

Chassis Management for Mellanox Switch Systems with Sysfs

User Manual

Rev 1.1

Vadim: to be updated



Table of Contents

[Document Revision History 5](#_Toc429301651)

[About this Manual 6](#_Toc429301652)

[1 Introduction 7](#_Toc429301653)

[1.1 Software Components 7](#_Toc429301654)

[1.2 Hierarchy and Structure 7](#_Toc429301655)

[1.3 Sysfs Initialization and Driver Registration 8](#_Toc429301656)

[2 Virtual SysFS Hierarchy 9](#_Toc429301657)

[2.1 Module Control 9](#_Toc429301658)

[2.1.1 Read Fan Status 9](#_Toc429301659)

[2.1.2 Read Power Supply Status 10](#_Toc429301660)

[2.1.3 Read Power Supply Power Status 10](#_Toc429301661)

[2.2 Fan Control 10](#_Toc429301662)

[2.2.1 Read Fan Speed 10](#_Toc429301663)

[2.2.2 Set Fan Speed 11](#_Toc429301664)

[2.2.3 Get Fan Min Speed 11](#_Toc429301665)

[2.3 Thermal Control 11](#_Toc429301666)

[2.3.1 Read Switch ASIC Temperature 11](#_Toc429301667)

[2.3.2 Read Switch CPU Temperature 12](#_Toc429301668)

[2.3.3 Read Switch Board Temperature 12](#_Toc429301669)

[2.3.4 Read Switch Port Temperature 12](#_Toc429301670)

[2.3.5 Read Switch Power Supply Temperature 13](#_Toc429301671)

[2.4 LED Control 13](#_Toc429301672)

[2.4.1 Control Fan Status LED 13](#_Toc429301673)

[2.4.2 Get Fan LED Capabilities 14](#_Toc429301674)

[2.4.3 Control Power Supply Status LED 14](#_Toc429301675)

[2.4.4 Get Fan LED Capabilities 15](#_Toc429301676)

[2.4.5 Control System Status LED 15](#_Toc429301677)

[2.4.6 Get Fan LED Capabilities 16](#_Toc429301678)

[2.5 CPLD Control 16](#_Toc429301679)

[2.5.1 Read CPLD Version Number 16](#_Toc429301680)

[2.6 EEPROM Control 16](#_Toc429301681)

[2.6.1 Read CPU EEPROM Data 16](#_Toc429301682)

[2.6.2 Read Fan Module EEPROM Data 17](#_Toc429301683)

[2.6.3 Read Power Supply Module EEPROM Data 17](#_Toc429301684)

[2.6.4 Read System Chassis EEPROM Data 17](#_Toc429301685)

[2.7 Control System Reset 18](#_Toc429301686)

[2.7.1 Reset the System 18](#_Toc429301687)

[3 Thermal Control 19](#_Toc429301688)

[3.1.1 Thermal Algorithm 19](#_Toc429301689)

[3.1.2 Starting and Stopping Thermal Algorithm 19](#_Toc429301690)

Document Revision History

Table 1 - Document Revision History

| Revision | Date | Description |
| --- | --- | --- |
| 1.0 | September 8, 2015 | First release |
| 1,1 | December 18, 2018 | Support for new systems |

To b

About this Manual

This manual describes using sysfs in order to control Mellanox switch HW.

Audience

This manual is intended for developers creating management software over Mellanox switches using Mellanox SDK.

Document Conventions

The following lists conventions used in this document.

|  |  |
| --- | --- |
|  | NOTE: Identifies important information that contains helpful suggestions. |

|  |  |
| --- | --- |
|  | CAUTION: Alerts you to the risk of personal injury, system damage, or loss of data. |

|  |  |
| --- | --- |
|  | WARNING: Warns you that failure to take or avoid a specific action might result in personal injury or a malfunction of the hardware or software. Be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents before you work on any equipment. |

# Introduction

Mellanox hw-management package uses a virtual file system provided by the Linux kernel called sysfs.

The sysfs file system enumerates the devices and buses attached to the system in a file system hierarchy that can be accessed from the user space.

The major advantage of working with sysfs is that it makes HW hierarchy easy to understand and control without having to learn about HW component location and the buses through which they are connected.

