
       L1 cache size:                     -
       L2 cache size:                     -
       L3 cache size:                     -
       Type:                              MIPS
Onboard devices
       Serial:                            c000
       Parallel:                          -
       SCSI/IDE:                          -
       Floppy/USB:                        -
       Bridges:                           -
       NVRAM[1.2]/RTC/kbd/mse/grph/aud:   -
       Special devices:                   e0
Onboard memory
       NVRAM:                             8 KByte
       SDRAM:                             512 KByte
Flash memory
       Device[0]:                         1 MByte
       Device[1]:                         1 MByte
       Device[2]:                         1 MByte
IBMU info
       PCB revision:                      0.1
       GUID:                              ad7ddb6db45311d68001008042109609
ok
```

eeprom-info

DESCRIPTION Displays all devices which are supported by the eeprom commands. Furthermore, it displays the respective device specifiers which are needed for some of the eeprom commands.

SYNTAX `eeprom-info ( — )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok eeprom-info
EEPROM information list
i2c-rom1                512 Byte: board ID-ROM
i2c-rom2                256 Byte: rear transition board ID-ROM
i2c-rom3                256 Byte: IPMI ID-ROM
i2c-rom4                256 Byte: memory module ID-ROM
i2c-rom5                256 Byte: SPD DIMM #0 onboard
i2c-rom6                256 Byte: SPD DIMM #1 onboard
i2c-rom7                256 Byte: SPD DIMM #2 on memory module
i2c-rom8                256 Byte: SPD DIMM #3 on memory module
82540-rom1       256 Byte: GBit ethernet #2 onboard srom
82540-rom2       256 Byte: GBit ethernet #3 onboard srom
82559-rom1       256 Byte: 82559 ethernet #4 onboard srom
ok
```

eeprom-read

DESCRIPTION Reads the contents of the I^2C-based memory device and displays the data in S-record format.

SYNTAX `eeprom-read (src-addr dest-addr size device-specifier — )`

PARAMETERS

Parameter	Description	
src-addr	Start address in the I^2C-based memory device. Possible range: 0..511.	
dest-addr	Start address in on-board memory for the copied data.	
size	Number of bytes to be copied. Possible values: 1..512	
device-specifier	Device-specifier of the respective ID PROM. Possible values are:	
	i2c-rom1	CPU board ID PROM
	i2c-rom2	RTB ID PROM
	i2c-rom3	IPMI controller ID PROM
	i2c-rom4	Memory module ID PROM
	i2c-rom5	CPU board SPD (DIMM 0)
	i2c-rom6	CPU board SPD (DIMM 1)
	i2c-rom7	Memory module SPD (DIMM 2)
	i2c-rom8	Memory module SPD (DIMM 3)
	82540-rom1	8254x GBit Ethernet controller 1 SROM
	82540-rom2	8254x GBit Ethernet controller 2 SROM
	82559-rom1	82559 Ethernet controller SROM

RETURNS None

EXAMPLE To read 100 byte from the CPU board's SPD, enter
`0 10000 100 i2c-rom5 eeprom-read`

```
ok 0 10000 100 i2c-rom5 eeprom-read
SPD DIMM selected
MSG: Begin of S-record
S1130000800404040D0A02480001A05402800808017B
S113001 08F0406010106 0EA0600000140F142D4089
S1130020150815080000000000000000000000000092
S113003000000000000000000000000000000012109A
S1130040303030303030303030003030303030 30DC
S1130050303030303030303030303030000000005C
S1130060303030303030303030303030303030308C
S1130070303030303030303030303030303064 0F69
S1130080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7C
S1130090FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C
S11300A0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5C
S11300B0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4C
S11300C0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3C
S11300D0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2C
S11300E0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1C
S11300F0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0C
S9030000FC
MSG: End of S-record
ok
```

eeprom-write

Caution

Board malfunction
BIB
Overwriting the BIB data in the ID PROM leads to board malfunction. Therefore, do not overwrite the BIB data in the ID PROM.

DESCRIPTION Writes S-record data into an I²C-based memory device.

SYNTAX `eeprom-write (dest-addr src-addr size device-specifier — )`

PARAMETERS

Parameter	Description
dest-addr	Start address in the I²C-based memory device where the data is copied to.
src-addr	Start address of on-board memory space the data in S-record format is copied to.
size	Number of bytes to be copied. Possible values: 1..512

Parameter	Description	
device-specifier	Device specifier of a device Possible values are:	
	i2c-rom1	CPU board ID PROM
	i2c-rom2	RTB ID PROM
	i2c-rom3	IPMI controller ID PROM
	i2c-rom4	Memory module ID PROM
	i2c-rom5	CPU board SPD (DIMM 0)
	i2c-rom6	CPU board SPD (DIMM 1)
	i2c-rom7	Memory module SPD (DIMM 2)
	i2c-rom8	Memory module SPD (DIMM 3)
	82540-rom1	8254x GBit Ethernet controller 1 SROM
	82540-rom2	8254x GBit Ethernet controller 2 SROM
	82559-rom1	82559 Ethernet controller SROM

RETURNS None

i2c@

DESCRIPTION Reads one byte of data from the specified SPD PROM.

SYNTAX

```
i2c@ (addr i2c-slave-addr — data)
```

PARAMETERS

Parameter	Description
addr	Offset within the SPD PROM's address range
i2c-slave-addr	I²C slave address of the SPD PROM For I²C slave addresses, refer to the *SPARC/CPSB-560 Reference Guide.*

RETURNS <data>: One byte read from the SPD PROM.

i2c!

DESCRIPTION Writes one byte of data to the specified SPD PROM.

Caution

Board malfunction
SPD PROM
Overwriting the data in the SPD PROM leads to board malfunction. Therefore, do not overwrite the data in the SPD PROM.

SYNTAX

```
i2c! (addr data i2c-slave-addr — )
```

PARAMETERS

Parameter	Description
addr	Offset within the SPD PROM's address range at which the data is to be stored
data	One byte of data to be transmitted
i2c-slave-addr	I^2C slave address of the SPD PROM For I^2C slave addresses, refer to the *SPARC/CPSB-560 Reference Guide.*

RETURNS None

idprom>mem

DESCRIPTION Copies a number of bytes from an ID PROM to the on-board memory.

Note: Before using this command, select the respective ID PROM with the command select-i2c.

SYNTAX

```
idprom>mem (src-addr dest-addr size — )
```

PARAMETERS

Parameter	Description
scr-addr	Start address of the source data in the ID PROM Possible range: 0..511
dest-addr	Start address for the copied data in the on-board memory
size	Number of bytes to be copied. Possible range: 1..512

RETURNS None

mem>idprom

DESCRIPTION Copies a number of bytes from the on-board memory to an ID PROM.

Note: Before using this command, select the respective ID PROM with the command select-i2c.

SYNTAX

```
mem>idprom (src-addr dest-addr size — )
```

PARAMETERS

Parameter	Description
scr-addr	Start address of the source data in the on-board memory
dest-addr	Start address in the ID PROM Possible range: 0..511
size	Number of bytes to be copied. Possible range: 1..512

RETURNS None

select-i2c

DESCRIPTION Selects an I²C-based memory device. This command must be executed before using the commands idprom>mem and mem>idprom.

SYNTAX

```
select-i2c (device-specifier — )
```

PARAMETERS

Parameter	Description	
device-specifier	i2c-rom1	CPU board ID PROM
	i2c-rom2	RTB ID PROM
	i2c-rom3	IPMI controller ID PROM
	i2c-rom4	Memory module ID PROM
	i2c-rom5	CPU board SPD (DIMM 0)
	i2c-rom6	CPU board SPD (DIMM 1)
	i2c-rom7	Memory module SPD (DIMM 2)
	i2c-rom8	Memory module SPD (DIMM 3)
	82540-rom1	8254x GBit Ethernet controller 1 SROM
	82540-rom2	8254x GBit Ethernet controller 2 SROM
	82559-rom1	82559 Ethernet controller SROM

RETURNS None

EXAMPLE To select the CPSB-560 ID PROM for further operations, enter **`i2c-rom1 select-i2c`**.

Probe Lists

The CPSB-560 devices connected to the PCI buses A-C are probed according to the contents of the probe lists which are defined via NVRAM configuration variables. For each PCI bus there is a probe list configuration variable. The following table shows the variable names and which IDs are used for the devices.

Table 8: *Probe List Variables*

Probe List Variable	Description	
pcia-probe-list	By default, all devices on the PCI bus A are probed. Therefore, all these devices are listed in the pcia-probe-list. The devices are represented in the probe list by IDs which are:	
	Advanced PCI Bridge (APB) Simba	ID 1
	SCSI controller	ID 2
	8254x GBit Ethernet controller 1	ID 3
	8254x GBit Ethernet controller 2	ID 4
pcib-probe-list	By default, all devices on the PCI bus B are probed. Therefore, all these devices are listed in the pcib-probe-list. The devices are represented in the probe list by IDs which are:	
	PCIO-2	ID 1
	82559 Ethernet controller	ID 2
	IDE controller	ID 3
pcic-probe-list	By default, the PMC module on the PCI bus C is probed. The ID is 1.	

To modify the settings, use the command setenv. For example, if you want to probe only the IDE controller and the 82559 Ethernet controller, enter **`setenv pcib-probe-list 2,3`**.

Note: If you reduce the items in the probe lists, be aware that the associated drivers will not be loaded, either.

To use the default values again, enter **`set-default`**.

To see which controllers are currently available in a probe list, enter **`printenv <probelistvariable>`**.
For a description of the ID numbers, see table above.

Keyboard/Mouse Type

OpenBoot provides the possibility to choose between a Sun and a USB type keyboard / mouse if a Sun and a USB keyboard / mouse are installed at the same time.

Table 9: *Keyboard/Mouse Configuration Variables*

Variable	Description	Options
keyboard	Specifies the keyboard type when two keyboards (Sun and USB) are connected to the SPARC / CPSB-560.	SUN (default) USB
mouse	Specifies the mouse type when two mice (Sun and USB) are connected to the SPARC / CPSB-560.	SUN (default) USB

In order to use the USB keyboard, for example, enter **`setenv keyboard USB`**.

Miscellaneous

The commands described in this section can be used to return the FPGA revision and to clear the status bits in the Reset Status register.

lca-id@

DESCRIPTION Returns the contents, an 8-bit data, of the FPGA Revision Status register, i.e. the revision of the FPGA.

