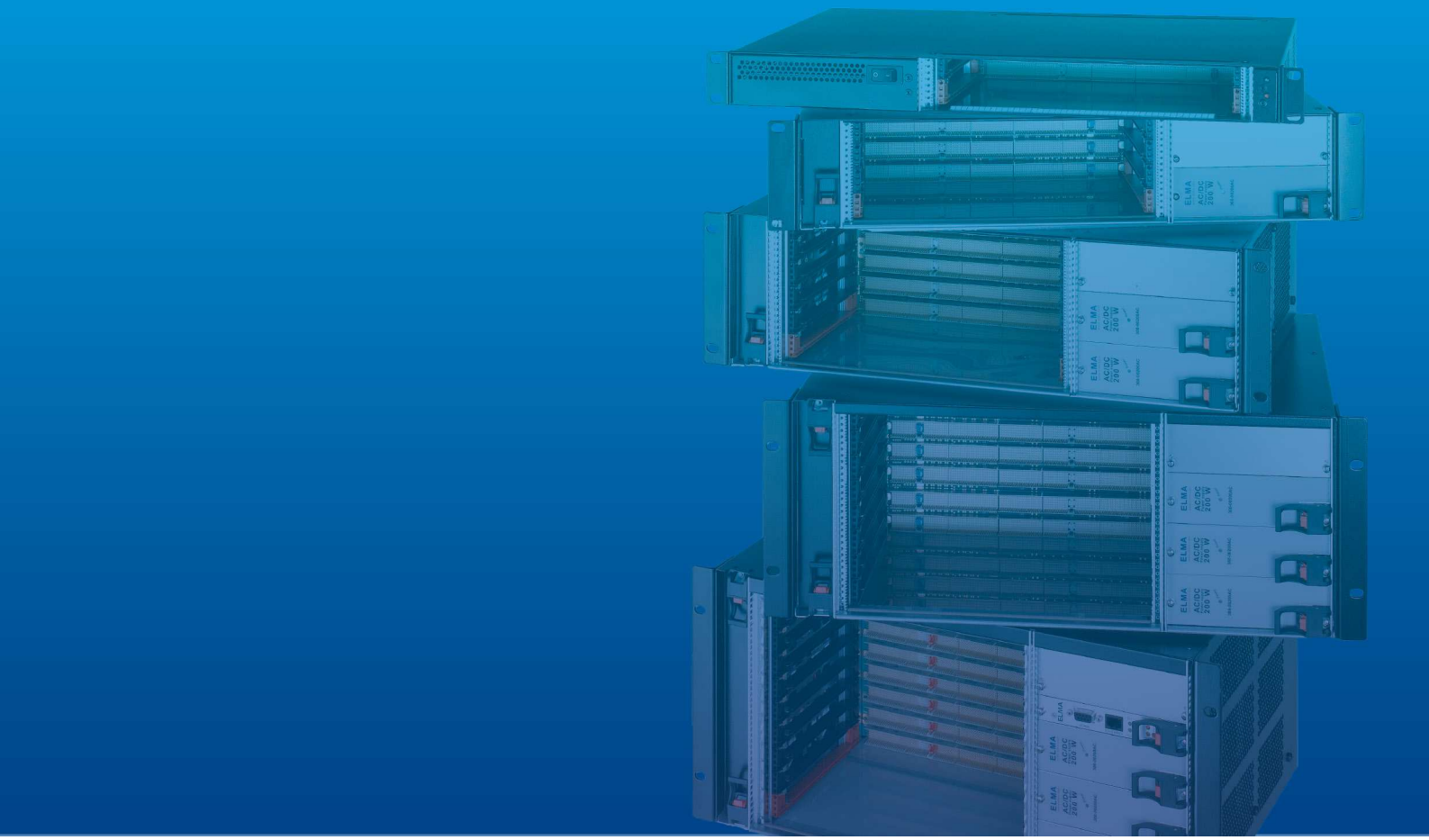

Sysmon3Gen User Manual
Revision 1.17

23.04.14

Table of Contents

List of Figures

List of Tables

1 Product description and functions

The Sysmon is a platform-independent system monitoring unit, which monitors the internal parameters of a System Platform such as voltages, currents, temperatures, and fan speeds.

The monitored parameters of the System Platform are measured, or read in, and analyzed at regular intervals. If a parameter value exceeds or falls below a user-defined limit, the Sysmon detects this as an "Event". In principle, fixed predefined actions are carried out when the corresponding events occur. These actions are described later. It is also possible to use such Events to trigger one or more of the digital Outputs.

The Sysmon uses Sensor Data Records, compliant to IPMI 2.0, to describe the monitored System parameters. Using SDRs a name can be specified for every monitored parameter,digital inputs and outputs. For any measured parameter up to 6 thresholds can be defined: lower non critical, lower critical, lower non-recoverable, upper non critical, upper critical and upper non-recoverable

For the pluggable version, any limit infringements occurring for 3.3V, 5V, 12V, -12V voltages, temperatures and fans are signaled via the front-panel LEDs. The measured values are retrievable at any time via the RS232 serial interface and via Telnet. In addition, limits and system parameters can be changed at any time with the unit in service. As a result, the Sysmon – and hence the connected system - can be controlled and monitored online via any computer with an Internet connection.

To complete the information spectrum, the integrated web page, which the user can design himself, is also continuously updated with the measured values. When the Sysmon is used in VME systems, it generates the ACFAIL* and SYSRESET* VME signals in accordance with VME specification ANSI/VITA 1-1994.

1.1 Voltage monitoring

- Monitoring of up to 8 voltages (+3.3V,+5V, +12V, -12V and 4 user defined voltages[1]).
- Up to 6 thresholds can be configured individually for each voltage
- Status LED for voltages : +3.3V,+5V, +12V, -12V.
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Monitored voltage ranges of the individual inputs:
  - V1(+3.3V input): 0..5V
  - V2(+5V input): 0..8V
  - V3(+12V input): 0..16.5V
  - V4(-12V input): -14..2.5V
  - V5(positive input): 0..5V

1 The 4 Analog Inputs used to monitor the 4 user defined voltages are shared with the Current monitoring Interface. For each of the 4 analog Inputs the Sysmon can monitor either a voltage or a current depending on the software and hardware configuration.

- V6(positive input): 0..5V
- V7(positive input): 0..5V
- V8(positive input): 0..5V

The ranges for the monitored voltages depend on the resistors assembled.

- Other configurations are available on request.
- Voltage monitoring parameters can be changed via the CLI or by upgrading the Sensor Data Records (SDRs) .
- CLI Commands for the voltage monitoring:

  Voltage
  sensor

1.2 Current monitoring

- Monitoring of up to 4 currents[2].
- Up to 6 thresholds can be configured individually for each current.
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Current monitoring parameters can be changed via the CLI or by upgrading the Sensor Data Records (SDRs) .
- CLI Commands for the current monitoring:

  current
  sensor

1.3 Temperature monitoring

- Monitoring of up to 6 analog temperature sensors (10 Kohm NTC thermistors with ß=3950) and 8 I2C TMP75 sensors
- Up to 6 thresholds can be individually configured for each temperature sensor.
- Temperature range from –20°C to +100°C
- Temperature measurement accuracy +/-3°C (max.)
- Status LED on the front panel for temperature error indication
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Any temperature sensor can be used in the fan control algorithm(The user can choose for each temperature sensor the fan control groups the sensor is active for.)
- Commands for the temperature monitoring:

  temp
  sensor

2 For measuring currents an auxiliary part has to be used: 041-970. The 4 Analog Inputs used to monitor the 4 currents are shared with the Voltage monitoring Interface. For each of the 4 analog inputs the Sysmon can monitor either a voltage or a current depending on the software and hardware configuration.

1.4 Fan monitoring and control

- Monitoring and control of up to 12 fans. The fans require a tachometer signal output as well as a PWM signal input.
- Up to 6 thresholds can be individually configured for each fans speed.
- Speed control via 4 fan control groups
- Control of non-PWM fans possible with FanCon add-on module
- Active High rotor locked fans can also be monitored.
- LED on the front panel for indication of fan errors
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Commands for the fan monitoring:

fan
sensor

1.4.1 Fan control

The Speed of the fans can be controlled using one of the 4 fan control groups. Each group controls an independent PWM3 signal using a user-defined temperature-speed characteristic.

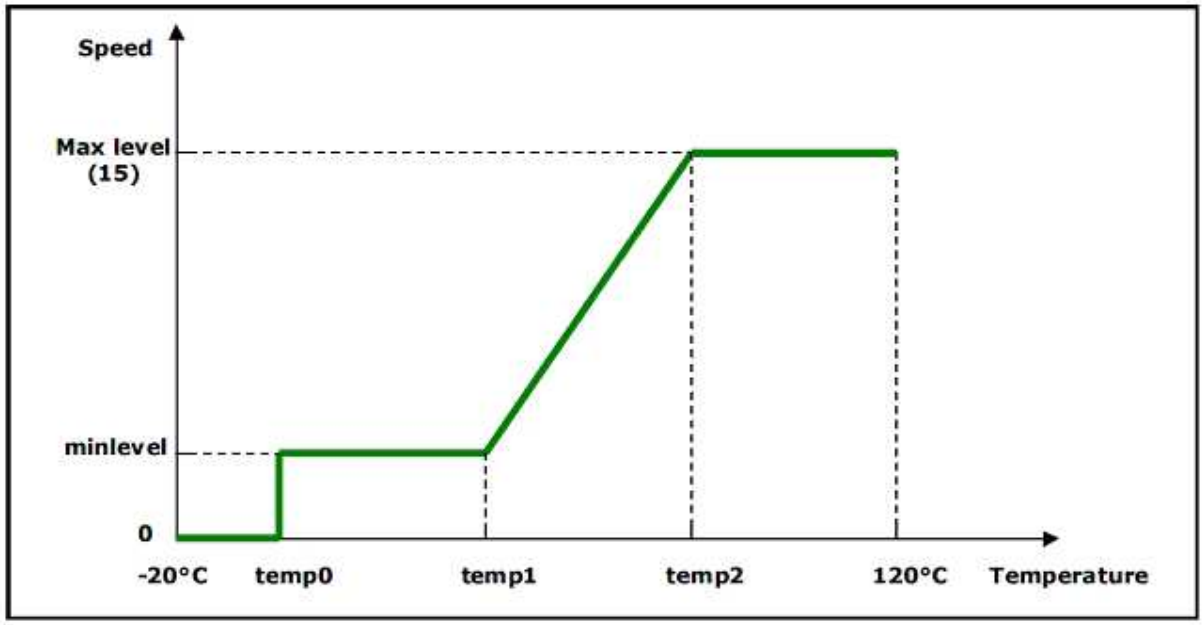

Figure 1: Fan Control Algorithm

The user can define the temperature-speed characteristic using 4 parameters: minlevel (the minimum level at which the fans operate) and 3 temperature thresholds (temp0, temp1, temp2).

The speed of the fans is split in 15 equal levels. At level 0 the fans are stopped and at level 15 the fans are running full speed.

3 The Sysmon can output 4 PWM signals with frequencies between 7Hz and 125KHz. The frequency of PWM1 is equal to that of PWM2, and the frequency of PWM3 is equal to that of PWM4, but the duty factors of the 4 PWM signals are independent.