## Software Components

Figure 1 presents the software architecture layout and Figur 2 presents layer separation for sysfs support.



Figure 1 System architecture layout



u

dev

rules handler

mellanox

-

system

-

event.sh handler

Figure 2 Sysfs layout

## Hierarchy and Structure

The package uses the Linux default hierarchy structure of sysfs under the directory /sys.

This path is used by existing applications that use auto-discovery to find existing HW components. Two examples for such applications are:

* libsysfs – the libraries provide a consistent and stable interface for querying system device information exposed through sysfs.
* systool – a utility built upon libsysfs that lists devices by bus, class, and topology.

The disadvantage of using this path is that the hierarchy model includes the BUS type and location model which is subject to change between different system types.

To resolve this limitation, the virtual hierarchy structure that is not HW dependent is supported. This hierarchy is a collection of soft links to the default sysfs structure. This document describes the way to work with this hierarchy in order to control the HW.

Chassis attributes information exported through sysfs can be utilized by a number of standard Linux tools. So, for example, the following are tools from the Linux packages lm-sensors and fancontrol, which are capable of operating on top of sysfs infrastructure:

* pwmconfig – tests the pulse width modulation (PWM) outputs of sensors and configures fancontrol
* fancontrol – automated software-based fan speed regulation
* sensors – print sensors information

## Sysfs Initialization and Driver Registration

As describe in the previous sections, sysfs structure provide access to HW drivers. These drivers need to be initialized before using sysfs. In addition, Mellanox virtual hierarchy also needs to be created in order to use it.

The package provides a simple way to initialize the drivers using the set of the shell scripts. These scripts support initialization and de-initialization of driver, virtual hierarchy structure, udev events handling, based on a set of Mellanox system specific udev rules.

Package contains the following files, used within the workload:

* /lib/systemd/system/hw-management.service: system entries for thermal control activation and de-activation.
* /lib/udev/rules.d/50-hw-management-events.rules: udev rules defining the triggers on which events should be handled. When trigger is matched, rule data is to be passed to the event handler (see below file /usr/bin/hw-management-events.sh).
* /usr/bin/hw-management-control.sh: contains thermal algorithm implementation.
* /usr/bin/hw-management-chassis-events.sh and /usr/bin/hw-management-thermal-events.sh: handle udev triggers, according to the received data, it creates or destroys symbolic links to sysfs entries. It allows to create system independent entries and it allows thermal controls to work over this system independent model. Raises signal to hw-management-control in case of fast temperature decreasing. It could happen in case one or few very hot port cables have been removed. Sets PS units internal FAN speed to default value when unit is connected to power source.
* /usr/bin/hw-management.sh: performs initialization and de-initialization, detects the system type, connects thermal drivers according to the system topology, activates and deactivates thermal algorithm.
* /usr/bin/hw-management-led-state-conversion.sh and /usr/bin/hw-management-power-helper.sh: helper scripts.
* /etc/modprobe.d/hw-management.conf and /etc/modules-load.d/hw-management-modules.conf: configuration for kernel modules loading.



For more details follow package README file.

# Virtual SysFS Hierarchy

Mellanox virtual hierarchy supports the following HW control ($bsp\_path below is a location of virtual SysFS hierarchy, in standard Linux distributions, like Debia, RedHat, Fedora, etcetera this is /var/run/hw-management folder, in custom distribution, like Sonic or ONL this is /bsp folder).

Table 2 - Mellanox Hierarchy Node Support

|  |  |
| --- | --- |
| Node Path | Purpose |
| $bsp\_path/module | Gets information on module status (e.g. insert or removed) |
| $bsp\_path/fan | Sets/gets fans speed and minimum allow speed |
| $bsp\_path/thermal | Gets information from thermal sensors |
| $bsp\_path/power | Gets information from power sensors |
| $bsp\_path/led | Sets/gets LED color |
| $bsp\_path/cpld | Gets CPLD version information |
| $bsp\_path/eeprom | Gets raw data from EEPROM in system modules |
| $bsp\_path/system | Controls system reset |

Detailed information on each of these nodes can be found in the following sections.