SYNTAX `lca-id@ ( — byte)`

PARAMETERS None

RETURNS <byte>: For a value description, refer to the *SPARC/CPSB-560 Reference Guide.*

?reset-clr

DESCRIPTION Clears the status bits in the Reset Status register after a reset has occurred.

SYNTAX `?reset-clr (true | false — )`

PARAMETERS

Return	Description
true	All status bits are cleared.
false	Status bits are not cleared

RETURNS None

3

Flash

Programming the Flash

The 8-MByte flash on the SPARC/CPSB-560 can be used as follows:

a) Completely as user flash with 8 MByte

b) As 1 MByte boot and 7 MByte user flash

You can choose one of the configurations with the switch SW2-1. To obtain an 8-MByte user flash, set switch SW2-1 to OFF. To obtain a 1-MByte boot and a 7-MByte user area, set switch SW2-1 to ON.

The SPARC/CPSB-560 OpenBoot can be used to program an executable image into the flash and to execute it from there. The executable image has to be one of the following:

- Binary image in a.out format
- FORTH program
- FCode program

The commands described in the following can be used to:

Note:

- **Before using these commands, use the command select-flash to select either the boot or user area of the flash for further operations (see section "select-flash" page 3-8).**
- **Before writing data on the flash, set switches SW2-2 and/or SW2-3 to ON to disable the write protection. For the location of the switches, refer to the *SPARC/CPSB-560 Reference Guide*.**

Table 10: *Flash-Specific Command Overview*

Task		Command	Page
Copy	Data from selected area of flash into main memory	flash>move	3-6
	Data from on-board memory into selected area of flash	move>flash	3-7
Display	Messages while flash is erased or programmed	flash-messages	3-6
Erase	Contents of selected area of flash	flash-erase	3-5
	Contents of selected area of flash and fill area with zeros	erase-flash	3-5

Table 10: *Flash-Specific Command Overview (cont.)*

Task		Command	Page
Program	Data into selected area of flash	c!-flash	3-4
	Half-word (16 bits) into the selected area of the flash	w!-flash	3-8
	Word (32 bits) into the selected area of the flash	l!-flash	3-7
Return	Virtual address of boot area of flash	boot-flash-va	3-4
	Virtual address of flash programming window	flash-va	3-7
	Virtual address of user area of the flash	user-flash-va	3-8
	Virtual address of variable flash-messages	flash-messages	3-6
Select	Boot or user area of the flash for further operations	select-flash	3-8

boot-flash-va

DESCRIPTION Returns the virtual base address of the flash boot area.

SYNTAX

```
boot-flash-va ( — vaddr)
```

PARAMETERS None

RETURNS <vaddr>: Virtual address of the flash boot area.

c!-flash

DESCRIPTION Programs data into the selected area of the flash.

SYNTAX

```
c!-flash (byte addr — )
```

PARAMETERS

Parameter	Description
byte	Data to be programmed into the flash
addr	Location at which the data is to be stored in the flash

RETURNS None

erase-flash

DESCRIPTION Erases the selected area of the flash and fills it with zeros.

SYNTAX `erase-flash (device-number — )`

PARAMETERS

Parameter	Description
device-number	Device number Since there is only one flash device, the only possible value is 0.

RETURNS None

fill-flash

DESCRIPTION Fills the selected area of the flash with a particular pattern.

SYNTAX `fill-flash (dest-addr count pattern — )`

PARAMETERS

Parameter	Description
dest-addr	Destination address in the selected area of the flash
count	Number of bytes to be programmed
pattern	One byte of data

RETURNS None

flash-erase

DESCRIPTION Erases a block in the selected area of the flash.

SYNTAX `flash-erase (offset size — )`

PARAMETERS

Parameter	Description
offset	Start address which must be aligned to the 128 KByte blocks
size	Size of area to be deleted which must be a multiple of 128 KByte

RETURNS None

flash-messages

DESCRIPTION Returns the virtual address of the variable flash-messages. With this variable you can define whether messages are displayed while erasing or programming the flash. To obtain the current setting of the variable i.e. whether messages are displayed or not, use the standard OpenBoot command c@ (see *OpenBoot 3.x Quick Reference*).

To prevent messages from being displayed, enter **`flash-messages off`**. To display messages, enter **`flash-messages on`**.

SYNTAX `flash-messages ( — vaddr)`

PARAMETERS None

RETURNS <vaddr>: Virtual address of the variable flash-messages

flash>move

DESCRIPTION Copies a certain amount of bytes from the selected area of the flash into memory.

SYNTAX `flash>move (source-addr dest-addr count — )`

PARAMETERS

Parameter	Description
source-addr	Address within the selected area of the flash from which data is to be copied
dest-addr	Destination address in memory
count	Number of bytes to be copied

RETURNS None

flash-va

DESCRIPTION Returns the virtual base address of the flash programming window. The address is only valid, if the flash has previously been selected with the command select-flash.

SYNTAX `flash-va ( — vaddr)`

PARAMETERS None

RETURNS <vaddr>: Virtual base address of the flash programming window.

l!-flash

DESCRIPTION Programs a word (32 bit) into the selected area of the flash.

SYNTAX `l!-flash (word addr — )`

PARAMETERS

Parameter	Description
word	32 bit of data
addr	Destination address within selected area of the flash

RETURNS None

move>flash

DESCRIPTION Programs the selected area of the flash at a specified address.

SYNTAX `move>flash (source-addr dest-addr count — )`

PARAMETERS

Parameter	Description
source-addr	Data address in memory
dest-addr	Destination address in the selected area of the flash
count	Number of bytes to be programmed

RETURNS None

select-flash

DESCRIPTION Selects either the boot or the user area of the flash and prepares it for programming. No further commands may follow in the same command line.

The number and size of the areas are determined via switch SW2-1. To obtain an 8-MByte user flash, set switch SW2-1 to OFF. To obtain a 1-MByte boot area and a 7-MByte user area, set switch SW2-1 to ON.

The flash programming window is mapped and the virtual base address of the window is stored internally. You can obtain the address by using the command flash-va.

SYNTAX `select-flash (USER | BOOT — )`

PARAMETERS

Parameter	Description
USER	Selects the user area of the flash for further operations
BOOT	Selects the boot area of the flash for further operations. If switch 2-1 is set to OFF, this option is not available.

RETURNS None

user-flash-va

DESCRIPTION Returns the virtual base address of the user flash.

SYNTAX `user-flash-va ( — vaddr)`

PARAMETERS None

RETURNS <vaddr>: Virtual base address of the user flash.

w!-flash

DESCRIPTION Programs a half word (16 bits) into the selected area of the flash.

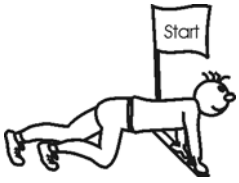
SYNTAX `w!-flash (half-word addr — )`

PARAMETERS

Parameter	Description
half-word	16 bit of data
addr	Destination address in the selected area of the flash

RETURNS None

Example: Programming the User Flash



1. To obtain 1-MByte boot area and 7-MByte user area, set switch SW2-1 to ON
2. Set switch SW2-2 to ON
3. Prepare user flash for programming by entering
   `select-flash USER`

```
ok select-flash USER
USER flash memory is selected for programming
1024 Kbyte BOOT flash memory is available at 0xff550000
7172 Kbyte USER flash memory is available at 0xff350000
```

   The return of the command select-flash informs you that the user area of the flash was made accessible. It displays the available boot and user area of the flash.

4. To erase 128 KByte sector starting at offset 0 in user area which is to be programmed, enter
   `0 h# 20000 flash-erase`

5. To write 128 KByte from memory at offset 10000_{16} to the user area of flash, enter
   `h# 10000 flash-va h# 20000 move>flash`

Flash Device Node Commands

Note: The commands described in this section are only necessary if you do not want to use the command "select-flash" to select the boot or user area of the flash. "Select-flash" already performs all necessary commands to be able to program the flash.

The device tree of OpenBoot contains a device node associated with the flash. This device node includes the standard commands recommended for a device. Before being able to use the commands described in this section you have to select the flash device node by entering one of the following two commands:

a) " flash" select-dev

b) select flash

After having selected the flash device node, you can display all commands available for the flash by entering
words

```
ok words
close          open          selftest      reset      load
write-blocks   read-blocks   seek          write      read
max-transfer   block-size
```

After entering the commands, deselect the current device node by entering
unselect-dev

The table below shows which tasks can be performed with the commands described in this section.

Table 11: *Flash Device Node Specific Command Overview*

Task		Command	Page
Close	Device node	close	3-11
Load	Stand-alone program from the flash into on-board memory	load	3-11
Open	Device node for for further operations	open	3-12
Read	Data from the flash and copy the data to a specified memory area	read	3-12
	Specified number of blocks from the flash and copy the data into the on-board memory.	read-blocks	3-13

Table 11: *Flash Device Node Specific Command Overview (cont.)*

Task		Command	Page
Reset	Flash	reset	3-13
Return	Size of selected area of flash	block-size	3-11
	Size of the largest single transfer the device can perform	max-transfer	3-12
	Whether the device supports write operations	write write-blocks	3-14 3-15
Set	Address pointer within the flash	seek	3-14

block-size

DESCRIPTION Returns the size of the selected flash area in byte which is always the size of the flash programming window.

SYNTAX `block-size ( — bytes)`

PARAMETERS None

RETURNS <bytes>: Size of selected flash area in byte

close

DESCRIPTION Makes all resources (I/O space, memory) available which have been allocated by command open.

SYNTAX `close ( — )`

PARAMETERS None

RETURNS None

load

DESCRIPTION Loads a stand-alone program from the flash beginning at offset 0_{16} and copies it into the on-board memory.

SYNTAX `load (addr — length)`

PARAMETERS <addr>: Address in on-board memory

RETURNS <length>: Number of bytes read from flash

max-transfer

DESCRIPTION Returns the size in byte of the largest single transfer the device can perform.

SYNTAX `max-transfer ( — bytes)`

PARAMETERS None

RETURNS <bytes>: Maximum number of bytes which can be transferred with one transfer. This is always a multiple of the value returned by the command block-size.

open

DESCRIPTION Prepares the package for subsequent use and allocates resources.

SYNTAX `open ( — true | false)`

PARAMETERS None

RETURNS

Return	Description
true	Flash device node has been opened successfully
false	Flash device node has not been opened successfully

read

DESCRIPTION Reads data from the flash and copies it to a specified memory area.

Note: Before using this command, you have to use the command seek to set an address pointer.