The 3 temperature thresholds split the operating range of the fan in 4 control regions:

Temperature T	Fans Behavior:
T< temp0	Stopped
temp0 < **T** < temp1	Running at minlevel
temp1<**T**<temp2	Running at a speed level proportional to **T**
temp2<**T**	Running at full speed

Table 1: Fan Control Algorithm

The Fan control algorithm can use any of the installed temperature sensors. The user can choose for each temperature sensor the fan control groups the sensor is active for.
The temperature that drives the fan control algorithm is the maximum value of the temperature sensors active for the respective fan control group.
If no temperature sensors are active for a particular fan group, the fan control algorithm sets the fans to run at the maximum level.
The 4 fan control groups are independent, each have their own minlevel, temp0, temp1, temp2 parameters and each control a different PWM signal.
The fan control can also be disabled. If the fan control is disabled the respective fan group is controlled by the manual fan level. Each fan control group has it's on own manual fan level which can be changed using the CLI via RS232 or Telnet.

Commands useful for fan control:

fancontrol
pwm
temp

1.5 Digital Inputs

- 16 digital inputs
- Logic level: 5V TTL.
- Each individual input can be declared as active low or active high
- Two input types available:
  - ON – OFF – regular digital input
  - Momentary Switch – this type of input is used for push buttons; the sensor assigned to a input of this type has it's value toggled at every active level of the input
- Any active input generates an internal event, and can control any of the 16 digital outputs
- Commands for inputs:

input
sensor

1.6 Digital Outputs

- 16 digital outputs
- Each individual output can be declared as active low or active high
- Manual setting and clearing of outputs
- Each output can be driven by more than one source. Potential output drivers are:
  - Internal Events : threshold violations for threshold sensors, active levels for discrete sensors
  - External Events: active inputs
- Each output can perform an **AND** or an **OR** function on the driving signals:
  - **AND** outputs are active if all the drivers are active
  - **OR** outputs are active when at least one of the drivers is active
- Commands for outputs:

  output
  sensor

1.7 Generating VME Signals

When used in a VME compatible system the Sysmon generates the ACFAIL* and SYSRESET* signals in accordance to VME specification ANSI/VITA 1-1994.

Any combination between the installed voltage sensors can be used to define the voltages that trigger ACFAIL* and SYSRESET*. The VME signals are triggered by a threshold infringement of any of the active voltages. The active threshold can be chosen between *Lower non-recoverable*, *Lower critical* and *Lower non-critical* thresholds, and has the same type for all the active voltages.

The active level for POWERFAIL is settable.

When SYSFAIL* is active, any combination of digital outputs can be asserted.

ACFAIL* and SYSRESET* signals can be picked up at the "VMEAS1" or "FCON" connectors. SYSFAIL* can be entered at the "VMEAS2" or "FCON" connectors and POWERFAIL can be entered at the "POWER" connector.

For correct behavior the Sysmon must be suitably set up via the ' vme ' command.

Timing diagram from VME specification ANSI/VITA 1-1994:

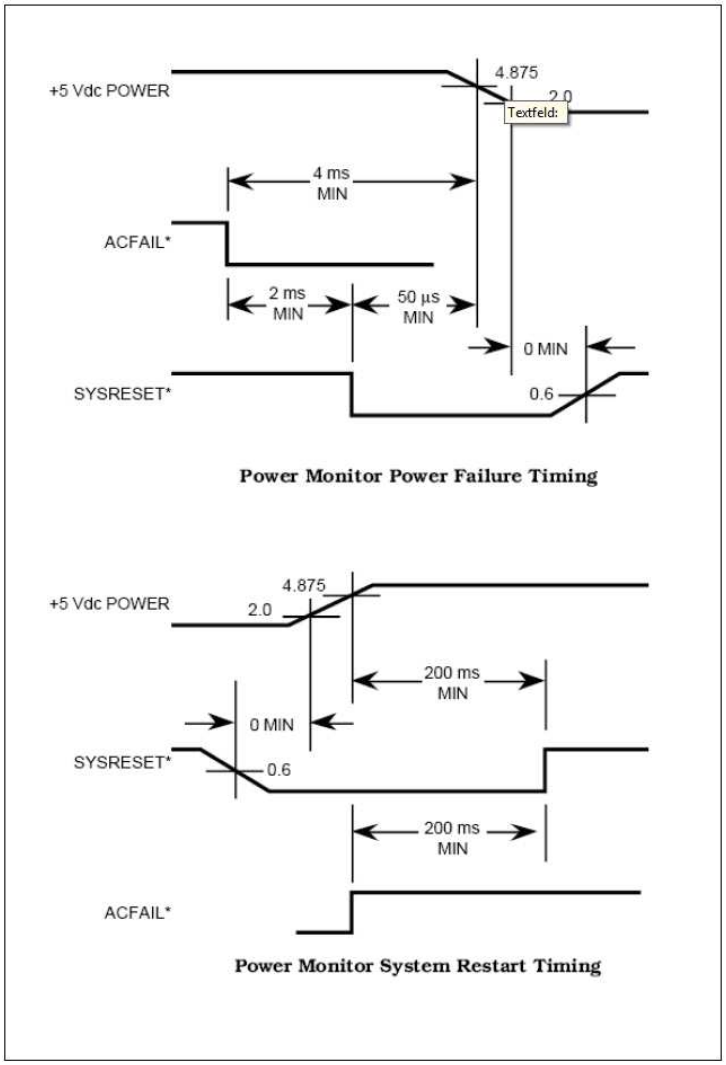

Figure 2: VME Timing Diagram

1.8 Led Display Support

The Sysmon provides support for ELMA's Led Display 025-384. It supports both single color(on-off) and multi-color LEDs.

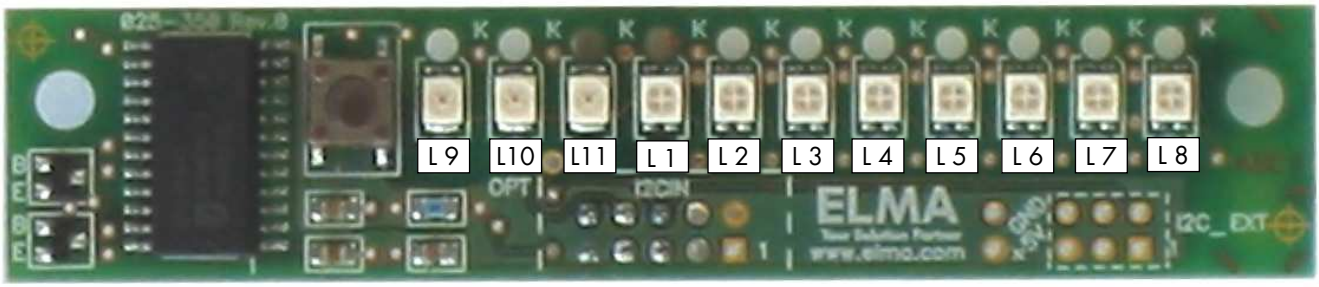

Figure 3: LED Display 025-384

The Led Display has 11 leds and 1 push button that can be used to generate a SYSRESET*.

The 11 leds are instantiated as 11 Compact SDRs. To use any of the leds, it's SDR has to be present in the SDR Records loaded in the NewSysmon.

Leds 1 to 8 are multi-color leds and can emit green,red or amber light,while leds 9-11 are single color leds and can only emit yellow light.

The NewSysmon allows it's users to assign a particular led to any of the available internal events: infringement for any of the 6 thresholds for threshold type sensors / active level for discrete sensors.

The way the Leds behave is defined in the next table.

Led Type	Driver State	
	Inactive	Active
Single color	Off	On
Multi-color	Green	Red Amber

Table 2: LED Behaviour

For single color LEDs, only LED On state can be triggered by NewSysmon's internal events.
For multi-color leds, the user can also choose which color (red,amber) will be emitted when the driver becomes active.

***Note:** This feature was designed to support signaling multiple thresholds infringements for the same sensor / group of sensors. In this case the led can be used as follows:

Green light - sensor is ok
Amber light – non-critical event
Red light – critical event

Keeping this application in mind, when a led will have both amber and red drivers active, the red ones will win the arbitration process.

The LEDs can have multiple **threshold** sensors assigned as drivers. The transfer function for the LED is OR on all its drivers.

****Note:** When considering multiple **discrete** drivers for a led, it is better to assign those drivers to an output and then assign that output to drive the led.
This type of assignment is recommended because events for discrete sensors are generated only on the signal's edges and leds track all edges without checking which signal they belong to.
It is better to use an output to track multiple discrete drivers because outputs perform logical AND or OR functions.

*****NOTE:** At power-up all leds that have SDRs loaded will be initialized. After that the **unused leds will be turned off**.

The sensor numbers for the Led Display Leds can be found in 2.2 Sensor Numbers .
The status of all the available leds can be checked using led command.
For more details about assigning a led to a particular internal event see 5.7.9. Changing the led assigned to a sensor event.

1.9 I2C Sensors Support

The Sysmon can monitor external I2C sensors that are connected to the bus available at the DIGITAL_TEMP connector.
Support has been implemented for 8 TMP75 I2C temperature sensors. All the aspects of the temperature monitoring algorithm are also in effect for I2C temperature sensors.
The sensor numbers for the I2C temperature sensors can be found in 2.2 Sensor Numbers .
For more details regarding the temperature measurements see 1.3 Temperature Monitoring.

1.10 I2C ADC Extension Boards

The monitoring capability of the Sysmon3Gen can be extended using the ADC extension boards. Each of these boards adds the possibility of monitoring of up to 4 additional Voltages and 4 additional currents. The boards support both positive and negative voltages and can monitor current up to 90 Amps.
For more information about available versions please contact us.

2 Sensor Data Records (SDRs)

The New Sysmon uses Sensor Data Records, compliant to IPMI 2.0, to describe the monitored System parameters.
Sensor Data Records are records that contain information about the type of sensors in the platform, sensor threshold support and event generation capabilities
By using SDRs to describe sensors the Sysmon becomes more flexible, easier to configure and makes it IPMI ready.

Using SDRs the user can:

- specify up to **6** thresholds
- decide the way the software treats a limit infringement(events and outputs can be enabled or disabled for each individual threshold)
- **set a name for every sensor**
- change the way the Sysmon operates by changing the sensor definitions
- enable/disable the monitoring of a sensor by loading/removing its SDR
- change monitoring parameters by CLI commands or by changing the SDRs

The Sysmon will only use 2 types of SDRs : Full Sensor Record (SDR type 01h) and Compact Sensor Record (SDR type 02h).
Full Sensor Records can describe any type of sensor. The *Compact* sensor record saves space, but has limitations in the sensors it can describe. The Full record is defined as a 64-byte record, while the Compact record is defined as 48-bytes.
The Sysmon will use Full sensor Records to describe threshold sensors: voltage, current, temperature, fan sensors; and Compact sensors Records to describe discrete sensors: rotor locked fan, input, output sensors.
The Sysmon will only monitor sensors for which the SDRs have been loaded.
For more information on SDRs you could refer to - *IPMI - Intelligent Platform Management Interface Specification Second Generation v2.0.*

2.1 Accessing Sensor Information

The Sysmon monitors many parameters of the system platform. Every monitored parameter has one, and only one, software sensor attached. So, each sensor corresponds to only one monitored parameter.
For accessing sensor information there are two options:

- displaying information for all the sensors of one type
- accessing each sensor individually using it's sensor's number

For accessing all the sensors of one type the user has to use the command associated to each sensor type: voltage, current, temp, fan, input, output.

The commands that displays information about all the sensors of one type have the same format:

```
%> xxxxxxxx
----------------------------Sensor List-----------------------------------
--no--Name------------Type--Value--Unit---State-------------------------
```

The command displays:

- **no** → sensor number; a unique number that identifies only one particular sensor.
- **Name** → sensor's name
- **Type** → sensor type : threshold or discrete
- **Value** → current value
- **Unit** → measuring unit
- **State** → current state for threshold sensors. Depending on the sensor's current value it's state could be:
  - Lower Non-Recoverable
  - Lower Critical
  - Lower Non-Critical
  - Upper Non-Critical
  - Upper Critical
  - Upped Non-Recoverable

The second option for accessing sensor information is using the sensor command. This command uses a sensor number to display all the available information for the sensor identified by that number. This command parses the SDR of the desired sensor and displays various information about the sensor:

- name
- type
- value
- threshold values
- sensor state
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- output function for output sensors
- active drivers for output sensors

2.2 Sensor Numbers

The correspondence between the sensor numbers and monitored parameters is defined in a table

presented bellow.