## Module Control

### Read Fan Status

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/module/fan<fan module number>\_status | | |
| Description | Read fan module status | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Status | Integer | 0 – not present  1 – present |
| Example | Get fan module 1 status: **cat $bsp\_path/module/fan1\_status** | | |

### Read Power Supply Status

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/module/psu<power supply module number>\_status | | |
| Description | Read power supply module status | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Status | Integer | 0 – not present  1 – present |
| Example | Get power supply 1 status: **cat $bsp\_path/module/psu1\_status** | | |

### Read Power Supply Power Status

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/module/psu<power supply module number>\_pwr\_status | | |
| Description | Read power supply power status | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Status | Integer | 0 – no power or cable not present  1 – power present |
| Example | Get power supply 1 status: **cat $bsp\_path/module/psu1\_pwr\_status** | | |

## Fan Control

### Read Fan Speed

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/fan/fan\_<fan number>\_get | | |
| Description | Read fan speed information in RPM | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Fan speed | Integer | 0-23000 RPM |
| Example | Get fan 1 speed: **cat $bsp\_path/fan/fan1\_speed\_get** | | |

### Set Fan Speed

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/fan/fan<fan number>\_set | | |
| Description | Set fan speed in parentage of maximum speed | | |
| Access | Write only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Fan speed | Integer | 0-255 (PWM duty cycles) |
| Example | Set fan module 1 to 60% (153 PWM duty cycles) of top speed.:  **echo 60 > $bsp\_path/fan/fan1\_speed\_set**  **Note:** Setting fan speed is allowed per fan module and not per fan unit inside a module.  Note: PWM to percent mapping is 0-100% to o-255 PWM. | | |

### Get Fan Min Speed

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/fan/fan<fan module number>\_min | | |
| Description | Get fan minimum allow speed in parentage of maximum speed | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Fan speed | Integer | 0-23000 RPM |
| Example | Get fan module 1 minimum speed: **cat $bsp\_path/fan/fan1\_speed\_min** | | |

## Thermal Control

### Read Switch ASIC Temperature

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/thermal/asic | | |
| Description | Read value of switch module ASIC temperature | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Temperature | Integer | Degrees in Celsius |
| Example | Get switch module ASIC temperature: **cat $bsp\_path/thermal/asic** | | |

### Read Switch CPU Temperature

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/thermal/cpu\_<core0 | core1 | pack> | | |
| Description | Read value of CPU module temperature | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Temperature | Integer | Degrees in Celsius |
| Example | Get CPU  Core 0 temperature:  **cat $bsp\_path/thermal/cpu\_core0** | | |

### Read Switch Board Temperature

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/thermal/board\_amb | | |
| Description | Read value of switch board ambient temperature | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Temperature | Integer | Degrees in Celsius |
| Example | Get switch board ambient temperature:  **cat $bsp\_path/thermal/board\_amb** | | |

### Read Switch Port Temperature

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/thermal/port\_amb | | |
| Description | Read value of switch port ambient temperature | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Temperature | Integer | Degrees in Celsius |
| Example | Get switch board ambient temperature:  **cat $bsp\_path/thermal/port\_amb** | | |

### Read Switch Power Supply Temperature

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/thermal/psu<psu module number> | | |
| Description | Read value of power supply temperature | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| Temperature | Integer | Degrees in Celsius |
| Example | Get switch power supply 1 temperature:  **cat $bsp\_path/thermal/psu1** | | |

## LED Control

### Control Fan Status LED

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_fan<fan module number> | | |
| Description | Read/write fan module status LED | | |
| Access | Read / Write | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED color | Integer | green\_blink red\_blink green red  none |
| Example | Get fan module 1 status LED color:  Vadim:  $bsp\_path/led/led\_fan1\_state(to fill state) **cat $bsp\_path/led/fan1 Vadim: is not correct should be $bsp\_path/led/led\_fan1**  Set fan module 1 LED color to green:  **echo green > $bsp\_path/led/led\_fan1 Vadim: echo 1 > $bsp\_path/led/fan1\_green** | | |