SYNTAX `read (addr length — actual)`

PARAMETERS

Parameter	Description
addr	Start address in on-board memory where the data is to be copied to
length	Minimum number of bytes to be read from the flash. The value does not depend on the device's block size.

RETURNS

Return	Description
actual	Indicates whether data was successfully read 0 or negative value: Reading failed 1: Reading successful

read-blocks

DESCRIPTION Reads a specified number of blocks from the flash and copies them into the on-board memory. Each block has the size in byte returned by the command block-size.

SYNTAX `read-blocks (addr block# #blocks — #read)`

PARAMETERS

Parameter	Description
addr	Address in on-board memory where the blocks are copied to
block#	Block number with which block transfer begins
#blocks	Number of blocks to be read

RETURNS <#read>: Number of blocks actually read

reset

DESCRIPTION Puts the flash device into quiet state.

SYNTAX `reset ( — )`

PARAMETERS None

RETURNS None

seek

DESCRIPTION Sets an address in flash from which data is to be read with the command read.

SYNTAX `seek (offset file# — error?)`

PARAMETERS

Parameter	Description
offset	Offset from which data has to be read with command read
file#	Enter any value. This value is ignored and has no effect.

RETURNS

Return	Description
error?	Indicates whether an error has occured. 0: Seek command was successfully executed. -1: An error has occured.

write

DESCRIPTION Discards the parameter values and informs the driver that the flash does not support write operations. This command is provided for compatibility reasons.

SYNTAX `write (addr length — 0A`$_{16}$`)`

PARAMETERS

Parameter	Description
addr	Any value will be discarded
length	Any value will be discarded

RETURNS $0A_{16}$: The flash does not support this command.

write-blocks

DESCRIPTION Discards the parameter values and informs the driver that the flash does not support write operations. This command is provided for compatibility reasons.

SYNTAX `write-blocks (addr block# #blocks — #written)`

PARAMETERS

Parameter	Description
addr	Any value will be discarded
block#	Any value will be discarded
#blocks	Any value will be discarded

RETURNS <#written>: 0_{16} is always returned which means that the flash does not support this function.

Executing Programs from Flash

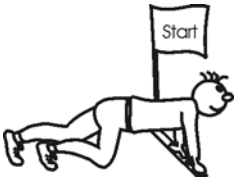
The SPARC / CPSB-560 OpenBoot provides a way to execute an image from the user flash. After having programmed the flash with an image, this image can either be loaded manually or automatically.

Manually

To load an image manually from flash, enter at the OpenBoot ok prompt:
`boot flash`

Automatically

To load an image automatically from user flash, follow the steps described below:



1. If your boot variable is boot-device, enter
   `setenv boot-device flash`
   If your boot variable is diag-device, enter
   `setenv diag-device flash`
2. Enter **`setenv auto-boot? true`**
3. Enter **`reset`**

4

Access to the IPMI Controller

Introduction

The system management controller (SMC) software in the OpenBoot firmware provides access to the on-board Vitesse VSC210 IPMI controller via its IPMI driver. It provides commands to:

- Send and receive IPMI messages
- Access all devices connected to the IPMI controller's private I^2C buses (with BIBs and sensors) and public I^2C buses (IPMB0 and IPMB1)
- Set and retrigger the IPMI controller's internal watchdog
- Retrieve SDR and FRU repository information
- Retrieve sensor data
- Retrieve asynchronous messages, for example, event messages

The commands can be used in the following cases:

- If you have written your IPMI system management software and errors occur, you can enter the commands at the prompt. If the IPMI commands are successfully executed from the prompt, it can be deducted that the problem is not hardware-related.
- If no operating system is used, you can implement the commands in a script which is stored in the NVRAM and which will be executed during power up (see "Entering OpenBoot Commands" page 1-5).

You can communicate with the IPMI controller via two interfaces:

a) CORE User Interface

b) OpenBoot Device Node Interface
Accessing the IPMI controller via this interface is easier than via the CORE User Interface.

Each interface provides different commands to access the IPMI controller, therefore, they are described in two different chapters.

For detailed information on the SMC driver itself, contact your local Motorola representative to obtain a non-disclosure agreement (NDA).

For a description of how to upgrade the IPMI firmware and Sensor Data Records refer to the current version of *IPMI Firmware for SPARC/CPCI-550 and SPARC/CPSB-560 Installation Guide.*

Access via the CORE User Interface

The CORE firmware user interface provides two commands to access the IPMI controller. They are entered at the CORE prompt which is obtained by pressing <Ctrl+u> during power up.

Note: Before using the CORE User Interface commands, you have to create buffers. For information on how to create a buffer for command "set-message-to-bmc", see page 4-8. For information on how to create buffers for command "get-message-from-bmc", see page 4-10.

Table 12: *CORE User Interface Command Overview*

Task	Command	Page
Retrieve asynchronous messages such as event messages	get-message-from-bmc	4-11
Send IPMI commands and IPMI messages via IPMB	set-message-to-bmc	4-9

Supported IPMI Commands

The following table shows the commands supported by the CORE user interface and:

- IPMI command codes (CMD)
- Network function codes (Netfnlun)
- Whether request data is needed for a specific IPMI command
- Whether response data is returned by a specific IPMI command

"CC" in the table below means command Completion Code, "XX" means request data byte, "YY" means response data byte, and "<>" means no request data is needed.

Table 13: *Supported IPMI Commands*

IPMI Command	Parameters Requested <CMD><netfnlun> <requestdata>	Parameters Responded <netfnlun><CMD><CC> <responsedata>
IPMI Device Global Commands		
Get Device ID	01_{16} 18_{16} < >	$1c_{16}$ 01_{16} <CC> <YY . . YY>
Cold Reset	02_{16} 18_{16} < >	$1c_{16}$ 02_{16} <CC>

Table 13: *Supported IPMI Commands (cont.)*

IPMI Command	Parameters Requested <CMD><netfnlun> <requestdata>	Parameters Responded <netfnlun><CMD><CC> <responsedata>
Warm Reset	03_{16} 18_{16} < >	$1c_{16}$ 03_{16} <CC>
Get Selftest Results	04_{16} 18_{16} < >	$1c_{16}$ 04_{16} <CC> <YY YY>
Manufacturing Test On	05_{16} 18_{16} < >	$1c_{16}$ 05_{16} <CC>
Set ACPI Power State	06_{16} 18_{16} <XX XX>	$1c_{16}$ 06_{16} <CC>
Get ACPI Power State	07_{16} 18_{16} < >	$1c_{16}$ 07_{16} <CC> <YY YY>
Get Device GUID	08_{16} 18_{16} < >	$1c_{16}$ 08_{16} <CC> <YY . . YY>
BMC System Interface Commands		
Set BMC Global Enable	$2e_{16}$ 18_{16}<XX>	$1c_{16}$ $2e_{16}$ <CC> <YY . . YY>
Get BMC Global Enable	$2f_{16}$ 18_{16} < >	$1c_{16}$ $2f_{16}$ <CC> <YY>
Clear Message Flag	30_{16} 18_{16} <XX>	$1c_{16}$ 30_{16} <CC>
Get Message Flag	31_{16} 18_{16} < >	$1c_{16}$ 31_{16} <CC> <YY>
Enable Message Channel Receive	32_{16} 18_{16} <XX XX>	$1c_{16}$ 32_{16} <CC> <YY YY>
Master Write Read I^2C	52_{16} 18_{16} <XX . . XX>	$1c_{16}$ 52_{16}<CC> <YY . . YY>
Get IPMB Message	33_{16} 18_{16} < >	$1c_{16}$ 33_{16} <CC> <YY . . YY>
Send IPMB Message	34_{16} 18_{16} <XX . . XX>	$1c_{16}$ 34_{16} <CC>
BMC Watchdog Timer Commands		
Reset Watchdog	22_{16} 18_{16} < >	$1c_{16}$ 22_{16} <CC>
Set Watchdog	24_{16} 18_{16} <XX . . XX>	$1c_{16}$ 24_{16} <CC>
Get Watchdog	25_{16} 18_{16} < >	$1c_{16}$ 25_{16} <CC> <YY . . YY>
Chassis Commands		
Get Chassis Status	01_{16} 00_{16} < >	04_{16} 01_{16} <CC> <YY . . YY>
Chassis Control	02_{16} 00_{16} < >	04_{16} 02_{16} <CC>
POH Counter	$0f_{16}$ 00_{16} < >	04_{16} $0f_{16}$ <CC> <YY . . YY>
SEL Commands		
Get SEL Info	40_{16} 28_{16} < >	$2c_{16}$ 40_{16} <CC> <YY . . YY>

Table 13: *Supported IPMI Commands (cont.)*

IPMI Command	Parameters Requested <CMD><netfnlun> <requestdata>	Parameters Responded <netfnlun><CMD><CC> <responsedata>
Get SEL Allocation Info	41_{16} 28_{16} < >	$2c_{16}$ 41_{16} <CC> <YY . . YY>
Reserve SEL	42_{16} 28_{16} < >	$2c_{16}$ 42_{16} <CC> <YY YY>
Get SEL Entry	43_{16} 28_{16} <XX . . XX>	$2c_{16}$ 43_{16} <CC> <YY . . YY>
Add SEL Entry	44_{16} 28_{16} <XX . . XX>	$2c_{16}$ 44_{16} <CC> <YY YY>
Delete SEL Entry	46_{16} 28_{16} <XX . . XX>	$2c_{16}$ 46_{16} <CC> <YY YY>
Clear SEL	47_{16} 28_{16} <XX . . XX>	$2c_{16}$ 47_{16} <CC> <YY>
Get SEL Time	48_{16} 28_{16} < >	$2c_{16}$ 48_{16} <CC> <YY . . YY>
Set SEL Time	49_{16} 28_{16} <XX . . XX>	$2c_{16}$ 49_{16} <CC>
SDR Repository Commands		
Get SDR Info	20_{16} 28_{16} < >	$2c_{16}$ 20_{16} <CC> <YY . . YY>
Get SDR Allocation Info	21_{16} 28_{16} < >	$2c_{16}$ 21_{16} <CC> <YY . . YY>
Reserve SDR	22_{16} 28_{16} < >	$2c_{16}$ 22_{16} <CC> <YY YY>
Get SDR Entry	23_{16} 28_{16} <XX . . XX>	$2c_{16}$ 23_{16} <CC> <YY . . YY>
Add SDR Entry	24_{16} 28_{16} <XX . . XX>	$2c_{16}$ 24_{16} <CC> <YY YY>
Partial add SDR Entry	25_{16} 28_{16} <XX . . XX>	$2c_{16}$ 25_{16} <CC> <YY YY>
Delete SDR Entry	26_{16} 28_{16} <XX . . XX>	$2c_{16}$ 26_{16} <CC> <YY YY>
Clear SDR	27_{16} 28_{16} <XX . . XX>	$2c_{16}$ 27_{16} <CC> <YY>
Get SDR Time	28_{16} 28_{16} < >	$2c_{16}$ 28_{16} <CC> <YY . . YY>
Set SDR Time	29_{16} 28_{16} <XX . . XX>	$2c_{16}$ 29_{16} <CC>
Run Initialization Agent	$2c_{16}$ 28_{16} <XX>	$2c_{16}$ $2c_{16}$ <CC> <YY>
FRU Inventory Device Commands		
Get FRU Inventory Info	10_{16} 28_{16} < >	$2c_{16}$ 10_{16} <CC> <YY . . YY>
Read FRU Inventory	11_{16} 28_{16} <XX . . XX>	$2c_{16}$ 11_{16} <CC> <YY . . YY>
Write FRU Inventory	12_{16} 28_{16} <XX . . XX>	$2c_{16}$ 12_{16} <CC> <YY>