! Depending on the monitored system the Sysmon can be set up differently. Not always all the sensors described bellow are available. To check out all the installed sensors you can use the sensor command.

The Sysmon has 4 analog inputs that can be used to monitor either a voltage (V5_MON..V8_MON) or a current (current1 ..current4). For monitoring currents an auxiliary part is needed.
For each one of the 12 Fan input signals the Sysmon can monitor either a tachometer signal or a Rotor Locked active High signal. For Fans with Rotor locked signals that are active low any of the 16 digital inputs can be used.

! The Sysmon monitors only the parameters for which a SDR has been uploaded. The monitored parameters set can be changed by uploading a new SDR set. This is accomplished using xmodem *sdr*.

Sensor Number	Monitored System Parameter
1	Hot Swap Switch
2	3.3V line Voltage
3	5V line Voltage
4	12V line Voltage
5	-12V line Voltage
6	V5_MON – user inputed voltage
7	V6_MON – user inputed voltage
8	V7_MON – user inputed voltage
9	V8_MON – user inputed voltage
16	User inputed current 1
17	User inputed current 2
18	User inputed current 3
19	User inputed current 4
26	Temperature sensor 1
27	Temperature sensor 2
28	Temperature sensor 3
29	Temperature sensor 4
30	Temperature sensor 5
31	Temperature sensor 6
32	Local Temperature sensor
37	Tachometer signal for Fan1
38	Tachometer signal for Fan2
39	Tachometer signal for Fan3
40	Tachometer signal for Fan4

41	Tachometer signal for Fan5
42	Tachometer signal for Fan6
43	Tachometer signal for Fan7
44	Tachometer signal for Fan8
45	Tachometer signal for Fan9
46	Tachometer signal for Fan10
47	Tachometer signal for Fan11
48	Tachometer signal for Fan12
49	Rotor Locked Signal for Fan1
50	Rotor Locked Signal for Fan2
51	Rotor Locked Signal for Fan3
52	Rotor Locked Signal for Fan4
53	Rotor Locked Signal for Fan5
54	Rotor Locked Signal for Fan6
55	Rotor Locked Signal for Fan7
56	Rotor Locked Signal for Fan8
57	Rotor Locked Signal for Fan9
58	Rotor Locked Signal for Fan10
59	Rotor Locked Signal for Fan11
60	Rotor Locked Signal for Fan12
64	Digital Input 1
65	Digital Input 2
66	Digital Input 3
67	Digital Input 4
68	Digital Input 5
69	Digital Input 6
70	Digital Input 7
71	Digital Input 8
72	Digital Input 9
73	Digital Input 10
74	Digital Input 11
75	Digital Input 12
76	Digital Input 13

77	Digital Input 14
78	Digital Input 15
79	Digital Input 16
80	Digital Output 1
81	Digital Output 2
82	Digital Output 3
83	Digital Output 4
84	Digital Output 5
85	Digital Output 6
86	Digital Output 7
87	Digital Output 8
88	Digital Output 9
89	Digital Output 10
90	Digital Output 11
91	Digital Output 12
92	Digital Output 13
93	Digital Output 14
94	Digital Output 15
95	Digital Output 16
96	Sysfail
97	Sysmon Power On
98	Sysreset
101	Led Display: Led 1
102	Led Display: Led 2
103	Led Display: Led 3
104	Led Display: Led 4
105	Led Display: Led 5
106	Led Display: Led 6
107	Led Display: Led 7
108	Led Display: Led 8
109	Led Display: Led 9
110	Led Display: Led 10
111	Led Display: Led 11
117	TMP75 I2c Temp 1
118	TMP75 I2c Temp 2
119	TMP75 I2c Temp 3
120	TMP75 I2c Temp 4
121	TMP75 I2c Temp 5
122	TMP75 I2c Temp 6
123	TMP75 I2c Temp 7
124	TMP75 I2c Temp 8

128	I2c Voltage 1
129	I2c Voltage 2
130	I2c Voltage 3
131	I2c Voltage 4
132	I2c Voltage 5
133	I2c Voltage 6
134	I2c Voltage 7
144	I2c Current 1
145	I2c Current 2
146	I2c Current 3
147	I2c Current 4
148	I2c Current 5
149	I2c Current 6
150	I2c Current 7
152	046-322 Current1
153	046-322 Current2
154	046-322 Current3
155	046-322 Current4
156	046-322 Voltage1
157	046-322 Voltage2
158	046-322 Voltage3
159	046-322 Voltage4
160	046-323 Current1
161	046-323 Current2
162	046-323 Current3
163	046-323 Current4
164	046-323 Voltage1
165	046-323 Voltage2
166	046-323 Voltage3
167	046-323 Voltage4

Table 3: Sensor Numbers

2.3 Changing Sensor parameters

The Sysmon provides two possibilities for changing sensor parameters:

- using a general call command for changing one parameter for all the sensors of a particular type (temp, fan, input, output)
- using the sensor command to change only a particular sensor's parameter

The commands that access all the sensors of one type can be used to set the same value for a particular parameter for all the sensors of that type.

Not all the sensor types allow a general parameter change. The permitted general parameter changes are the following:

- temperature sensors : thresholds, hysteresis, fan control active mask
- fan speed sensors: thresholds,hysteresis
- input sensors: active level, input type
- output sensors: active level, output type

For more details and syntax refer to each command's section of this user manual.

If a change in parameters is desired for only a particular sensor the sensor command has to be used. Using this command the following parameters can be modified:

- thresholds
- hysteresis
- outputs assigned to a particular threshold event
- fan control mask for temperature sensors
- active level for input and output sensors
- output type
- input type

For more details and syntax refer to the *sensor* command section of this user manual.

2.4 Sensor Event Log (SEL)

The Sysmon uses a non volatile Flash memory to log the events generated by the sensor monitoring function. The Log can hold information for up to 65534 events.

! After the log reaches full capacity, a warning message will be displayed and all new events will be disregarded ! New Events will be logged only after the sel is cleared.

Logging is enabled for each event individually using the SDRs. The Sysmon uses the ***Assertion Events Mask*** and ***Deassertion Events Mask*** fields in each SDR to decide which events are logged.

! Only events that are supported by the sensor, and are accordingly marked by the SDR will be logged.

For each event the log provides the following information:

- time stamp: amount of time that has passed between the Sysmon's power up and the current event
- number and name of the sensor that generated the event
- event type: - for threshold sensors: UNR,UC,UNC,LNC,LC,LNR
  - for discrete sensors: 1 (Asserted), 0 (De-Asserted)
- sensor value that triggered the event and threshold value (only for threshold sensors)

Example 1: Sensor event log

```
 %>sel
------------------------------------------------------------------------------
                             Sensor  Event Log

Record ID.  Days: h: m: s  Sensor No. Name            Event  Ev.Dir  Value  Threshold
------------------------------------------------------------------------------
------------------------------------------------------------------------------
  0x0001     000:00:00:00     97      Sysmon Power ON   1 (Asserted)
  0x0002     000:00:00:14     2       +3.3V             LNC    As     2.82     2.86
  0x0003     000:00:00:14     3       +5V               LC     As     0.16     4.76
  0x0004     000:00:00:14     4       +12V              LC     As     10.53    11.38
  0x0005     000:00:00:14     5       -12V              LC     As     -12.64   -12.64
```

Each Sysmon restart is marked by an event 1 (Asserted) for the *"Sysmon Power On Sensor"*. All events are timestamped in relation to the last known power up.

For more details and syntax refer to the *sel* command section of this user manual.

3 Ethernet interface

The integrated 10/100Mbps Ethernet interface allows the Sysmon to be linked to any existing network. The interface supports DHCP, SNMP, TFTP, HTTP and TELNET protocols via TCP/IP and UDP.

All monitored system parameters can be displayed via a standard browser (HTTP protocol). The Command Line Interface (CLI) is accessible via TELNET, allowing remote control of the Sysmon. The use of standard protocols avoids the need for special software or drivers and so achieves platform-independence. The TCP/IP protocol supports up to 10 simultaneous connections.

! The factory default setting for the Sysmon is DHCP enabled so it negotiates automatically all the necessary addresses. If another IP address is desired, DHCP must be disabled and the IP address has to be set manually. For all these operations all you need is the lanconfig command.

Default Ethernet Addresses:

IP: 172.21.35.102
Network Mask: 255.255.255.0
Gateway: 172.21.35.255

Terminal settings:

- Local echo: *off*
- Local line editing: *off*
- Backspace key: *Control-H*

3.1 WEB

The Sysmon also includes a default web page presenting the system status. The web page is a SHTML page (a HTML page with server side includes (SSI)). Only the #printenv directive will be supported from the SSI directives. The web page will be accessed at addresses: *<IP_ADDRESS>/*

The default web page could be changed anytime using xmodem file transfer on the serial interface. The maximum page size is 128Kbytes.

Implementation:

The web page includes in the HTML code strings like:

<!--#printenv Environment_Variable -->

A HTML browser will interpret this string as a comment but the web server running on the system monitor will replace it with the appropriate parameter that is identified by the *Environment_Variable (Table 4)*.

Environment Variables

Name	Description
MAC_ADDRESS	MAC address of the Sysmon 00:14:DB:xx:xx:xx
IP_ADDRESS	The system monitor IP address
PART_NUMBER	Sysmon part number
SERIAL_NUMBER	Sysmon serial number
FIRMWARE_VERSION	Sysmon firmware version
TITLE	A 30 characters string that will be set as the web pages title
NAME *x*	The name of the sensor identified by the sensor number *x*
VALUE *x*	The value of the sensor identified by the sensor number *x*
COLOR *x*	Depending on the state of the sensor identified by the sensor number *x* the color is: • **black/grey** for Output sensors, depending if the user is allowed to change or not the output, and white for all other discrete sensors • **green** for threshold sensor values that are Ok • **yellow** for threshold sensor values that infringe non-critical thresholds (*Lower Non-Critical* or *Upper Non-Critical*) • **red** for threshold sensor values that infringe critical or non-recoverable thresholds (Lower Critical, Lower Non-Recoverable, Upper Critical or Upper Non-Recoverable)

Table 4: Web Environment Variables

For a list of the default sensor numbers refer to paragraph 2.2. Sensor Numbers .

Buttons on the Web page:

It is possible to use buttons in the web page in order to turn on/off some outputs or generate a SYSRESET pulse. The buttons will be usable only if the user have the permission to turn on off outputs.

The HTML text bellow is an example for a sysreset button that is black if the user is allowed to generate the sysreset.

```
<input style="font-family:verdana; color:<!–#printenv COLOR 98–> ; font-size:12" type="submit" name=98 value="Sysreset">
```

The name of button must be the sensor number, in this example name=98

Webpage refresh:

Using the meta refresh HTML directive, it is possible to force the web browser to refresh the web page. If this option is used please redirect all the time the webpage to the local IP address as in the example bellow:

```
<head>
```

<meta http-equiv="refresh" content="10; url=http://<!–#printenv IP_ADDRESS–>">

</head>

3.2 Telnet

The Telnet interface supports two operating modes: - Command Line Interface (CLI) and Serial Over Lan (SOL).

The CLI is accessible if the operator is logged on as “user” or “admin” profile.