### Get Fan LED Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_fan<fan module number>capability | | |
| Description | Read fan module status LED | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED capabilities | Integer | green\_blink red\_blink green red  none |
| Example | Get fan module 1 capabilities: **cat $bsp\_path/led/fan1\_capability** | | |

### Control Power Supply Status LED

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_psu< power supply module number> | | |
| Description | Set/get power supply module status LED | | |
| Access | Read / Write | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED color | Integer | green\_blink red\_blink green red  none |
| Example | Get power supply module 1 status LED color:  Vadim: $bsp\_path/led/led\_psu1\_state(to fill state)  **cat $bsp\_path/led/psu1 Vadim: is not correct should be $bsp\_path/led/psu1\_led**  Set power supply module 1 LED color to green:  **echo green > $bsp\_path/led/led\_psu1 Vadim: echo 1 > $bsp\_path/led/led\_psu1\_green** | | |

### Get Fan LED Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_psu< power supply module number>capability | | |
| Description | Set/get power supply module status LED | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED capabilities | Integer | green\_blink red\_blink green red  none |
| Example | Get power supply module 1 capabilities:  **cat $bsp\_path/led/psu1\_capability** | | |

### Control System Status LED

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_system | | |
| Description | Set/get system status LED | | |
| Access | Read / Write | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED color | Integer | green\_blink red\_blink green red  none |
| Example | Get system status LED color:  $bsp\_path/led/led\_status\_state(to fill state) **cat $bsp\_path/led/led\_status**  Set system LED color to green:  **echo green > $bsp\_path/led/psu1 Vadim: change to**  **echo 1 > $bsp\_path/led/led\_status\_green** | | |

### Get Fan LED Capabilities

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/led/led\_system\_capability | | |
| Description | Set/get system status LED | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| LED capabilities | Integer | green\_blink red\_blink green red  none |
| Example | Get system status LED capabilities: **cat $bsp\_path/led/led\_system\_capability** | | |

## CPLD Control

### Read CPLD Version Number

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/cpld/< cpld\_brd | cpld\_mgmt | cpld\_port>\_version | | |
| Description | Read cpld version number | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| CPLD version | Integer | >= 0 |
| Example | Get management board CPLD version:  **cat $bsp\_path/cpld/cpld\_brd\_version** | | |

## EEPROM Control

### Read CPU EEPROM Data

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/eeprom/cpu\_info | | |
| Description | Read CPU raw data in hex format | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| EEPROM information | Hex | Hex dump format of memory |
| Example | Get CPU EEPROM information:  **cat $bsp\_path/eeprom/cpu\_info** | | |

### Read Fan Module EEPROM Data

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/eeprom/fan<fan module number>\_info | | |
| Description | Read fan module raw data in hex format | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| EEPROM information | Hex | Hex dump format of memory |
| Example | Get fan module 1 EEPROM information:  **cat $bsp\_path/eeprom/fan1\_info** | | |

### Read Power Supply Module EEPROM Data

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/eeprom/psu<power supply module number>\_info | | |
| Description | Read power supply module raw data in hex format | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| EEPROM information | Hex | Hex dump format of memory |
| Example | Get power supply module 1 EEPROM information:  **cat $bsp\_path/eeprom/psu1\_info** | | |

### Read System Chassis EEPROM Data

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/eeprom/vpd\_info | | |
| Description | Read system chassis raw data in hex format | | |
| Access | Read only | | |
| Release version | 1.0 | | |
| Arguments | Name | Data type | Values |
| EEPROM information | Hex | Hex dump format of memory |
| Example | Get system chassis EEPROM information:  **cat $bsp\_path/eeprom/vpd\_info** | | |

## Control System Reset

### Reset the System

|  |  |  |  |
| --- | --- | --- | --- |
| Node name | $bsp\_path/system/pwr\_cycle | | |
| Description | Reset the system | | |
| Access | Write only | | |
| Release version | 1.0 | | |
| Arguments  Vadim: should be N/A (no arguments) | Name | Data type | Values |
| Reset | Hex | 1 – reset the system |
| Example | Reset the switch system:  **cat $bsp\_path/system/power\_cycle Vadim: changed $bsp\_path/system/pwr\_cycle** | | |

# Thermal Control







### The thermal algorithm controls is described in a separate document.