Table 13: *Supported IPMI Commands (cont.)*

IPMI Command	Parameters Requested <CMD><netfnlun> <requestdata>	Parameters Responded <netfnlun><CMD><CC> <responsedata>
Sensor Device Commands		
Get Device SDR Info	20_{16} 10_{16} < >	14_{16} 20_{16} <CC> <YY . . YY>
Get Device SDR	21_{16} 10_{16} <XX . . XX>	14_{16} 21_{16} <CC> <YY . . YY>
Reserve Device SDR	22_{16} 10_{16} < >	14_{16} 22_{16} <CC> <YY YY>
Get Sensor Reading Factors	23_{16} 10_{16} <XX XX>	14_{16} 23_{16} <CC> <YY . . YY>
Set Sensor Hysteresis	24_{16} 10_{16} <XX . . XX>	14_{16} 24_{16} <CC>
Get Sensor Hysteresis	25_{16} 10_{16} <XX XX>	14_{16} 25_{16} <CC> <YY YY>
Set Sensor Threshold	26_{16} 10_{16} <XX . . XX>	14_{16} 26_{16} <CC>
Get Sensor Threshold	27_{16} 10_{16} <XX>	14_{16} 27_{16} <CC> <YY . . YY>
Set Sensor Event Enable	28_{16} 10_{16} <XX . . XX>	14_{16} 28_{16} <CC>
Get Sensor Event Enable	29_{16} 10_{16} <XX . . XX>	14_{16} 29_{16} <CC> <YY . . YY>
Rearm Sensor Events	$2a_{16}$ 10_{16} <XX . . XX>	14_{16} $2a_{16}$ <CC>
Get Sensor Event Status	$2b_{16}$ 10_{16} <XX>	14_{16} $2b_{16}$ <CC> <YY . . YY>
Get Sensor Reading	$2d_{16}$ 10_{16} <XX>	14_{16} $2d_{16}$ <CC> <YY . . YY>
Set Sensor Type	$2e_{16}$ 10_{16} <XX>	14_{16} $2e_{16}$ <CC>
Get Sensor Type	$2f_{16}$ 10_{16} <XX>	14_{16} $2f_{16}$ <CC> <YY YY>
Set Event Receiver	00_{16} 10_{16} <XX XX>	14_{16} 00_{16} <CC>
Get Event Receiver	01_{16} 10_{16} < >	14_{16} 01_{16} <CC> <YY YY>
Motorola-Specific Commands		
Get Geographical Address	04_{16} 30_{16} <XX . . XX>[1)]	34_{16} 04_{16} <CC> <YY YY>[1)]
Change Role	03_{16} 30_{16} <XX>[1)]	34_{16} 03_{16} <CC>
Set Shadow Repository Enable	05_{16} 30_{16} <XX>[1)]	34_{16} 05_{16} <CC>

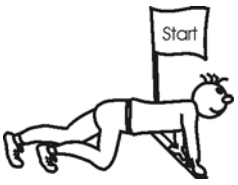
Table 13: *Supported IPMI Commands (cont.)*

IPMI Command	Parameters Requested <CMD><netfnlun> <requestdata>	Parameters Responded <netfnlun><CMD><CC> <responsedata>
Miscellaneous Commands		
Read Event Message Buffer	35_{16} 18_{16} < >	$1c_{16}$ 35_{16} <CC> <YY . . YY>
Firmware Update Start	$1b_{16}$ 20_{16} <XX . . XX>	24_{16} $1b_{16}$ <CC>
Firmware Update Continue	$1c_{16}$ 20_{16} <XX . . XX>	24_{16} $1c_{16}$ <CC>
Firmware Update End	$1e_{16}$ 20_{16} <XX . . XX>	24_{16} $1e_{16}$ <CC>

1) For a description of the request and response values, refer to the *Intelligent Board Management Unit Reference Guide*.

Requirements for Using set-message-to-bmc

Before using the command set-message-to-bmc, you need to create a data buffer to store the request data and the response data returned by an IPMI command. To use the command, do the following:



1. Allocate memory for data buffer of specific size by entering
   `malloc <size>`
   In this example we assume that the returned buffer address is $fffe02800_{16}$
2. If necessary, write request data into data buffer by using command poke which has the syntax poke <address> <cellsize> <data>. To write one byte of request data into the buffer at address $fffe02800_{16}$, for example, enter:
   `poke fffe02800 1 <requestdata>`

3. Execute command set-message-to-bmc
4. To read <responsedata>, if any, use command peek with the syntax peek <address> <cells>. To read 8 byte of response data from $fffe02800_{16}$, for example, enter:
   peek fffe02800 8

set-message-to-bmc

DESCRIPTION Sends IPMI commands or messages and receives the response data, if any. For the supported IPMI commands, refer to "Supported IPMI Commands" page 4-4.

SYNTAX

```
set-message-to-bmc <cmd><netfnlun><addr>[<len>][<rqSA>][<rsSA>]
[<chn>]
```

PARAMETERS

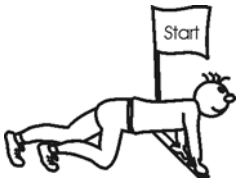
Parameters	Needed for		Description
	IPMI Commands	IPMI Messages	
cmd	x	x	Code for IPMI command For command codes, see Table 13 "Supported IPMI Commands" page 4-4.
netfnlun	x	x	Network function code and logical unit number For network function codes, see Table 13 "Supported IPMI Commands" page 4-4.
addr	x	x	Address of data buffer obtained by command malloc
len	-	x	Size of data buffer

Parameters	Needed for		Description
	IPMI Commands	IPMI Messages	
rqSA	-	x	Requester's slave address Values depend on the geographical address of the board in the system. Refer to the system's backplane description to find the geographical addresses.
			Geographical Address → Slave Address
			0 → Disabled
			1 → $B0_{16}$ (or 20_{16} if configured as BMC)
			2 → $B2_{16}$
			3 → $B4_{16}$
			4 → $B6_{16}$
			5 → $B8_{16}$
			6 → BA_{16}
			7..30 → BC_{16} .. EA_{16}
			31 → EC_{16}
rsSA	-	x	Responder's slave address. For values, see description of <rqSA>.
chn	-	x	I^2C port [0..3]. Possible values are: 00_{16}: IPMB 0 02_{16}: IPMB 1 04_{16}: Private bus 0 (BIBs) 06_{16}: Private bus 1 (Sensor)

Requirements for Using get-message-from-bmc

The command get-message-from-bmc requires three buffers to store:

- Asynchronous message
- Size of message
- IPMI command code



1. For each of the three buffers, enter **`malloc <size>`**
   This command returns the buffer address.
2. Execute command
   `get-message-from-bmc <databuf><len><cmd>`
3. To see whether event messages are present and of which type, enter
   `peek <cmd> 1`
4. To see how many bytes must be read, enter
   `peek <len> 1`
   This command will return the data buffer size.

5. To read event message, enter
   `peek <databuf> <databuffersize>`
   For the meaning of the event message, see the *IPMI Specification*, table 19-1 “SEL Event Records”.

get-message-from-bmc

DESCRIPTION Retrieves asynchronous messages from IPMI controller’s event message buffer.

SYNTAX `get-message-from-bmc <databuf><len><cmd>`

PARAMETERS

Parameter	Description
databuf	Address of data buffer
len	Address to store the size of data buffer
cmd	Address to store the IPMI command

RETURNS None

CORE Help

To get a list of all available general CORE commands and their syntax, enter help at the CORE prompt. The screen output is seen below.

```
> help

To get this help         :  help
To allocate memory buffer:  malloc <size>
To free memory buffer    :  free <addr>
To block copy memory     :  bcopy <src> <dest> <#bytes>
To dump memory           :  dump <addr> <#bytes> [asi]
To read an address       :  [safe-]peek <addr> <1|2|4|8> [asi]
To write to an address   :  poke <addr> <1|2|4|8> <data> [asi]
To load a file           :  load <device> <file-path> <addr>
Jump to an address       :  go <addr>
Execute client           :  execute [client-name]
Print NVRAM data         :  print-nvram
Write to NVRAM variable  :  set-nvram <variable-name|ID> <data>
Read an NVRAM variable   :  get-nvram       <variable-name|ID>
Delete an NVRAM variable :  delete-nvram    <ID>
Set NVRAM vars to default:  set-defaults
Call a trap function     :  trap   <trap#> <par0> ... <par5>
Soft Reset               :  reset
To change input device   :  input-device <tty|kbd>
To initialize PCI        :  init-pci
To show all pci devices  :  show-pci-devs
To show pci config space :  show-pci-space <bus#> <device#> <function#> <offset>
To show pci nexus nodes  :  show-nexus-nodes
To remove a pci device   :  rm-pci-dev <device#>
To add a pci device      :  add-pci-dev <device#>
To remove all pci devices:  rm-pci-devs
To add all pci devices   :  add-pci-devs
To execute UI cmd in loop:  loop <count> <command>
To read from I2C memory  :  i2c-read-byte <slave#> <offset>
To write into I2C memory :  i2c-write-byte <slave#> <offset> <data>
To execute memory-test   :  main-mem-test
To issue a message to SMC:  set-message-to-bmc <cmd> <netfn> <addr> [<len>][<rqSA>][<rsSA>][<chn>]
To get a message from SMC:  get-message-from-bmc <databuf> <*len> <*cmd>

NOTE: All numbers are considered as HEX numbers
```

Figure 1: *CORE User Interface Short Help*

Access via the OpenBoot Device Node Interface

The OpenBoot device tree provides commands to send IPMI commands to the SPARC / CPSB-560 IPMI controller and to send IPMI messages to IPMI controllers on other boards.