If the operator is logging into the sysmon as “serial” profile the SOL mode is activated. Whenever the sysmon enters in SOL mode the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on “serial” profile. The data received from the telnet connections opened in “serial” mode is transferred to the serial port. This feature allows remote control of a CPU card inside the system. For this the CPU terminal screen should be redirected to the serial port and the serial port of the CPU needs to be connected with the serial port of the sysmon.

Default access settings:

login: serial
password: SERIAL

In SOL mode the SysRq key could be send by sending a Telnet break.

Note: If SOL mode is used it is recommended to make the SOL mode as default using “sol” command.

3.3 SNMP

The sysmon supports SNMP v2c and v1. The community names are the passwords defined for the user and admin profiles. The user profiles have read-only access to all parameters while the admin profile could also set the variables that are writable.

The sysmon does not send SNMP traps.

The SNMP variable tree is build under the node:
.iso.org.dod.internet.private.enterprises.elma.products.sysmon (.1.3.6.1.4.1.37968.1.1)

The SNMP data is structured in some categories:

.system

- contains information about the system monitor:name, part number, serial number, MAC address and software version

.temp

- contains the number of temperature sensors and for each sensor, the name, value, state, thresholds and hysteresis.

.voltage

- contains the number of voltage sensors and for each sensor, the name, value, state, thresholds and hysteresis

.current

– contains the number of current sensors and for each sensor, the name, value, state, thresholds and hysteresis

.fan

– contains the number of fan sensors, the speed level of each group of fans and for each sensor, the name, value, state, thresholds and hysteresis

.input

– contains the number of inputs and for each input, the name and state

.output

– contains the number of outputs and for each output, the name and state

.control

– this category allows different configurations to be done to the system monitor. A SYSRESET could be generated, the RS232 speed and the user and admin passwords could be read or changed. The configuration settings could be saved and the system monitor could be restarted.

4 RS232 serial interface

The Sysmon provides an RS232 serial interface through which the commands of the Command Line Interface (CLI) can be sent.
On Windows systems, we recommend the use of "Tera Term" or "Hyperterminal" as the terminal programs.

Terminal settings:

- 19200 bits per second
- data bits: 8
- parity: none
- stop bit: 1

The baud rate of the RS232 Serial Interface can be changed. The available baud rates are 9600, 19200, 38400. To change the speed of the Serial Interface use the scispeed command

In addition, the xmodem command can be used via the serial interface for file transfer.

! Use a 1:1 serial cable for direct connection to the serial port of a PC.
When using *xmodem* in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.

5 Command Line Interface (CLI)

The Command Line Interface (short-form: CLI) is available via both Telnet (3 Ethernet interface) and the RS232 serial interface (4 RS232 serial interface).

The user can read or newly configure and save system parameters via the CLI.

Access is divided into 2 profiles and is password-protected.

"user" profile:
System parameters can only be read in this profile – the exception to this write-protect is the lanconfig command for setting the IP, subnet and gateway addresses.

"admin" profile:
Full access to all system parameters is granted. All available CLI commands can be executed. To avoid possible damage or malfunctions, the access data for this profile must only be available to trained personnel with appropriate knowledge and competence relating to the system in which the Sysmon is used!
The profiles can be changed using the logout command.

The measured values are available at any time via the RS232 serial interface and via Telnet. In addition, limits and system parameters can be changed at any time with the unit in service. As soon as you have established a connection, you will be prompted to log in.

Default access settings:

login: user
password: USER

login: admin
password: ADMIN

! The passwords can be changed using the passw command.

General syntax conventions

Command [parameter1 | parameter2 | parameter3 [*value*]]

[ ] = optional
A command on its own with no entry of other parameters returns all available current values associated with the command.
The parameters separated by "|" rule out each other.
If the command line contains a *value*, this is assigned to the corresponding parameter and saved temporarily in the RAM. The change is active immediately. If the new value is desired to be valid after the reboot, the environment variables must be saved with the saveenv command. Changes not confirmed with saveenv are lost after a reboot.

! Sometimes the value entered for a parameter may not be exactly the typed number because the value has to be converted to a 10 bit ADC scale.

5.1 voltage command

Syntax: voltage

Functions:
Displays information about all the installed voltage sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1: voltage sensor values read out

```
%>voltage
---------------------------Sensor List--------------------------
--no--Name------------Type--Value--Unit---State-----------------

*  2  +3.3V            Thr   2.73   V      Lower Non-Recoverable
*  3  +5V              Thr   5.02   V      Ok
*  4  +12V             Thr   13.40  V      Upper Critical
*  5  -12V             Thr   -11.22 V      Ok
```

In this example the Sysmon monitors 4 voltage sensors.
Sensor number 2 is a threshold sensor monitoring the 3.3V voltage. Its current value is 2.73, it's measuring unit is Volts and it's state is *Lower Non-Recoverable* because its current value is now lower than the Lower Non Recoverable Threshold.

5.2 current command

Syntax: current

Functions:
Displays information about all the installed current sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1: current sensor values read out

```
%>current
---------------------------Sensor List--------------------------
--no--Name------------Type--Value--Unit---State-----------------
* 16  +3.3V Current    Thr   3.00   A      Ok
* 17  +5V Current      Thr   3.30   A      Ok
* 18  +12V Current     Thr   10.63  A      Upper Critical
* 19  -12V Current     Thr   1.50   A      Ok
```

In this example the Sysmon monitors 4 current sensors.
Sensor number 18 is a threshold sensor monitoring the +12V current. Its current value is 10.63, it's

measuring unit is Amps and it's state is *Upper Critical* because its current value is now grater than the Upper Critical Threshold.

5.3 fan command

Syntax: fan [threshold threshold_code *value* | *disable*] **| [hysteresis** hysteresis_code *value*]

- threshold code defined in Table 5: Threshold_Code
- value: RPM value
- hysteresis code defined in this Table 6: Hysteresis_Code

Threshold	threshold_code
Lower Non-Recoverable	lnr
Lower Critical	lc
Lower Non-Critical	lnc
Upper Non-Critical	unc
Upper Critical	uc
Upper Non-Recoverable	unr

Table 5: Threshold Code

Hysteresis	*hysteresis_code*
Negative going Hysteresis Value	neg
Positive going Hysteresis Value	pos

Table 6: Hysteresis Code

Functions:
When used without any parameter the command displays information about all the installed fan speed sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1: fan sensor values read out

```
%>fan
----------------------------Sensor List---------------------------
--no--Name------------Type--Value--Unit---State-----------------
* 37  Fan1            Thr   8800   RPM    Ok
* 38  Fan2            Thr   2700   RPM    Ok
* 39  Fan3            Thr   2700   RPM    Ok
* 40  Fan4            Thr   2600   RPM    Ok
* 41  Fan5            Thr   4800   RPM    Ok
* 42  Fan6            Thr   8800   RPM    Ok
```

In this example the Sysmon monitors 6 Fan sensors.
Sensor number 39 is a threshold sensor monitoring the Fan3 's speed . Its current value is 2700, its measuring unit is RPM and it's state is *Ok*.

The command can also set the same value for thresholds or hysteresis, to all fan speed sensors. It can also be used for disabling a particular threshold for all fan sensors.
For threshold and hysteresis codes and more info on how to change the respective parameter refer to paragraphs 5.7.1 Changing a threshold and 5.7.2 Changing the hysteresis .

Example 2: Lower critical threshold value change, disable Lower Non_critical threshold for all fans

```
%>fan threshold lc 800
--no--Sensor-------------Status------------------
* 37  Fan1  :  Operation Successful!
* 38  Fan2  :  Operation Successful!
* 39  Fan3  :  Operation Successful!
* 40  Fan4  :  Operation Successful!

%>fan threshold lnc disable
--no--Sensor-------------Status------------------
* 37  Fan1  :  Threshold disabled!
* 38  Fan2  :  Threshold disabled!
* 39  Fan3  :  Threshold disabled!
* 40  Fan4  :  Threshold disabled!
```

In this example the Sysmon monitors 4 Fan sensors. The threshold value change was successful for all of them. This is not always the case. To see all the possible results to a threshold value change refer to the sensor command's section of this users manual.

Example 3: Positive going hysteresis value change

```
%>fan hysteresis pos 300
--no--Sensor-------------Status------------------
* 46  Fan10 :  Operation Successful!
* 48  Fan12 :  Operation Successful!
```

In this example the Sysmon monitors 2 Fan sensors. The hysteresis value change was successful for both. This is not always the case. To see all the possible results to a hysteresis value change refer to the sensor command's section of this users manual.

5.4 temp command

Syntax: temp [threshold threshold_code *value* | *disable* **] | [hysteresis** hysteresis_code *value***] | [fancontrol** fancontrol_mask**]**

- threshold code defined in Table 5: Threshold_Code
- value: Celsius degrees
- hysteresis code defined in Table 6: Hysteresis_Code
- fancontrol_mask defined in Table 7: Fancontrol_Mask

Functions:
Displays information about all the installed temperature sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type

- current value
- measuring unit
- current state

fancontrol_mask bit	Bit value	Details
7 ... 4	0	Reserved
3	Sensor active state for fancontrol 4	1: sensor active 0: sensor inactive
2	Sensor active state for fancontrol 3	1: sensor active 0: sensor inactive
1	Sensor active state for fancontrol 2	1: sensor active 0: sensor inactive
0	Sensor active state for fancontrol 1	1: sensor active 0: sensor inactive

Table 7: Fancontrol Mask

Example 1: temperature sensor values read out

```
%>temp
----------------------------Sensor List-----------------------------------
--no--Name------------Type--Value--Unit---State-------------------------
* 26  Temp1           Thr   26.00  deg C  Ok
* 27  Temp2           Thr   26.00  deg C  Ok
* 28  Temp3           Thr   26.00  deg C  Ok
* 29  Temp4           Thr   Not Present!
* 30  Temp5           Thr   Not Present!
* 31  Temp6           Thr   Not Present!
```

In this example the Sysmon monitors 6 temperature sensors, but 3 of them are not present.
Sensor number 26 is a threshold sensor monitoring the the temperature of temp sensor number 1. It's current value is 26, it's measuring unit is degrees Celsius and it's state is *Ok*.

The command can also set the same value for thresholds or hysteresis, for all temperature sensors. It can also be used for disabling a particular threshold for all fan sensors.
For threshold and hysteresis codes and more info on changing the respective parameter refer to paragraphs 5.7.1 Changing a threshold and 5.7.2 Changing the hysteresis .

Example 2: Upper critical threshold value change,disable Upper Non-Recoverable threshold for all temperature sensors

```
%>temp threshold uc 45
--no--Sensor------------Status-----------------
* 26  Temp1 :  Operation Successful!
* 27  Temp2 :  Operation Successful!
* 28  Temp3 :  Operation Successful!
* 29  Temp4 :  Operation Successful!
* 30  Temp5 :  Operation Successful!
* 31  Temp6 :  Operation Successful!

%>temp threshold unr disable
--no--Sensor------------Status-----------------
* 26  Temp1 :  Threshold disabled!
* 27  Temp1 :  Threshold disabled!
* 28  Temp3 :  Threshold disabled!
* 29  Temp4 :  Threshold disabled!
* 30  Temp5 :  Threshold disabled!
* 31  Temp6 :  Threshold disabled!
```

In this example the Sysmon monitors 6 temp sensors. The threshold value change was successful for all of them. This is not always the case. To see all the possible results to a threshold value change refer to

the sensor command's section of this user manual.

Example 3: Negative going hysteresis value change

```
%>temp hysteresis neg 2
--no--Sensor-------------Status------------------
* 27  Temp2 :  Operation Successful!
* 28  Temp3 :  Operation Successful!
* 31  Temp6 :  Operation Successful!
```

In this example the Sysmon monitors 3 temperature sensors. The hysteresis value change was successful for all of them. This is not always the case. To see all the possible results to a hysteresis value change refer to the sensor command's section of this user manual.