Before using the device node interface commands, you have to select the device node associated with the IPMI controller called sysmgmt by entering:

```
" /sysmgmt" select-dev
```

After using the commands, deselect the current device node by entering:

```
unselect-dev
```

The table below shows which task can be performed with which command.

Table 14: *OpenBoot Device Node IPMI Command Overview*

Task		Command	Page
Change	Operation mode of IPMI controller (BMC / PM)	smc-change-role	4-18
Copy	FRU data into memory	smc-read-fru	4-33
	FRU data from memory to a FRU device	smc-write-fru	4-34
Delete	Unread data from the receive message queue and event message buffer	smc-clear-msg-flags	4-25
Display	Event messages incl. time stamp and affected sensor	.sel-info	4-45
	Number of PMs in the system with respective geographical address, product ID, manufacturer ID, IPMI and firmware version	.probe-pm	4-19
	Sensor information: record ID, record type, sensor numbers, sensor type, threshold values, actual sensor values	.sdr-info	4-35
Enable	Message reception of a channel	smc-ena-msg-channel-receive	4-26
Return	Asynchronous messages, e.g. event messages	smc-get-async-message	4-27
	BMC global enables	smc-get-bmc-global-enables	4-27

Table 14: *OpenBoot Device Node IPMI Command Overview (cont.)*

Task		Command	Page
	Geographical address	smc-get-ga	4-18
	I^2C slave address of the event message receiver	smc-get-event-receiver	4-43
	IPMI firmware version, IPMI version and the manufacturer ID of the IPMI controller	smc-get-fw-version	4-20
	IPMI watchdog timer settings	smc-get-wdt	4-22
	Number of sensors	smc-sdr-get-device-sdr-info	4-41
	Operation mode of IPMI controller (BMC / PM)	smc-get-role-info	4-19
	SDR data	smc-sdr-get-device-sdr	4-36
	SDR reservation ID	smc-sdr-reserve-sdr	4-41
	Self-test results of the Motorola IPMI firmware	smc-post-info	4-21
	SEL record from event message buffer	smc-read-event-msg-buf	4-30
	SEL time	smc-get-sel-time	4-45
	Sensor reading data, e.g. current temperature or voltage values	smc-sdr-get-sensor-reading .sdr-info	4-37 4-35
	Threshold values for a sensor	smc-sdr-get-sensor-threshold	4-40
	Whether the receive message queue and event message buffer are cleared	smc-get-msg-flags	4-28
Send	IPMI commands and IPMB messages	execute-smc-cmd	4-16
	IPMI messages	smc-send-message	4-31
	Master Write-Read I^2C command	smc-master-write-read-i2c	4-29
Set	BMC global enables	smc-set-bmc-global-enables	4-32
	I^2C slave address of the event message receiver	smc-set-event-receiver	4-44
	IPMI watchdog timer	smc-set-wdt	4-24

Table 14: *OpenBoot Device Node IPMI Command Overview (cont.)*

Task		Command	Page
	SEL time	smc-set-sel-time	4-46
	Threshold values for a sensor	smc-sdr-set-sensor-threshold	4-42
Start	IPMI watchdog timer	smc-reset-wdt	4-24
Test	IPMI controller and commands	selftest	4-16

Device Node-Specific Commands

close

DESCRIPTION Closes the device node and makes all resources (memory, I/O space) available which have been allocated by the command open.

SYNTAX `close ( ── )`

PARAMETERS None

RETURNS None

open

DESCRIPTION Prepares the node for subsequent use.

SYNTAX `open ( ── true | false)`

PARAMETERS None

RETURNS

Returns	Description
true	Device node was successfully opened
false	Device node was not successfully opened

selftest

DESCRIPTION Executes self-test methods and tests all IPMI commands available in the operation mode the SPARC / CPSB-560 is currently operated in.

SYNTAX

```
selftest ( — false|true)
```

PARAMETERS None

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

execute-smc-cmd

DESCRIPTION Sends IPMI commands and IPMB messages and returns responses from the IPMI commands and IPMB messages. For information on the supported IPMI commands, see Table 13 “Supported IPMI Commands” page 4-4.

SYNTAX

```
execute-smc-cmd ([channel netfnLUN2 rsSA rqSA cmd2] reqN..req0 N cmd
netfnLUN — {resp0 .. respM length false}|true)
```

PARAMETERS

Parameters	Parameter Needed for		Description
	IPMI Commands	IPMB Messages	
channel	-	x	I^2C channel [0 .. 3] 00_{16}: IPMB 0 02_{16}: IPMB 1 04_{16}: Private bus 0 (BIBs) 06_{16}: Private bus 1 (Sensor)
netfnLUN2	-	x	Network function or Logical Unit Number (LUN) for the IPMB command issued to the PM. For the network function value of an IPMI message, see the *IPMI Specification*.
rsSA	-	x	Responder’s slave address For slave addresses, see the parameter description table of command “set-message-to-bmc” page 4-9.

Parameters	Parameter Needed for		Description
	IPMI Commands	IPMB Messages	
rqSA	-	x	Requester's slave address For slave addresses, see the parameter description table of command "set-message-to-bmc" page 4-9.
cmd2	-	x	Code of IPMI command issued to another IPMI controller For the code of the respective command, see the *IPMI Specification*.
reqN..req0	x	x	IPMI / IPMB request data For the request data of the respective IPMI command, refer to the *IPMI Specification*. For the request data of a Motorola OEM command, refer to the *Intelligent Board Management Unit Reference Guide*.
N	x	x	Number of parameters
cmd	x	x	IPMI command code sent to IPMI controller. For the code of the respective command, see the *IPMI Specification*.
netfnLUN	x	x	Network function or LUN

RETURNS

Returns	Bits	Description
resp0..respM	7..0	Response data of the respective IPMI command. For a value description of the response data, see description of the respective command in the *IPMI Specification*.
length	7..0	Amount of bytes returned
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Command was not successfully executed.

Motorola-Specific IPMI Commands

smc-change-role

DESCRIPTION Changes the operation mode of the IPMI controller (BMC or PM).

Note: Use this command with care. For example, it would be possible to set the IPMI controller as BMC, even though the system already has a BMC. As a result, the two IPMI controllers set as BMC might not work or even conflict with each other. If such a mistake happens, change the role again to PM.

SYNTAX `smc-change-role (role — false|true)`

PARAMETERS

Parameter	Description
role	IPMI controller operation mode 0_{16}: BMC 1_{16}: BMC stand-by 2_{16}: PM

RETURNS

Returns	Description
false	Command was executed successfully
true	Accessing the IPMI controller failed

smc-get-ga

DESCRIPTION Returns the geographical address of the board.

SYNTAX `smc-get-ga ( — {ga false}|true)`

PARAMETERS None

RETURNS

Returns	Description
ga	Geographical address
false	Command was executed successfully. The return value is located in the stack beneath.
true	Retrieving the geographical address was not possible.

smc-get-role-info

DESCRIPTION Returns the IPMI controller's operation mode and the CompactPCI role of the board.

SYNTAX

```
smc-get-role-info ( — {host-role smc-role false}|true)
```

PARAMETERS None

RETURNS

Returns	Bits	Description
host-role		CompactPCI role
	7..4	0001: Node board
	3..0	Reserved
smc-role		IPMI controller operation mode
	7..4	Reserved
	3..0	0000: BMC 0001: BMC stand-by 0010: PM
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Accessing the IPMI controller failed

IPM Device Global Commands

.probe-pm

DESCRIPTION Displays all PMs installed in a system and their:

- I^2C slave address of the slot the board is installed in
- Product ID
- Manufacturer ID

- IPMI version
- Firmware version
- Board type

Note: This command can only be executed if the SPARC/CPSB-560 is operated as BMC.

SYNTAX `.probe-pm ( - )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok .probe-pm
IPMBAddr AuxFwVer ProdID ManufID IPMIDev IPMIVer FwVer
------------------------------------------------------------------------
      b6      -      888  480e00      2b       1    95   FORCE/CPCI-560
\
ok
```

smc-get-fw-version

DESCRIPTION Returns the following information:

- IPMI controller's hardware manufacturer identifier
- Product ID
- IPMI version
- Firmware revision
- Sensor and event interface information

SYNTAX `smc-get-fw-version ( — {FirmwareRev IPMIVersion IPMIDevices ManID ProductID length false}|true)`

PARAMETERS None

RETURNS

Returns	Description
FirmwareRev	Firmware revision
IPMIVersion	IPMI version
IPMIDevices	Sensor and event interface information

Returns	Description
ManID	IPMI controller's hardware manufacturer ID
ProductID	Product identifier
length	Amount of bytes returned
false	Command was executed successfully. The return values are located in the stacks beneath.
true	Accessing the IPMI controller failed

smc-post-info

DESCRIPTION Returns the self-test results of the Motorola IPMI firmware. While the self-test is only executed during power up the command smc-post-info can be issued at any time.

SYNTAX `smc-post-info ( — {minor major false}|true)`

PARAMETERS None

RETURNS

Returns	Value (in hex)	Description
minor	0	Return value is 0 if <major> is 55, 56 or ff
		If the return value for <major> is 57, the <minor> return values have the following meaning:
	07	Cannot access SEL device
	06	Cannot access SDR repository
	05	Cannot access BMC FRU
	04	IPMB signal lines do not respond
	03	SDR repository empty
	02	Internal use area of BMC FRU corrupted
	01	BMC update firmware corrupted
	00	BMC operational firmware corrupted
	Device specific	A device-specific value is returned if <major> return value is 58 or other. For the Motorola OEM definitions, refer to section "Self Test" in the *Intelligent Board Management Unit Reference Guide.*

Returns	Value (in hex)	Description
major	55	No error, self-test passed successfully
	56	Self-test not implemented
	57	Corrupted or inaccessible data (<minor> gives further information on the cause)
	58	Fatal BMC hardware error (<minor> gives further information on the cause)
	ff	Reserved
	Other	Device specific internal failure
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	IPMI controller could not be accessed

BMC Watchdog Timer Commands

smc-get-wdt

DESCRIPTION Returns the current settings of the IPMI controller's watchdog timer.

SYNTAX

```
smc-get-wdt ( — {tmr-use tmr-actions pre-timeout tmr-use-flags cnt1
cnt2 act-cnt1 act-cnt2 false}|true)
```

PARAMETERS None

RETURNS

Returns	Bits	Description
tmr-use		Contains the timer use and logging information
	7	1: Not logged 0: Logged
	6	1: Timer has started 0: Timer is stopped
	5..3	Reserved
	2..0	Timer use (logged on expiration) 000_{16}: Reserved 001_{16}: BIOS / FRB2 002_{16}: BIOS / POST 003_{16}: OS load 004_{16}: SMS / OS 005_{16}: Reserved

Returns	Bits	Description
tmr-actions		Holds the interrupt type and the time-out action set
	7	Reserved
	6..4	Pre-time-out interrupt 000_{16}: None 001_{16}: SMI / ABORT 002_{16}: Reserved 003_{16}: Messaging interrupt
	3	Reserved
	2..0	Time-out action 000_{16}: No action 001_{16}: Hard reset 002_{16}: Power down 003_{16}: Power cycle
pre-timeout	7..0	Gives the pre-time-out interval in seconds
tmr-use-flags		Returns the timer use expiration flags
	7..5	Reserved
	4	SMS / OS
	3	OS load
	2	BIOS / POST
	1	BIOS / FRB2
	0	Reserved
cnt1	7..0	Least significant byte of the initial countdown value
cnt2	7..0	Most significant byte of the initial countdown value
act-cnt1	7..0	Least significant byte of the actual countdown value
act-cnt2	7..0	Most significant byte of the actual countdown value
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	IPMI controller could not be accessed

The timer use expiration flags remain valid across BMC reset and system power cycles. These bits are cleared by using the command smc-set-wdt.

smc-reset-wdt

DESCRIPTION Starts and restarts the IPMI controller's watchdog timer.

Note: If a pre-time-out interrupt has been configured, the watchdog cannot be restarted again.

SYNTAX

```
smc-reset-wdt ( — false | true)
```

PARAMETERS None

RETURNS

Returns	Description
false	Command was executed successfully
true	IPMI controller could not be accessed

smc-set-wdt

DESCRIPTION Sets the IPMI controller's watchdog timer and also stops the timer. If the timer is already running, it stops the timer and clears the watchdog pre-time-out interrupt flag.

SYNTAX

```
smc-set-wdt (cnt2 cnt1 tmr-use-flags pre-timeout tmr-actions tmr-use
— false|true)
```

PARAMETERS

Parameter	Bits	Description
cnt2	7..0	Most significant byte of the initial countdown value The watchdog timer increment is one count / 100ms.
cnt1	7..0	Least significant byte of the initial countdown value The watchdog timer increment is one count / 100ms.
tmr-use-flags		Selects the timer use expiration flags
	7..5	Reserved
	4	SMS / OS
	3	OS load
	2	BIOS / POST
	1	BIOS / FRB2
	0	Reserved
pre-timeout	7..0	Sets the pre-time-out interval in seconds

Parameter	Bits	Description
tmr-actions		Sets the interrupt type and the time-out action
	7	Reserved
	6..4	Pre-time-out interrupt 000_{16}: None 001_{16}: SMI / ABORT 002_{16}: Reserved 003_{16}: Messaging interrupt
	3	Reserved
	2..0	Time-out action 000_{16}: No action 001_{16}: Hard reset 002_{16}: Power down 003_{16}: Power cycle
tmr-use		Selects the timer use and configures whether an event will be logged on expiration or not.
	7	1: Do not log event 0: Log event
	6..3	Reserved
	2..0	Timer use (logged on expiration) 000_{16}: Reserved 001_{16}: BIOS / FRB2 002_{16}: BIOS / POST 003_{16}: OS load 004_{16}: SMS / OS 005_{16}: Reserved

RETURNS

Returns	Description
false	Command was executed successfully
true	IPMI controller could not be accessed

BMC System Interface Commands

smc-clear-msg-flags

DESCRIPTION Flushes unread data from the receive message queue and / or from the event message buffer.

SYNTAX `smc-clear-msg-flags (state — false|true)`

PARAMETERS

Parameter	Bits	Description
state	7..4	Reserved
	3	1: Clear watchdog pre-time-out interrupt flag 0: Do not clear watchdog pre-time-out interrupt flag
	2	Reserved
	1	1: Clear event message buffer 0: Do not clear event message buffer
	0	1: Clear receiver message queue 0: Do not clear receiver message queue

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

smc-ena-msg-channel-receive

DESCRIPTION Enables message reception and returns the channel number and the channel state.

SYNTAX

```
smc-ena-msg-channel-receive (ch-state #ch — {#ch ch-state false}|true)
```

PARAMETERS

Parameters	Bits	Description
ch-state	7..2	Reserved
	1..0	00_{16}: Disable channel 01_{16}: Enable channel 02_{16}: Get channel enable / disable state
#ch	7..4	Reserved
	3..0	Channel number

RETURNS

Returns	Bits	Description
#ch	7..4 3..0	Reserved Channel number
ch-state	7..2 1..0	Reserved 00_{16}: Channel disabled 01_{16}: Channel enabled 02_{16}: Get channel enable / disable state
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Command was not successfully executed

smc-get-async-message

DESCRIPTION Returns asynchronous messages from the IPMI controller.

SYNTAX `smc-get-async-message ( — { resp0 .. respM cmd false}|true)`

PARAMETERS None

RETURNS

Returns	Description
resp0..respM	Response data of the IPMI command <cmd> For further information, refer to the response data description of the respective command in the *IPMI Specification*.
cmd	IPMI command code (refer to the *IPMI Specification*)
false	Command was executed successfully. The return values are located in the stacks beneath.
true	Command was not successfully executed.

smc-get-bmc-global-enables

DESCRIPTION Returns the settings of the BMC global enables.

SYNTAX `smc-get-bmc-global-enables ( — {global false}|true)`

PARAMETERS None

RETURNS

Returns	Bits	Description
global	7	1: System event logging enabled 0: System event logging disabled
	6..3	Reserved
	2	1: Event message buffer enabled 0: Event message buffer disabled
	1	1: Event message buffer full interrupt enabled 0: Event message buffer full interrupt disabled
	0	1: Receive message queue interrupt enabled 0: Receive message queue interrupt disabled
false	-	Command was executed successfully. The return value is located in the stack beneath.
true	-	IPMI controller could not be accessed

smc-get-msg-flags

DESCRIPTION Returns the state of the message and event buffer queue.

SYNTAX `smc-get-msg-flags ( — {state false}|true)`

PARAMETERS None

RETURNS

Returns	Bits	Description
state	7..4	Reserved
	3	1: Watchdog pre-time-out interrupt flag is cleared 0: Watchdog pre-time-out interrupt flag is not cleared
	2	Reserved
	1	1: Event message buffer is cleared 0: Event message buffer is not cleared
	0	1: Receiver message queue is cleared 0: Receiver message queue is not cleared
false	-	Command was executed successfully. The return value is located in the stack beneath.
true	-	Accessing the IPMI controller failed

smc-master-write-read-i2c

DESCRIPTION Performs low-level I^2C write, read or write-read accesses to / from devices attached to one of the four I^2C buses of the IPMI controller.

Caution

Board malfunction
BIB in serial EPROM
Overwriting the BIB data leads to board malfunction. Therefore, do not overwrite the BIB data. Before writing data to the serial EPROM read the contents with the command eeprom-read to see which areas of the EPROM are free.

SYNTAX

```
smc-master-write-read-i2c (byteN..byte0 count slave-addr bus-id N — {byte0..byteM false}|true)
```

PARAMETERS

Parameters	Description
byteN..byte0	Read operation: Address of data to be read. If you want to read the temperature value from a sensor, enter the address of the sensor's temperature register (see Example 1 below). Write operation: Data to write and address of device / register (see Example 2 below)
count	Read operation: Number of bytes to be read Write operation: 00_{16}
slave-addr	I^2C slave address of the device For the addresses of the on-board sensor and BIBs, refer to the *SPARC/CPSB-560 Reference Guide.*
bus-id	Bus ID. Possible values are: 00_{16}: IPMB 0 02_{16}: IPMB 1 04_{16}: Private bus 0 (BIBs) 06_{16}: Private bus 1 (sensor)
N	Number of bytes put on the stack

RETURNS

Returns	Description
byte0..byteM	Data read from specified slave address
false	Command was executed successfully. The return value is located in the stack beneath.
true	Command was not successfully executed

EXAMPLE 1 Suppose you want to read the temperature (one byte of data) from the MAX1617 temperature sensor via the temperature register at address 00_{16}. The following table shows the parameter values.

Address Byte 0 <byteN..byte0>	Number of Bytes to read <count>	Slave Address <slave-addr>	Bus ID <bus-id>	<N>
00_{16}	1_{16}	30_{16}	06_{16}	04_{16}

To read the temperature, enter at the ok prompt:

```
0 1 30 6 4 smc-master-write-read-i2c
```

EXAMPLE 2 Suppose you want to write two bytes of data (numbers 20 and 9) to address 00_{16} of the BIB PROM connected to the private I^2C bus 0. The following table shows the Master Write-Read I2C command parameter values.

<byteN..byte0>			Read Count <count>	Slave Address <slave-addr>	Bus ID <bus-id>	<N>
data byteN .. byte2	MSB Address Byte1[1)]	LSB Address Byte0[1)]				
20_{16} 09_{16}	00_{16}	00_{16}	00_{16}	$a0_{16}$	04_{16}	07_{16}

1) Since the serial PROM is a 16-bit I^2C device, two address bytes have to be entered.

To write the two numbers into the EPROM, enter at the ok prompt:

```
20 9 0 0 0 a0 4 7 smc-master-write-read-i2c
```

Note: Only 16-bit addressing is needed for the BIB EPROM. Take this into account if you want to read I^2C data, too.

smc-read-event-msgbuf

DESCRIPTION Returns 16 byte of data in SEL record format from the event message buffer. Reading this buffer automatically clears the full flag returned if executing the command smc-get-msg-flags.

SYNTAX

```
smc-read-event-msgbuf ( — {byte0..byteM false}|true)
```

PARAMETERS None

RETURNS

Returns	Description
byte0..byteM	16 byte of data in SEL record format. For a value description, refer to *Table 14-8 "Read Event Message Buffer"* and *Table 19-1 "SEL Event Records"* in the *IPMI Specification*. If the event was generated by a discrete sensor (ejector switch, POST error, CPCI signal, or critical IRQ sensor), refer also to the *Intelligent Board Management Unit Reference Guide*.
false	Command was executed successfully. The return value is located in the stack beneath.
true	Command was not successfully executed

smc-send-message

DESCRIPTION Sends IPMI messages from the IPMI controller of the SPARC / CPSB-560 to another IPMI controller and receives the IPMI command's response data.

SYNTAX