Each temperature sensor can be part of the fan control algorithm. For controlling the fan speeds the Sysmon has 4 independent fan control groups, each with it's own designated PWM signal and temperature-speed characteristic. Basically, the speed of the fans is proportional with the temperature. The temperature that is used in the speed processing algorithm is the maximum value of the temperature sensors that are active for each fan control group. For more details regarding the Fan control algorithm refer to 1.4.1 Fan Control of this user manual.
The values of the least significant 4 bits in the fancontrol_mask decide the fan control groups the temperature sensor is active for. The fancontrol_mask has 8 bits, but only the least significant 4 are used: The fancontrol_mask is entered as a hex value. A sensor that is active for all 4 fancontrol groups has a fancontrol_mask of 0x0F.

Example 4: Fan Control mask change. Set all temperature sensors active for Fan Control Groups 1 and 2

```
%>temp fancontrol 0x03
--no--Sensor-------------Status------------------
* 27  Temp2 :  Operation Successful!
* 28  Temp3 :  Operation Successful!
```

In this example the Sysmon monitors 2 temperature sensors. The fancontrol mask change was successful for both.

5.5 input command

The Sysmon supports 16 digital inputs. For all of them a discrete sensor exists.

Syntax: input [activelevel value**]** | **[input_type** mswitch | on_off**]**

- value 0 | 1

Functions:
Displays information about all the input sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value

Example 1: Input sensors values read out

```
%>input
---------------------------Sensor List-----------------------------------
--no--Name------------Type--Value--Unit---State--------------------------
* 64  Input1           Disc  1
* 65  Input2           Disc  1
* 66  Input3           Disc  0
```

```
* 67  Input4           Disc  0
* 68  Input5           Disc  0
* 69  Input6           Disc  1
* 70  Input7           Disc  1
* 71  Input8           Disc  1
* 72  Input9           Disc  1
* 73  Input10          Disc  1
* 74  Input11          Disc  0
* 75  Input12          Disc  0
* 76  Input13          Disc  1
* 77  Input14          Disc  0
* 78  Input15          Disc  1
* 79  Input16          Disc  1
```

The command can also change the active level for all inputs. Accepted values are "0" and "1". If any other value is entered it is considered to be "1".

Example 2: Change active level of inputs

```
%>input activelevel 0
--no--Sensor------------Status-----------------
* 64  Input1  :  Operation Successful!
* 65  Input2  :  Operation Successful!
* 66  Input3  :  Operation Successful!
* 67  Input4  :  Operation Successful!
* 68  Input5  :  Operation Successful!
* 69  Input6  :  Operation Successful!
* 70  Input7  :  Operation Successful!
* 71  Input8  :  Operation Successful!
* 72  Input9  :  Operation Successful!
* 73  Input10 :  Operation Successful!
* 74  Input11 :  Operation Successful!
* 74  Input12 :  Operation Successful!
* 76  Input13 :  Operation Successful!
* 77  Input14 :  Operation Successful!
* 78  Input15 :  Operation Successful!
* 79  Input16 :  Operation Successful!
```

In this example the active level for all the 16 inputs is changed to 0 (low).

The Sysmon supports two input types for the digital inputs:

- ON – OFF – regular digital input
- Momentary Switch – this type of input is used for push buttons; the sensor assigned to a input of this type has it's value toggled at every active level of the input

Example 3: Setting all inputs as momentary switches

```
%>input input_type mswitch
--no--Sensor------------Status-----------------
* 64  Input1  :  Operation Successful!
* 65  Input2  :  Operation Successful!
* 66  Input3  :  Operation Successful!
* 67  Input4  :  Operation Successful!
* 68  Input5  :  Operation Successful!
* 69  Input6  :  Operation Successful!
* 70  Input7  :  Operation Successful!
* 71  Input8  :  Operation Successful!
* 72  Input9  :  Operation Successful!
* 73  Input10 :  Operation Successful!
* 74  Input11 :  Operation Successful!
* 75  Input12 :  Operation Successful!
* 76  Input13 :  Operation Successful!
* 77  Input14 :  Operation Successful!
* 78  Input15 :  Operation Successful!
* 79  Input16 :  Operation Successful!
```

5.6 output command

The Sysmon supports 16 digital outputs. For all of them a discrete sensor is assigned.

Syntax: output [activelevel value**] | [output_type** AND | OR**] | [assert** *hex_outputs***] | [deassert** *hex_outputs***]**

- value 0 | 1
- hex_outputs defined in Table 8: Hex_Outputs

Functions:

Displays information about all the output sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value

hex_outputs bit	Value
15	Output 16
14	Output 15
13	Output 14
12	Output 13
11	Output 12
10	Output 11
9	Output 10
8	Output 9
7	Output 8
6	Output 7
5	Output 6
4	Output 5
3	Output 4
2	Output 3
1	Output 2
0	Output 1

Table 8: Hex Outputs Bit Mask

Example 1: Output sensors values read out

```
%>output
---------------------------Sensor List-----------------------------------
--no--Name------------Type--Value--Unit---State--------------------------
* 80  Output1          Disc  0
* 81  Output2          Disc  1
* 82  Output3          Disc  0
* 83  Output4          Disc  0
* 84  Output5          Disc  0
* 85  Output6          Disc  0
* 86  Output7          Disc  1
* 87  Output8          Disc  1
* 88  Output9          Disc  1
* 89  Output10         Disc  0
```

```
* 90  Output11         Disc  0
* 91  Output12         Disc  0
* 92  Output13         Disc  0
* 93  Output14         Disc  0
* 94  Output15         Disc  1
* 95  Output16         Disc  1
```

The command can change the active level for all outputs. Accepted values are "0" and "1". If any other value is entered it is considered to be "1".

Example 2: Change active level of outputs

```
%>output activelevel 1
--no--Sensor-------------Status-----------------
* 80  Output1  :  Operation Successful!
* 81  Output2  :  Operation Successful!
* 82  Output3  :  Operation Successful!
* 83  Output4  :  Operation Successful!
* 84  Output5  :  Operation Successful!
* 85  Output6  :  Operation Successful!
* 86  Output7  :  Operation Successful!
* 87  Output8  :  Operation Successful!
* 88  Output9  :  Operation Successful!
* 89  Output10 :  Operation Successful!
* 90  Output11 :  Operation Successful!
* 91  Output12 :  Operation Successful!
* 92  Output13 :  Operation Successful!
* 93  Output14 :  Operation Successful!
* 94  Output15 :  Operation Successful!
* 95  Output16 :  Operation Successful!
```

In this example the activelevel for all the 16 outputs is changed to 1(high).

The output command can also change the output function for all output sensors. The outputs can perform an *AND* or an *OR* operator on their drivers.
An *AND* output is asserted when all it's driving signals are asserted.
An *OR* output is asserted when at least one of it's driving signals is asserted.

Example 3: Change output function for all outputs

```
%>output output_type AND
--no--Sensor-------------Status-----------------
* 80  Output1  :  Operation Successful!
* 81  Output2  :  Operation Successful!
* 82  Output3  :  Operation Successful!
* 83  Output4  :  Operation Successful!
* 84  Output5  :  Operation Successful!
* 85  Output6  :  Operation Successful!
* 86  Output7  :  Operation Successful!
* 87  Output8  :  Operation Successful!
* 88  Output9  :  Operation Successful!
* 89  Output10 :  Operation Successful!
* 90  Output11 :  Operation Successful!
* 91  Output12 :  Operation Successful!
* 92  Output13 :  Operation Successful!
* 93  Output14 :  Operation Successful!
* 94  Output15 :  Operation Successful!
* 95  Output16 :  Operation Successful!
```

In this example the output function for all the 16 outputs is changed to *AND*.

The output command can be used to manually assert or deassert the digital outputs.
If a bit of *hex_outputs* is 1 the corresponding output is affected by the assert/deassert command. For example if outputs 1, 5 and 8 need to be asserted *output assert 0x0091* is entered.

Example 4: Assert outputs

```
%>output assert 0x0f03
 Done!
```

In this example outputs 1, 2, 9, 10, 11, 12 are asserted.

5.7 sensor command

Syntax: sensor [*sensor_no* **[threshold** threshold_code *value***] |**
[hysteresis hysteresis_code *value***] |**
[output output_code hex_outputs**] |**
[activelevel value**] |**
[fancontrol fancontrol_mask**] |**
[output_type OR | AND**] |**
[input_type mswitch | on_off**] |**
[assert | deassert] |
[led [threshold_code] led_sensor_no **[color]]|**
[led en| di | (time *time_value*)**] |**
[debounce debounce_value**] |**
[delay as|de time_value**]**

- sensor_no : the sensor number that identifies a particular sensor(2.2 . Sensor Numbers)
- threshold_code is defined in Table 5: Threshold_Code
- hex_outputs defined in Table 8: Hex_Outputs
- value: Volts for voltage sensor, Amps for current sensor, Celsius degrees for temperature sensors, RPM for fan sensors
- hysteresis code is defined in Table 6: Hysteresis_Code
- activelevel value : 0 or 1
- fancontrol_mask defined in Table 7: Fancontrol_Mask
- led_sensor_no : the sensor number that identifies the led that will be assigned to the current sensor(2.2 . Sensor Numbers)
- color : “amber” | “red”; parameter will be omitted for single color leds.
- time_value : 0…128
- debounce_value : 0 … 255
- time_value : 0 … 255 seconds

Functions:

The sensor command can be used either to access information about the sensors, or to change different sensor parameters.

If the command is used without any parameter, it will return basic information about all the active

sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit (only for threshold sensors)
- current state (only for threshold sensors)

Example 1: Read all the active sensors

```
%>sensor
----------------------------Sensor List---------------------------
--no--Name------------Type--Value--Unit---State-----------------
*  2  +3.3V           Thr   3.27   V      Ok
*  3  +5V             Thr   5.04   V      Ok
*  4  +12V            Thr   0.00   V      Lower Critical
*  5  -12V            Thr   -11.88 V      Ok
* 16  +3.3V Current    Thr   0.50   A      Ok
* 17  +5V Current      Thr   0.00   A      Lower Critical
* 18  +12V Current     Thr   0.30   A      Ok
* 19  -12V Current     Thr   0.30   A      Ok
* 26  Temp1            Thr   27.00  deg C  Ok
* 27  Temp2            Thr   27.00  deg C  Ok
* 45  Fan9             Thr   2100   RPM    Ok
* 46  Fan10            Thr   2100   RPM    Ok
* 47  Fan11            Thr   2100   RPM    Ok
* 48  Fan12            Thr   2100   RPM    Ok
* 64  Input1           Disc  1
* 65  Input2           Disc  1
* 66  Input3           Disc  1
* 67  Input4           Disc  1
* 68  Input5           Disc  0
* 69  Input6           Disc  0
* 70  Input7           Disc  1
* 71  Input8           Disc  1
* 72  Input9           Disc  1
* 73  Input10          Disc  1
* 74  Input11          Disc  0
* 75  Input12          Disc  1
* 76  Input13          Disc  1
* 77  Input14          Disc  0
* 78  Input15          Disc  1
* 79  Input16          Disc  0
* 80  Output1          Disc  0
* 81  Output2          Disc  0
* 82  Output3          Disc  1
* 83  Output4          Disc  0
* 84  Output5          Disc  1
* 85  Output6          Disc  0
* 86  Output7          Disc  1
* 87  Output8          Disc  1
* 88  Output9          Disc  0
* 89  Output10         Disc  0
* 90  Output11         Disc  0
* 91  Output12         Disc  1
* 92  Output13         Disc  1
* 93  Output14         Disc  0
* 94  Output15         Disc  0
* 95  Output16         Disc  0
```

In this example the Sysmon monitors: 4 voltage , 4 current , 2 temperature , 4 fan, 16 input and 16 output sensors.

If only the sensor number parameter is entered, the command will print a detailed description of the sensor identified by that particular sensor number. The command parses the SDR of the desired sensor and displays various information about the sensor:

- name
- type
- value
- sensor units
- sensor state
- sensor maximum and minimum values
- threshold values
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- input type for input sensors
- output type for output sensors
- active drivers for output sensors

Example 2: Get detailed information about the 5V sensor

```
%>sensor 3
-----------------Sensor Details------------------
* Name: +5V
* Type: Threshold
* Value: 0.00
* Sensor Units: V
* State: Lower Critical going Low
* Sensor Maximum Reading: 7.98
* Sensor Minimum Reading: 0.00
* Upper critical threshold: 5.26
  Assigned Outputs:
    Sensor 82 Output3
* Lower critical threshold: 4.76
  Assigned Outputs:
    Sensor 84 Output5
    Sensor 92 Output13
* Positive-going threshold hysteresis value: 0.03
* Negative-going threshold hysteresis value: 0.03
```

In this example the details of sensor 3 are displayed:

- Sensor 3 is a threshold sensor
- name : "+5V"
- has 2 thresholds defined : upper critical(5.26 V) and lower critical(4.76 V)
- supports hysteresis
- has outputs 5 and 13 assigned to a lower critical threshold infringement, and output 3 assigned to a upper critical threshold infringement

The sensor command can also be used to change different sensor parameters. This command will change the desired parameter for only the sensor identified by the entered sensor number.
If a parameter needs to be modified for all the sensors of one type a general call command could be used.(temp, fan , input , output)
Depending on the sensors type there are different parameters that can be changed using general call commands:

- temperature sensors : thresholds, hysteresis, fan control active mask
- fan speed sensors: thresholds,hysteresis
- input sensors: active level, input type
- output sensors: active level, output type
- voltage: none
- current: none

Using the *sensor* command the following parameters can be changed:

- temperature sensors : thresholds, hysteresis, fan control active mask, outputs assigned to threshold events
- fan speed sensors: thresholds, hysteresis, outputs assigned to threshold events
- input sensors: active level, outputs assigned to active level,input type
- output sensors: active level, output drivers logic function, input type
- voltage: thresholds, hysteresis, outputs assigned to threshold events
- current: thresholds , hysteresis, output assigned to threshold events

The syntax for parameters change is the same in both cases. The only difference is the suffix entered before the parameter that needs to be changed.

- for general call the suffix is the command that defines the sensor type for which a general parameter needs to be changed
  suffix = ***temp*** | ***fan*** | ***input*** | ***output***
- *when the change affects only one sensor the suffix is the sensor command and the sensor number that identifies the sensor whose parameter needs to be changed*
  suffix = ***sensor sensor_no***

5.7.1 Changing a threshold's value | Disabling a Threshold

Syntax: *suffix* **threshold** threshold_code *value*| *disable*

- threshold_code is defined in Table 5: Threshold_Code

Thresholds can be modified only for threshold sensors. Because only temperature and fan sensors permit a general threshold change the ***suffix*** can be either *temp*, or *fan*(when changing a threshold for all the sensors of one type) or *sensor sensor_no* (when changing the threshold of the sensor identified by *sensor_no*).

Whatever the case of *suffix*, the syntax that follows is the same.

The Sysmon uses SDRs compliant to IPMI V2.0. For sensors that monitor voltages, currents, temperatures or fan speeds, the Sysmon uses Full Sensor Records.
Sensors described by full sensor records support 6 thresholds, but in some cases not all of them are active.
To activate an inactive threshold all you have to do is set an appropriate value for it.

The *threshold_code* parameter represents an abbreviation of the threshold that needs to be changed.

The value change operation will be successful only if the new value for the threshold is compliant to the monotony rule for thresholds:

lnr < lc < lnc < unc < uc < unr

The new value is compared only with the values of active thresholds.

For disabling a threshold the command has to be used with *disable* as parameter.

Example 3: Changing the upper critical threshold of temp 3 (sensor 28) and changing the upper critical threshold for all temperature sensors

```
%>sensor 28 threshold uc 45
 Operation Successful!


%>temp threshold uc 45
--no--Sensor-------------Status-----------------
* 26  Temp1 :  Operation Successful!
* 27  Temp2 :  Operation Successful!
* 28  Temp3 :  Operation Successful!
* 29  Temp4 :  Operation Successful!
* 30  Temp5 :  Operation Successful!
* 31  Temp6 :  Operation Successful!
```

Example 4: Disabling the upper critical threshold of temp 3 (sensor 28) and disabling the upper critical threshold for all fan sensors

```
%>sensor 28 threshold uc disable
 Threshold disabled!

%>fan threshold uc disable
--no--Sensor-------------Status-----------------
* 38  Fan2  :  Threshold disabled!
* 39  Fan3  :  Threshold disabled!
* 40  Fan4  :  Threshold disabled!
* 41  Fan5  :  Threshold disabled!
```

5.7.2 Changing the hysteresis

Syntax: *suffix* **hysteresis** hysteresis_code *value*

- hysteresis code is defined in Table 6: Hysteresis_Code

Hysteresis can be modified only for threshold sensors. Because only temperature and fan sensors permit a general hysteresis change the ***suffix*** can be either *temp or fan*(when changing a hysteresis value for all the sensors of one type) or *sensor sensor_no* (when changing a hysteresis value of the sensor identified by *sensor_no*).
Whatever the case of *suffix*, the syntax that follows is the same.
The Sysmon uses SDRs compliant to IPMI V2.0. For sensors that monitor voltages, currents, temperatures, fan speeds, the Sysmon uses Full Sensor Records.
Sensors described by full sensor records support 2 hysteresis values.
The *hysteresis_code* parameter represents a abbreviation of the hysteresis that needs to be changed.

! Hysteresis values can be changed only for sensors that support hysteresis.
In other cases the following warning message will be displayed:

```
Sensor does not support Hysteresis!
```

Example 5: Changing the positive going hysteresis of 3.3V (sensor 2) and changing the negative going hysteresis for all fan sensors

```
%>sensor 2 hysteresis pos 0.04
Operation Successful!
%>fan hysteresis neg 400
--no--Sensor-------------Status-----------------
* 37  Fan1  :  Operation Successful!
* 38  Fan2  :  Operation Successful!
```

```
* 39  Fan3  :  Operation Successful!
* 40  Fan4  :  Operation Successful!
* 41  Fan5  :  Operation Successful!
```

5.7.3 Changing the outputs linked to a sensor event

Syntax: **sensor sensor_no output** [*threshold_code*] *hex_outputs*

- *threshold_code is defined in* Table 5: Threshold_Code
- *hex_outputs is defined in* Table 8: Hex_Outputs

The outputs assigned to sensor events parameter can only be set individually for each sensor and each event.

Outputs can be linked to thresholds infringements for threshold sensors or to the active level for discrete sensors.
For threshold sensor the command uses the threshold_code parameter for identifying the event to which the outputs defined by *hex_outputs* are linked.
In the case of discrete sensor the outputs are automatically linked with the active level so the *threshold_code* parameter is not used.

If a bit of *hex_outputs* is 1 the corresponding output will be linked to the corresponding event. For example if outputs 1, 5 and 8 need to be linked with an event *hex_output* = 0x0091.

Example 6: Assigning outputs 1 and 2 to the lower critical threshold of fan 2(sensor 38)

```
%>sensor 38 output lc 0x0003
 Operation Successful!
```

Example 7: Assigning outputs 7 and 14 to the active level of input 6(sensor 69)

```
%>sensor 69 output 0x2040
 Operation Successful!
```

5.7.4 Changing the activelevel

Syntax: *suffix* **activelevel** *value*

Active level can be modified only for Discrete sensors, so the ***suffix*** can be either *input or output*(when changing a threshold for all the sensors of one type) or *sensor sensor_no* (where *sensor_no* identifies a discrete sensor).

Whatever the case of *suffix*, the syntax that follows is the same.

! activelevel can be changed only for discrete sensors.
value can be either 0 (low) or 1 (high). Any other number entered as value is considered 1 (high).

Example 8: Setting the activelevel to high for input 2 (sensor 65) and setting the activelevel to low for all the outputs

```
%>sensor 65 activelevel 1
 Operation Successful!

%>output activelevel 0
--no--Sensor------------Status-----------------
* 80  Output1  :  Operation Successful!
* 81  Output2  :  Operation Successful!
* 82  Output3  :  Operation Successful!
* 83  Output4  :  Operation Successful!
```

```
* 84  Output5  :  Operation Successful!
* 85  Output6  :  Operation Successful!
* 86  Output7  :  Operation Successful!
* 87  Output8  :  Operation Successful!
* 88  Output9  :  Operation Successful!
* 89  Output10 :  Operation Successful!
* 90  Output11 :  Operation Successful!
* 91  Output12 :  Operation Successful!
* 92  Output13 :  Operation Successful!
* 93  Output14 :  Operation Successful!
* 94  Output15 :  Operation Successful!
* 95  Output16 :  Operation Successful!
```

5.7.5 Changing the fancontrol mask

Syntax: *suffix* **fancontrol** *fancontrol_mask*

- *fancontrol_mask defined in* Table 7: Fancontrol_Mask

The fancontrol parameter is used only for temperature sensors and determines if the sensor is active for any of the fan control groups. For more details on fan control refer to 1.4.1 Fan control .
suffix may be *temp* (when setting the same value for fancontrol_mask to all temperature sensors) or *sensor sensor_no* (where sensor_no identifies a temperature sensor).

The fancontrol_mask has 8 bits, but only the least significant 4 are used. The fancontrol_mask is inputed as a hex value. A sensor that is active for all 4 fancontrol groups has a fancontrol_mask = 0x0F.

Example 9: Setting temp 3(sensor 28) active for fan control groups 1 and 4 and setting all fans active for fan control groups 1 and 3

```
%>sensor 28 fancontrol 0x09
 Operation Successful!

%>temp fancontrol 0x05
--no--Sensor-------------Status-----------------
* 26  Temp1 :  Operation Successful!
* 27  Temp2 :  Operation Successful!
* 28  Temp3 :  Operation Successful!
* 29  Temp4 :  Operation Successful!
* 30  Temp5 :  Operation Successful!
* 31  Temp6 :  Operation Successful!
```

5.7.6 Changing the output type

Syntax: *suffix* **output_type** *AND | OR*

The *output_type* parameter is available only for output sensors and determines the logic function the sensor performs on it's driving signals.
suffix may be *output* (for setting the same output type for all outputs) or *sensor sensor_no* (where *sensor_no* identifies an output sensor).
An AND output is asserted when all it's driving signals are asserted.
An OR output is asserted when at least one of it's drivers is asserted.

Example 10: Setting output 1's logic function(sensor 80) to *AND*, setting all the outputs to have an *OR* output function

```
%>sensor 80 output_type AND
 Operation Successful!

%>output output_type OR
--no--Sensor-------------Status-----------------
* 80  Output1  :  Operation Successful!
```

```
* 81  Output2  :  Operation Successful!
* 82  Output3  :  Operation Successful!
* 83  Output4  :  Operation Successful!
* 84  Output5  :  Operation Successful!
* 85  Output6  :  Operation Successful!
* 86  Output7  :  Operation Successful!
* 87  Output8  :  Operation Successful!
* 88  Output9  :  Operation Successful!
* 89  Output10 :  Operation Successful!
* 90  Output11 :  Operation Successful!
* 91  Output12 :  Operation Successful!
* 92  Output13 :  Operation Successful!
* 93  Output14 :  Operation Successful!
* 94  Output15 :  Operation Successful!
* 95  Output16 :  Operation Successful!
```

5.7.7 Changing the input type

Syntax: *suffix* **input_type** *mswitch | on_off*

The *input_type* parameter is available only for input sensors and determines the way the value of the sensor, assigned to the digital input, changes.