```
smc-send-message (channel rsSA rqSA cmd [byteN..byte0] N —
{[byte0 .. byteM] false}|true)
```

PARAMETERS

Parameters	Description
channel	Message channel. Possible values are: 00_{16}: IPMB 0 02_{16}: IPMB 1
rsSA	Responder's slave address Values depend on the geographical address of the board in the system. Refer to the system's backplane description to find the geographical addresses.

Geographical Address	Slave Address
0	Disabled
1	$B0_{16}$ (or 20_{16} if configured as BMC)
2	$B2_{16}$
3	$B4_{16}$
4	$B6_{16}$
5	$B8_{16}$
6	BA_{16}
7..30	BC_{16}..EA_{16}
31	EC_{16}

rqSA	Requester's slave address For values, refer to description of <rsSA>.

Parameters	Description
cmd	Command code of IPMI command (refer to the *IPMI Specification*)
byteN..byte0	Request data of respective IPMI command (refer to the *IPMI Specification*)
N	Amount of bytes transmit via parameter values

RETURNS

Returns	Description
byte0..byteM	Response data of sent IPMI command. For a description of the values, refer to the *IPMI Specification*.
false	Command was executed successfully. The return value is located in the stack beneath.
true	Command was not successfully executed

EXAMPLE To issue the IPMI command "Get Device ID" (cmd = 01_{16}) from a PM located at the geographical address 03 (rqSA = $B4_{16}$) to the BMC (rsSA = 20_{16}) via the IPMB bus 0 (channel = 00_{16}), enter the following command:

```
0 20 B4 1 0 smc-send-message
```

Since the command "Get Device ID" does not require request data (see *IPMI Specification*) no data is given for the parameter <byteN..byte0> and the data count <N> is 00_{16}.

In case of successful operation, the IPMI response containing the device ID data <byte0 .. byteM> of the BMC is left on the stack together with the status <false> on top. For a description of <byte0 .. byteM>, refer to the *IPMI Specification*. For a list of device IDs for Motorola CPU boards, refer to the *Intelligent Board Management Unit Reference Guide.*

smc-set-bmc-global-enables

DESCRIPTION Enables message reception into message buffers.

SYNTAX

```
smc-set-bmc-global-enables (global — false|true)
```

PARAMETERS

Parameter	Bits	Description
global	7	1: Enable system event logging 0: Disable system event logging
	6..3	Reserved
	2	1: Enable event message buffer 0: Disable event message buffer
	1	1: Enable event message buffer full interrupt 0: Disable event message buffer full interrupt
	0	1: Enable receive message queue interrupt 0: Disable receive message queue interrupt

RETURNS

Returns	Description
false	Command was executed successfully
true	IPMI controller could not be accessed

FRU Inventory Device Commands

smc-read-fru

DESCRIPTION Stores up to 64 KByte of FRU inventory data from a specified FRU device at a specified area in virtual memory. The command first extracts the FRU inventory size by executing the IPMI command “Get FRU Inventory Area Info” and then executes the IPMI command “Read FRU Inventory Data” 32-byte wise.

SYNTAX `smc-read-fru (vaddr devID — false|true)`

PARAMETERS

Parameters	Description
vaddr	Virtual memory address where the FRU data is to be stored
devID	Device ID of logical FRU device. Possible values are: 0_{16}: SPARC / CPSB-560 1_{16}: IPMI controller 2_{16}: Memory module (if present)

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

EXAMPLE To copy the SPARC / CPSB-560 FRU inventory data to address 100000 in virtual memory, enter at the ok prompt:

```
100000 0 smc-read-fru
```

smc-write-fru

DESCRIPTION Writes a specified amount of FRU inventory data to a logical FRU device.

Note: Before executing this command, the FRU inventory data has to be set within virtual memory at address <vaddr>.

SYNTAX

```
smc-write-fru (vaddr len devID — false|true)
```

PARAMETERS

Parameters	Description
vaddr	Virtual address from which the FRU inventory data is copied
len	Number of bytes of FRU data to be copied
devID	Device ID of logical FRU device. Possible values are: 0_{16}: SPARC / CPSB-560 1_{16}: IPMI controller 2_{16}: Memory module (if present)

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

EXAMPLE To write 100 byte of FRU data from virtual memory at address 10000 to the SPARC / CPSB-560, enter at the ok prompt:

```
10000 100 0 smc-write-fru
```

Sensor Device Commands

.sdr-info

DESCRIPTION Displays the following data of sensors on the SPARC/CPSB-560:

- Record ID
- Record type
- Sensor number
- Sensor type
- Lower and upper critical threshold values (see column “low-crit thres” and “up-crit thres”)
- Actual sensor values (see column “nominal reading”)

SYNTAX `.sdr-info ( - )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok .sdr-info
record  record  sensor  sensor  low-crit  nominal  up-crit  string
  ID     type     No     type     thres   reading   thres
-------------------------------------------------------------------
  1       12       -       -         -       -        -     VSC215
  2       14       -       -         -       -        -
  3        1       0       1         0      25        0     system temp
  4        1       1       1         0      39       55     cpu temp
  5        1       2       2        6f      99       b9     scsi term S0
  6        1       3       2        7b      ab       cc     ethernet 1.5
  7        1       4       2        8b      bf       e8     +1.7V
  8        1       5       2        67      96       ac     ethernet 2.5
  9        1       6       2        86      c5       e0     +12.0V
  a        1       7       2        88      c5       e3     +3.3V
  b        1       8       2        6f      9c       b9     +5.0V
  c        1       9       2        1b      15       2d     -12.0V
  d        1       a       f         0       0        0     post error
  e        1       b      d2         0       0        0     ejector switch
  f        1       c      d2         0       2        0     cpci signal
 10        1       d      d2         0       0        0     crit irq status
 11        1       e      23         0       0        0     watchdog
ok
```

For the meaning of the reading value for the discrete sensors (POST error, ejector switch, CPCI signal, and critical IRQ status sensors), refer to the *Intelligent Board Management Unit Reference Guide.*

smc-sdr-get-device-sdr

DESCRIPTION Returns SDR information for the specified Logical Unit Number (LUN).

Note: This command can be executed with the SDR reservation ID only. With the reservation ID any requestor can recognize whether a record has been changed or not. Therefore, before using this command, execute the command "smc-sdr-reserve-sdr" to get the reservation ID of the SDR (see page 4-41).

SYNTAX

```
smc-sdr-get-device-sdr (count offset msb-record lsb-record
msb-reservation lsb-reservation — {lsb-record msb-record
byte0..byteM false}|true)
```

PARAMETERS

Parameters	Description
count	Number of bytes to be read from an SDR
offset	Offset address of the record
msb-record	Most significant byte of record ID To get the record ID of a SPARC/CPSB-560 sensor, use the command .sdr-info.
lsb-record	Least significant byte of record ID To get the record ID of a SPARC/CPSB-560 sensor, use the command .sdr-info.
msb-reservation	Most significant byte of reservation ID To get the value, use command smc-sdr-reserve-sdr (see page 4-41).
lsb-reservation	Least significant byte of reservation ID To get the value, use command smc-sdr-reserve-sdr (see page 4-41).

RETURNS

Returns	Description
lsb-record	Least significant byte of record ID of the next record
msb-record	Most significant byte of record ID of the next record
byte0..byteM	For a description of the values, refer to chapter "Sensor Data Record Formats" in the *IPMI Specification.*
false	Command was executed successfully. The return values are located in the stacks beneath.
true	Command was not successfully executed

smc-sdr-get-device-sdr-info

DESCRIPTION Returns the number of sensors of a logical SDR sensor device.

SYNTAX
```
smc-sdr-get-device-sdr-info ( — {sensors flags t-stamp3 t-stamp2
t-stamp1 t-stamp0 false}|true)
```

PARAMETERS None

RETURNS

Returns	Bits	Description
sensors	7..0	Number of sensors
flags	7	1: Dynamic sensor population (number of sensors varies) 0: Static sensor population
	6 ..4	Reserved
	3	1: LUN3 has sensors 0: LUN3 has no sensors
	2	1: LUN2 has sensors 0: LUN2 has no sensors
	1	1: LUN1 has sensors 0: LUN1 has no sensors
	0	1: LUN0 has sensors 0: LUN0 has no sensors
t-stamp3	7..0	Timestamp for sensor population change indication (MSB)
t-stamp2	7..0	Timestamp for sensor population change indication (third byte)
t-stamp1	7..0	Timestamp for sensor population change indication (second byte)
t-stamp0	7..0	Timestamp for sensor population change indication (LSB)
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Command was not successfully executed

smc-sdr-get-sensor-reading

DESCRIPTION Returns the current sensor data of a specified sensor.

SYNTAX
```
smc-sdr-get-sensor-reading (sensor — {reading update threshold-
comparism1 threshold-comparism2 false}|true)
```

PARAMETERS <sensor>: Sensor number. Use command .sdr-info to obtain the sensor number of a specific sensor.

RETURNS

Returns	Bit	Description
reading	7..0	Sensor reading For a description of the values returned by the discrete sensors on the SPARC / CPSB-560 (ejector switch, POST error, CPCI signal, or critical IRQ sensors), refer to the *Intelligent Board Management Unit Reference Guide.*
update	7	1: Event messages enabled 0: Event messages disabled
	6	1: Sensor scanning enabled 0: Sensor scanning disabled
	5	1: Initial update in progress
	4..0	Reserved

Returns	Bit	Description
threshold-comparism1		Threshold comparism status. Values depend on whether the sensor is a threshold-based or a discrete sensor.
		Threshold-based sensors:
	7..6	Reserved
	5	1: At or above upper non-recoverable threshold 0: Below upper non-recoverable threshold
	4	1: At or above upper critical threshold 0: Below upper critical threshold
	3	1: At or above upper non-critical threshold 0: Below non-critical threshold
	2	1: At or below lower non-recoverable threshold 0: Above lower non-critical threshold
	1	1: At or below lower critical threshold 0: Above lower critical threshold
	0	1: At or below lower non-critical threshold 0: Above lower non-critical threshold
		Discrete sensors:
	7	1: State 7 asserted 0: State 7 not asserted
	6	1: State 6 asserted 0: State 6 not asserted
	5	1: State 5 asserted 0: State 5 not asserted
	4	1: State 4 asserted 0: State 4 not asserted
	3	1: State 3 asserted 0: State 3 not asserted
	2	1: State 2 asserted 0: State 2 not asserted
	1	1: State 1 asserted 0: State 1 not asserted
	0	1: State 0 asserted 0: State 0 not asserted

Returns	Bit	Description
threshold-comparism2		Discrete sensors (otherwise optional):
	7	Reserved
	6	1: State 14 asserted 0: State 14 not asserted
	5	1: State 13 asserted 0: State 13 not asserted
	4	1: State 12 asserted 0: State 12 not asserted
	3	1: State 11 asserted 0: State 11 not asserted
	2	1: State 10 asserted 0: State 10 not asserted
	1	1: State 9 asserted 0: State 9 not asserted
	0	1: State 8 asserted 0: State 8 not asserted
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Command was not successfully executed

smc-sdr-get-sensor-threshold

DESCRIPTION Returns the threshold values set for a specific sensor and indicates whether these threshold values are enabled.