There are two input types:

- ON – OFF – regular digital input
- Momentary Switch – this type of input is used for push buttons; the sensor assigned to a input of this type has it's value toggled at every active level of the input

suffix may be *input* (for setting the same type for all inputs) or *sensor sensor_no* (when *sensor_no* identifies an input sensor).

Example 11: Setting input 1 as a momentary switch, setting the on-off input type for all inputs

```
%>sensor 64 input_type mswitch
 Operation Successful!

%>input input_type on_off
--no--Sensor------------Status-----------------
* 64  Input1  :  Operation Successful!
* 65  Input2  :  Operation Successful!
* 66  Input3  :  Operation Successful!
* 67  Input4  :  Operation Successful!
* 68  Input5  :  Operation Successful!
* 69  Input6  :  Operation Successful!
* 70  Input7  :  Operation Successful!
* 71  Input8  :  Operation Successful!
* 72  Input9  :  Operation Successful!
* 73  Input10 :  Operation Successful!
* 74  Input11 :  Operation Successful!
* 75  Input12 :  Operation Successful!
* 76  Input13 :  Operation Successful!
* 77  Input14 :  Operation Successful!
* 78  Input15 :  Operation Successful!
* 79  Input16 :  Operation Successful!
```

5.7.8 Asserting/Deasserting an output

Syntax: **output** assert|deassert *hex_outputs*

or

sensor sensor_no assert|deassert

- hex_outputs defined in Table 8: Hex_Outputs

Only output sensors can be activated or deactivated. There are two way of accomplishing such an operation:

- using the output command
- using the sensor command for an output sensor

Using the output command all the outputs identified by *hex_outputs* can be asserted/deasserted, while the sensor command can assert/deassert only the output identified by sensor_no.

hex_outputs is a 16 bits hex value describing the outputs that will be affected by the command that uses this parameter. For example if outputs 1, 5 and 8 need to be asserted/deasserted *hex_output* = 0x0091.

Example 12: Asserting Outputs 1,4,5, deasserting output 4

```
%>output assert 0x0019
 Done!

%>sensor 83 deassert
 Done!
```

5.7.9 Changing the led assigned to a sensor event

Syntax: **sensor** sensor_no *led* [threshold_code] (led_sensor_no [color] | disable)

- sensor_no : identifies the sensor that will be set up to drive the led.
- threshold_code is defined in Table 5: Threshold_Code
- led_sensor_no: identifies the led
- color: red | amber for multi-color leds. This parameter is omitted for single color leds.

The led driver was designed to support signaling multiple thresholds infringements for the same sensor / group of sensors. In that case the led can be used as follows:

Green light - sensor is ok
Amber light – non-critical event
Red light – critical event

Keeping this application in mind, when a led will have both amber and red drivers active, the red ones will win the arbitration process.

The leds can have multiple **threshold** sensors assigned as drivers. The transfer function for the led is OR on all it's drivers.

****Note:** When considering multiple **discrete** drivers for a led, it is better to assign those drivers to an output and then assign that output to drive the led.
This type of assignment is recommended because events for discrete sensors are generated only on the signal's edges and leds track all edges without checking which signal they belong to.
It is better to use an output to track multiple discrete drivers because outputs perform logical AND or OR functions.
For any event the led assignment can be removed by using the command with the parameter disable.

Example 13: Assigning a multi-color led (led 1) to a threshold sensor event

```
%>sensor 2 led uc 101 amber
 Operation Successful! Save environment and reboot!
```

Example 14: Assigning a single color led (led 9) to a threshold sensor event

```
%>sensor 2 led lc 109
 Operation Successful! Save environment and reboot!
```

Example 15: Assigning a single color led (led 9) to a discrete sensor event

```
%>sensor 64 led 109
 Operation Successful! Save environment and reboot!
```

Example 16: Removing the led assignment from a sensor event

```
%>sensor 2 led uc disable
 Operation Successful! Save environment and reboot!
```

5.7.10 Changing the LCD display option

Syntax: **sensor** sensor_no lcd en | di |(time *time_value*)

- sensor_no : identifies the sensor that will be set up to drive the led.
- time_value: 0-128 seconds.

Enables or Disabled the printing of information on the LCD for the sensor selected by sensor_no. The information for the displayed sensor is refreshed once a second. The *time* parameter defines the total of amount of time the information for a particular sensor is displayed. The default time value is 2 seconds. The maximum display time is 128 seconds.

Example 17: Enable sensor 2 to be displayed on the LCD

```
%>sensor 2 lcd en
 Done! Default Display Time set to 2 seconds!
```

Example 18: Disable sensor 2 to be displayed on the LCD

```
%>sensor 2 lcd di
 Done!
```

Example 19: Set LCD display time for sensor 2 to 5 seconds

```
%>sensor 2 lcd time 5
 Done!
```

5.7.11 Changing the debounce option

Syntax: **sensor** sensor_no debounce *debounce_value*

- sensor_no : identifies the sensor that will be set up to drive the led.
- debounce_value: 0-254

Enables or Disables the debounce option for a particular sensor. Debounce is available only for threshold,discrete and input sensors.
The debounce parameter establishes the number of times a sensor is allowed to be out of spec before it is reported as failing. If debounce is 0 the sensor is reported as failing the first time it is out of spec.

Example 17. Set debounce value for sensor 2

```
%>sensor 2 debounce 2
 Done!
```

In this case, sensor 2 has to fail 3 times before it is reported as failing.

5.7.12 Changing the delay for an output assertion / deassertion

Syntax: sensor delay as | de time_value

- time_value: 0-255 seconds

This command allows the user to define delays between the moment a driver becomes active and the moment the output is changed. The user can specify different delays for assertion and deassertion.
By default both delays are 0, so all outputs are asserted/deasserted immediately after the driver commands an output change.

Example 18. Set a 6 seconds assertion delay for output 1

```
%>sensor 80 delay as 6
 Done!
```

5.8 fancontrol command

Syntax: fancontrol [fanboost boost_level **]** |
[group_no (auto | manual) | (level value) | (minlevel value) |
(temp0 | temp1 | temp2 value)**]** |
[sensor**]**

- boost_level: 5..15
- group_no : 1 – 4
- level and minlevel value : 0 - 15
- temp_no : 0 - 2
- value: Celsius degrees

Functions:

The fancontrol command is used to modify parameters for the fan control algorithm. For more details about the algorithm that is used to control the fan speeds refer to 1.4.1 Fan control.

If the command is used without any parameter it displays the current state of all the fan control groups and their respective parameters.

Example 1: Viewing fancontrol status

```
%>fancontrol

Fans--Control--Current Fanlevel--Manual Fanlevel--Min Fanlevel--Temp0--Temp1--Temp2
  1   Auto            3               1              3         0      30     60
  2   Auto            3               2              3         0      30     60
  3   Auto            3               3              3         0      30     60
  4   Auto            3               4              3         0      30     60
```

```
  Fan Boost Level : 8
```

Each fan control group computes the necessary fan speed using a formula based on the maximum temperature of it's assigned temperature sensors. To view the temperature sensors assigned to each fan group you can use the sensor parameter.

```
%>fancontrol sensor

 Sensors assigned to Fan Group 1:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
 Sensors assigned to Fan Group 2:
   Sensor 27 Temp2 : 19.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
 Sensors assigned to Fan Group 3:
   Sensor 27 Temp2 : 19.00 deg C
   Sensor 28 Temp2 : 20.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
 Sensors assigned to Fan Group 4:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
```

Every fancontrol group can be controlled automatically or manually. When in auto mode the fan level is controlled by the fan control algorithm, and when in manual mode the fan control is equal to the manual level.

Example 2: Setting fancontrol group 4 to manual control

```
%>fancontrol 4 manual
 Done! Fan Group 4 is controlled manually using the manual fanlevel value!
```

Using the fancontrol command the user can change the value for all the algorithm's parameters : manuallevel, minlevel, temp0, temp1, temp2. Each fancontrol group has it's own parameters.

Example 3: Setting manuallevel of fancontrol group 2,minlevel of fancontrol 3,and temp2 of fancontrol 1

```
%>fancontrol 2 manuallevel 14
 Manual Fanlevel = 14

%>fancontrol 3 minlevel 4
 Minimum Fanlevel = 4

%>fancontrol 1 temp2 90
 Temp2=90
```

5.9 pwm command

Syntax: pwm [freq *pwm_no* freq_no**]**

- pwm_no : 1 - 4
- freq_no : 1 - 10

Functions:

If used without parameters displays the frequencies of all the PWM signals.

Example 1: Reading the frequencies of all pwm signals

```
%>pwm
Pwm1=62.50KHz Duty Cycle=28%
Pwm2=62.50KHz Duty Cycle=28%
Pwm3=7.81KHz Duty Cycle=28%
Pwm4=7.81KHz Duty Cycle=28%
```

The pwm command can also be used to change the frequency of the PWM signals.

PWM1 and PWM2 share the same frequency.
PWM3 and PWM4 share the same frequency.
If the frequency is changed for one signal it will change for both in that group.
All 4 PWM signals can have independent duty cycles.

The PWMs can operate at fixed frequencies. The correspondence between the frequency values and *freq_no is* displayed in the following table.

freq_no	Frequency
1	125 KHz
2	62.5 KHz
3	7.81 KHz
4	3.9 KHz
5	1.95 KHz
6	0.97 KHz
7	0.49 KHz
8	122 Hz
9	30 Hz
10	7 Hz

Table 9: PWM Frequency

Example 2: Changing PWM3 and PWM4 frequencies to 3.9KHz

```
%>pwm freq 3 4
 Done!
```

5.10 xmodem command

Syntax: xmodem usersettings|web | sdr | sensoroptions | configfile

Functions:
Sends via RS232 the user settings, the web page or the SDRs.

The sensoroptions file contains behavioral options of Sensors described by SDRs: active levels, outputs for sensor events, input type, output type and other options.

The configfile contains all data needed to describe a system: usersettings, SDRs and sensoroptions.

After the command is entered, the Sysmon goes into data receive mode and waits for the data to be sent. You can then start the file transfer with your terminal program and select XMODEM as the protocol.

When using "**xmodem**" in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.

Example 1: Sending of usersettings

```
%>xmodem usersettings
 Please upload the file...
%>...Done!
```

5.11 logout command

Syntax: logout

Function:
Logs out the current user and permits a new log in.

5.12 scispeed command

Syntax: scispeed 9600 | 19200 | 38400

Functions:
Changes the baud rate at which the CLI for the Sysmon and the bootloader framework operate. For the change to become valid the environment has to be saved using the saveenv command and the Sysmon has to be restarted by using the reboot command.

Example 1:
```
 %>scispeed 9600
Baud rate changed to 9600.Save Environment and reboot.
```

5.13 passw command

Syntax: passw

Function:
Changes the password for the current user.

5.14 reboot command

Syntax: reboot

Function:
Restarts the system monitor.

5.15 uptime command

Syntax: uptime

Function:
Displays the amount of time which has past since the system monitor became operational.

5.16 saveenv command

Syntax: saveenv

Function:
Saves the parameters that were changed. If the modified parameters are not saved, they will be lost at a reboot.

! The operation may take up to 3 seconds. While the save procedure is active the monitoring stops. Thus it is recommended to enter all the desired changes and use saveenv once, rather than enter individual changes followed by multiple saveenv commands.

5.17 version command

Syntax: version

Function:
Prints information about the System monitor : Part Number,Software Version,MAC Address, and Serial Number.