SYNTAX

```
smc-sdr-get-sensor-threshold (sensor — {setting lower-non-crit-thr
lower-crit-thr lower-non-recov-thr upper-non-crit-thr upper-crit-thr
upper-non-recov-thr false}|true)
```

PARAMETERS <sensor>: Sensor number. Use command .sdr-info to obtain the sensor number of a specific sensor.

RETURNS

Returns	Bit	Description
setting		Indicates whether the threshold values are enabled.
	7..6	Reserved
	5	1: Upper non-recoverable threshold is set 0: Upper non-recoverable threshold is not set
	4	1: Upper critical threshold is set 0: Upper critical threshold is not set
	3	1: Upper non-critical threshold is set 0: Upper non-critical threshold is not set
	2	1: Lower non-recoverable threshold is set 0: Lower non-recoverable threshold is not set
	1	1: Lower critical threshold is set 0: Lower critical threshold is not set
	0	1: Lower non-critical threshold is set 0: Lower non-critical threshold is not set
lower-non-crit-thr	7..0	Lower non-critical threshold
lower-crit-thr	7..0	Lower critical threshold
lower-non-revov-thr	7..0	Lower non-recoverable threshold
upper-non-crit-thr	7..0	Upper non-critical threshold
upper-crit-thr	7..0	Upper critical threshold
upper-non-recov-thr	7..0	Upper non-recoverable threshold
false	-	Command was executed successfully. The return values are located in the stacks beneath.
true	-	Command was not successfully executed

smc-sdr-reserve-sdr

DESCRIPTION Returns the SDR reservation ID which is needed for the command smc-sdr-get-device-sdr.

SYNTAX

```
smc-sdr-reserve-sdr ( — {msb-reservation lsb-reservation false}|
true)
```

PARAMETERS None

RETURNS

Returns	Description
msb-reservation	Most significant byte of reservation ID
lsb-reservation	Least significant byte of reservation ID
false	Command was executed successfully. The return values are located in the stacks beneath.
true	Command was not successfully executed

smc-sdr-set-sensor-threshold

DESCRIPTION Sets threshold values for a specified sensor.

SYNTAX

```
smc-sdr-set-sensor-threshold (upper-non-recov-thr upper-crit-thr
upper-non-crit-thr lower-non-recov-thr lower-crit-thr
lower-non-crit-thr setting sensor — false|true)
```

PARAMETERS

Parameters	Bit	Description
upper-non-recov-thr	7..0	Upper non-recoverable threshold
upper-crit-thr	7..0	Upper critical threshold
upper-non-crit-thr	7..0	Upper non-critical threshold
lower-non-recov-thr	7..0	Lower non-recoverable threshold
lower-crit-thr	7..0	Lower critical threshold
lower-non-crit-thr	7..0	Lower non-critical threshold

Parameters	Bit	Description
setting		Sets a specific threshold
	7..6	Reserved
	5	1: Set upper non-recoverable threshold 0: Do not set threshold
	4	1: Set upper critical threshold 0: Do not set threshold
	3	1: Set upper non-critical threshold 0: Do not set threshold
	2	1: Set lower non-recoverable threshold 0: Do not set threshold
	1	1: Set lower critical threshold 0: Do not set threshold
	0	1: Set lower non-critical threshold 0: Do not set threshold
sensor	7..0	Sensor number Use command .sdr-info to obtain the sensor number of a specific sensor.

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

EXAMPLE To set 85°C as upper non-recoverable threshold, 75°C as upper critical threshold, 60°C as upper non-critical threshold, 0°C as lower non-recoverable threshold, 5°C as lower critical threshold and 10°C as lower non-critical threshold for sensor 1 (CPU temperature sensor), enter:

```
d#85 d#75 d#60 d#0 d#5 d#10 3f 1 smc-set-sensor-thresh-
old
```

Event Commands

smc-get-event-receiver

DESCRIPTION Returns the I^2C address and LUN of the event message receiver.

SYNTAX `smc-get-event-receiver ( — {slave LUN false}|true)`

PARAMETERS None

RETURNS

Returns	Description
slave	I^2C slave address of the event message receiver
LUN	LUN of the event message receiver
false	Command was successfully executed. The return values are located in the stacks beneath.
true	Command was not successfully executed

smc-set-event-receiver

DESCRIPTION Sets the I^2C slave address and LUN of the event message receiver.

SYNTAX `smc-set-event-receiver (LUN slave — false|true)`

PARAMETERS

Parameters	Description
LUN	LUN of the event message receiver
slave	I^2C slave address of the event message receiver Values depend on the geographical address of the board in the system. Refer to the system's backplane description to find the geographical addresses.

Geographical Address	Slave Address
0	Disabled
1	$B0_{16}$ (or 20_{16} if configured as BMC)
2	$B2_{16}$
3	$B4_{16}$
4	$B6_{16}$
5	$B8_{16}$
6	BA_{16}
7..30	BC_{16} .. EA_{16}
31	EC_{16}

RETURNS

Returns	Description
false	Command was executed successfully
true	Command was not successfully executed

SEL Commands

.sel-info

DESCRIPTION Displays all event messages occurred in the system.

Note: This command can only be executed if the SPARC/CPSB-560 is the BMC.

The sensor number indicates which sensor has generated the event. To see to which sensor the sensor number belongs, use the command .sdr-info. For a description of event data sent by a discrete sensor (ejector switch, POST error, CPCI signal, or critical IRQ sensor), refer to the *Intelligent Board Management Unit Reference Guide.*

SYNTAX `.sel-info ( - )`

PARAMETERS None

RETURNS None

EXAMPLE

```
ok .sel-info
record  record      SEL eventlog I2Caddr&channel sensor sensor  event  event
  ID     type              time   of generator   type  number   type   data
-------------------------------------------------------------------------------
   c      2  00/01/1970 00:00:00     20    0      2       9     AS  52 16 17
   d      2  10/11/2002 00:23:49     20    0      1       1     AS  59 55 55
```

smc-get-sel-time

DESCRIPTION Returns the SEL time representing the local time as the number of seconds passed since 00:00:00, January 1, 1970.

SYNTAX `smc-get-sel-time ( — {byte-lsb byte1 byte2 byte-msb false}|true)`

PARAMETERS None

RETURNS

Return Value	Description
byte-lsb	Least significant byte of SEL time
byte1	Second byte of SEL time

Return Value	Description
byte2	Third byte of SEL time
byte-msb	Most significant byte of SEL time
false	Command was successfully executed. The return values are located in the stacks beneath.
true	Command was not successfully executed

smc-set-sel-time

DESCRIPTION Sets the SEL time with four one-byte values representing the local time as the number of seconds passed since 00:00:00, January 1, 1970.

SYNTAX

```
smc-set-sel-time (byte-msb byte2 byte1 byte-lsb — false|true)
```

PARAMETERS

Parameters	Description
byte-msb	Most significant byte of SEL time
byte2	Second byte of SEL time
byte1	Third byte of SEL time
byte-lsb	Least significant byte of SEL time

RETURNS

Return Values	Description
false	Command was successfully executed
true	Command was not successfully executed

IPMI Watchdog Default Values and Configuration

OpenBoot provides configuration variables to start the watchdog timer automatically during power on and to set the time-out value.

Table 15: *IPMI Watchdog Configuration Variables*

Variable	Description
watchdog-ena?	Used to enable / disable the IPMI watchdog timer. By default, the IPMI watchdog is disabled, i.e. <watchdog-ena?> is set to false. To start the IPMI watchdog, enter **`setenv watchdog-ena? true.`**
watchdog-timeout	Used to set the time-out value (in tenths of seconds). By default, the time-out value is 1800 (180 seconds). To change the the value to six minutes, for example, enter **`setenv watchdog-timeout 3600`**.

The table below shows the default values set by the OpenBoot IPMI driver during power up.

Table 16: *Default Values for IPMI Watchdog Timer Set during Power Up*

Byte of IPMI Command "Get Watchdog Timer"	Default Value	Description
2	02_{16}	Timer use: BIOS / POST
3	11_{16}	Pre-time-out interrupt: SMI Time-out action: Hard reset
4	00_{16}	Pre-time-out interval in seconds: 0
5	04_{16}	Timer use flags = BIOS / POST
6	08_{16}	Initial countdown value LSB
7	07_{16}	Initial countdown value MSB

BIOS POST Codes

The OpenBoot kernel sends BIOS POST codes to the KCS1 interface during power up. If a SPARC / CPSB-560 does not boot, the BIOS POST codes can be read, for example, to get information at which phase of POST the SPARC / CPSB-560 encountered an error.

You can read the POST codes with your system management software with the IPMI command “Get Sensor Reading”. The request data you need for this command is the sensor number. To get the number for the BIOS POST code sensor, use the command “.sdr-info” in OpenBoot.

The following table shows the meaning of the POST codes written into response data byte 2 (sensor reading) of the IPMI command “Get Sensor Reading”.

Table 17: *BIOS POST Codes*

Value in Response Data Byte 2 of “Get Sensor Reading” (BIOS POST Code)	Description
09_{16} 1)	Set in POST flag
45_{16} 1)	POST device initialization
49_{16} 1)	Initializes PCI bus and devices
$B1_{16}$ 1)	End of POST
$B2_{16}$	Prepare to boot operating system
$F7_{16}$	POST done

1) Not available if POST was disabled on the SPARC / CPSB-560, i.e. <diag-switch> is set to <false>.

Index

L

M

O

P

R

S

T

U

W