5.18 restore command

Syntax: restore

Function:
Restores all parameters to the default values. For the restore to be complete a reboot is necessary.

5.19 lanconfig - command

Syntax:

lanconfig [ip | mask| gateway [*address*]] **| [dhcp [***on* **|** *off*]] **|**
speed[auto | 100mb | 10mb**]]**

Functions:
Readout or setting of network parameters.

- **no parameter** – return of IP, mask and gateway addresses of the LAN interface
- **ip** – returns IP address of the Sysmon
- **mask** – returns network mask
- **gateway** – returns standard gateway
- **address** – if any address is inputed, it's value is assigned to the parameter entered before it
- **dhcp** – displays the current dhcp state -enabled/disabled.
- on – if entered after dhcp enables it
- off – if entered after dhcp disables it
- **speed** – parameter that defines the LAN speed
  auto – the LAN interface speed will be defined by the auto-negotiate protocol
  100mb – the lan speed is forced to 100 Mbps
  10mb – the lan speed is forced to 10 Mbps

! If DHCP is enabled the IP, Mask and Gateway displayed by lanconfig are the ones the Sysmon has negotiated. When DHCP is disabled lanconfig displays the user defined values for IP, Mask and Gateway.

After a new address is set, for the change to become effective, it must be saved with **saveenv** and the Sysmon restarted, either using the reboot command or using the reset key.

Example 1: Readout of all network parameters

```
%>lanconfig
```

```
IP=193.155.166.51
Mask=255.255.255.0
Gateway=193.155.166.100
```

Example 2: Readout of the IP address

```
%>lanconfig ip

IP=193.155.166.51
```

Example 3: Changing the IP address

```
%>lanconfig ip 196.100.100.1

IP=196.100.100.1
```

Example 4: Force the link to 10Mbps

```
%>lanconfig speed 10mb
 Speed: 10Mbps
```

5.20 upgradefirmware command

Syntax: upgradefirmware *server_ip_address file_name*

Functions:
Upgrade the firmware using TFTP file transfer over Ethernet interface. The *server_ip_address* parameter represents the IP address of the TFTP server where the firmware image, indicated by *file_name* parameter, is located.
After the firmware was successful transferred from the TFTP server the sysmon must be restarted.

5.21 help command

Syntax: help

Functions:
Displays a list of all available commands. For all commands only the syntax is displayed. For a detailed description you should use the user manual.

5.22 vme command

Syntax: vme [(*on* | *off*) | (*powerfail* [activehigh|activelow]) |
(*activevoltage* [hex_voltage_value]) |
(*threshold* lnr|lc|lnc) | (sysreset [assert | deassert | (user on|off)])**]**

- hex_voltage_value : 8 bits hex value

Functions:

The system monitor generates ACFAIL and SYSRESET signals in accordance with VME specification ANSI/VITA 1-1994. The "vme" command is used to display or modify parameters used in VME signals generation.

If the "vme" command is used without any parameter it displays all the available information about the VME signals.

Example 1: Readout of all vme parameters

```
%>vme

 VME ON

 SYSRESET*  : 0
 ACFAIL*    : 0
 POWERFAIL* : 1
 SYSFAIL*   : 1

 Voltages active for VME signals generation:
   Sensor 2  +3.3V
   Sensor 5  -12V
 Active threshold for VME signals generation: Lower Non-Critical
```

In this example the Sysmon is set to use the Lower Non-Critical threshold of voltages +3.3V and -12V for ACFAIL* and SYSRESET* signals generation.

Any change in vme parameters will be active after a reboot. The change is active only if it is saved using saveenv command.

VME signal generation could be turned on or off using "vme on" or "vme off" comands. If the VME signals are turned off the SYSRESET signal could be used as a push button reset signal.

Example 2: Disable VME signals generation:

```
%>vme off
 VME signals generation has been disabled!
 Save changes and reboot!
```

The POWERFAIL signal can be defined as active high or active low.

Example 3: Defining VME POWERFAIL signal activelow:

```
%>vme powerfail activelow
 POWERFAIL signal is active low!
 Save changes and reboot!
```

For ACFAIL* and SYSRESET generation any combination of the voltage sensors can be used. This combination can be described by a hex value. If a bit of *hex_voltage_value* is 1 the corresponding voltage will be used in VME signals generation. For example if +3.3V, +5V, and V5_MON will be used *hex_voltage_value* = 0x13.

hex_voltage_value bit	Value
7	V8_MON
6	V7_MON
5	V6_MON
4	V5_MON
3	-12V
2	+12V
1	+5V
0	+3.3V

Table 10: Hex voltage Bit Mask

Example 4: Setting +12V and V8_MON as active voltage for VME signal generation:

```
%>vme activevoltage 0x84
 Done!
```

The VME signals are controlled by a threshold infringement. The same threshold is used for all the active voltages and it can be chosen between : *Lower Non-Recoverable (lnr), Lower Critical (lc), Lower Non-Critical(lnc).*

Example 5: Setting Lower Critical as active threshold for VME signal generation:

```
%>vme threshold lc
 Lower Critical threshold of all active voltages is active for VME signals generation!
 Save changes and reboot!
```

The "vme sysreset" command could be used to generate a reset pulse. The "vme sysreset assert" will force the reset signal LOW while "vme sysreset deassert" will release the signal. In order to generate a reset signal the administrator (admin login profile) must allow this using "vme sysreset user on"

Example 6: Generate a SYSRESET* pulse:

```
%>vme sysreset
 System restarted...
```

5.23 led command

Syntax: led

Functions:

Displays the current status of all the installed leds. For multi-color leds the status can be: green/red/amber, and for single color leds the status can be on/off. Leds that have no active drivers will have the "not used status" and will be turned off.

Example 1: Readout of led status

```
%>led

----------------------------Sensor List---------------------------
--no--Name------------Type----Value--Unit---State-----------------
* 101  Led1           Led     Green
* 102  Led2           Led     Not Used
* 103  Led3           Led     Red
* 104  Led4           Led     Red
```

```
* 105  Led5            Led     Amber
* 106  Led6            Led     Amber
* 107  Led7            Led     Not Used
* 108  Led8            Led     Green
* 109  Led9            Led     Off
* 111  Led11           Led     On
```

5.24 sol command

Syntax: sol [on|off]

Functions:

Displays or changes the current status of the Serial Over Lan (SOL) configuration.
Whenever the SOL mode is activated the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on "serial" profile. The data received from the telnet connections opened in "serial" mode is transferred to the serial port.

5.25 sel command

Syntax: sel count | clr | print [start_record_no [end_record_no]]

Functions: The command can display the number of sel entries, display a particular set of these entries or clear the sel.
To print the whole log: ***sel print***
To print all the records starting with a particular one: ***sel print record_no***
To print all records in a give interval: ***sel print start_record_no end_record_no***

Example 1: Readout of SEL

```
 %>sel print
-------------------------------------------------------------------------------
                             Sensor  Event Log

Record ID.  Days: h: m: s  Sensor No. Name            Event  Ev.Dir  Value  Threshold
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
  0x0001     000:00:00:00     97      Sysmon Power ON   1 (Asserted)
  0x0002     000:00:00:14     2       +3.3V             LNC    As     2.82     2.86
  0x0003     000:00:00:14     3       +5V               LC     As     0.16     4.76
  0x0004     000:00:00:14     4       +12V              LC     As     10.53    11.38
  0x0005     000:00:00:14     5       -12V              LC     As     -12.64   -12.64
```

Example 2: SEL clearing

```
 %>sel clr
  Done! Sel is empty!
```

5.26 sanitize command

Syntax: sanitize

Functions: the command erases all the non-volatile memories of the Sysmon. Only admin can sanitize.

Example 1: Sanitize process

```
%>sanitize
 Process Started!
*Sanitizing 256Kb EEPROM ...........
```

```
*Sanitizing 2MB External Flash ..............
*Sanitizing 512KB Internal Flash ............

 Done! The System will restart!
```

5.27 elapsedtime command

Syntax: elapsedtime

Functions: displays the total amount of time the Sysmon has been in service

Example 1: Check out of elapsed time

```
%>elapsedtime

 Elapsed Time in service [days hours:minutes:seconds] = 2 days 00:15:12
```

5.28 time command

Syntax: time[hh:mm:ss]

Functions: displays or sets the current time

Example 1: Check out of time

%>time

Time [hh:mm:ss] = 16:52:27

Example 2: Setup of time

%>time 16:53:00
Done!

5.29 date command

Syntax: date[dd.mm.yyyy]

Functions: displays or sets the current date

Example 1: Check out of date

%>date

Date [dd.mm.yyyy] = 04.09.2012

Example 2: Setup of date

%>date 13.10.2013
Done!

To keep the internal real time clock running, a 3V backup battery type CR2025 is required

5.30 telnetping command

Functions: Defines or displays the timeout for the telnet ping interval. If there is no communication on Telnet for the timeout interval, measured in seconds, the sysmon sends a Telnet WILL ECHO . This helps to release an abnormal terminated telnet connection on the server side (for example server lost power).

Example 1: Set telnet ping interval to 5 seconds

```
%>terlnetping 5

 Telnet ping interval= 5
```

5.31 exit command

Syntax: exit

Functions: Closes a telnet connection.

Example 1: Close a telnet connection

```
%>exit
Closing connection..
```

6 Restore to factory defaults procedure

Only an admin can perform a system restore. In order to restore all parameters to their default value the following steps need to be followed:

- Login using the admin account (for more details refer to *5. Command Line Interface (CLI)*)
- Use the restore command
- Reboot. By using the reboot command.

When using restore the System Monitor disregards all the changes applied to the SDRs, Sensor options and User Settings and uses a predetermined set for all these parameters. This predetermined set of parameters may be different then the one loaded on the System Monitor when it was shipped out.

To go back to a particular setup you have to use xmodem configfile, and upload a Configuration file that contains the required setup.

7 Appendix

7.1 Technical data

Power supply	+5V DC
Current consumption	500mA max.
Operating temperature	0°C to +70°C
Storage temperature	-40°C to +85°C
Physical dimensions	100.00 x 160.00 mm (PCB)
Load of PWM Signal	short-circuit-proof to GND. Open Collector
PWM signal level	5V TTL
Signal level - inputs	5V TTL
Signal level - outputs	5V TTL
Max. loading - outputs	IOH: -25mA IOL: 25mA

Table 11: Technical Data

7.2 Pin Assignment

Power

Pin	Signal
1	
2	
3	GND
4	GND
5	+5V
6	3.3V OUT
7	PWRFAIL

VMEAS 1

Pin	Signal
1	SYSRESET#
2	ACFAIL#
3	U4 -12V
4	GND
5	U3 +12V
6	GND
7	U1 +3.3V
8	GND
9	U2 +5V
10	GND

VMEAS 2

Pin	Signal
1	SYSFAIL#
2	VREF
3	U7 IN
4	GND
5	U8 IN
6	GND
7	U6 IN
8	GND
9	U5 IN
10	GND

FCON

Pin	Signal
A1	GND
A2	SYSRESET#
A3	Sysfail#
A4	ACFAIL#
A5	GND
A6	GND
B1	GND
B2	U4_-12V
B3	U3_+12V
B4	U1_+3.3
B5	U2_+5V
B6	NC

Fan

Pin	Signal
1	+3.3V
3	+3.3V
5	GND
7	GND
9	TACH_10
11	PWM_2
13	TACH_8
15	GND
17	PWM_1
19	TACH_4
21	TACH_3
23	GND
25	TACH_1

Pin	Signal
2	
4	
6	PWM_3
8	TACH_11
10	TACH_9
12	+5V
14	TACH_7
16	TACH_6
18	TACH_5
20	+5V
22	PWM_0
24	TACH_2
26	TACH_0

INPUTS

Pin	Signal
1	IN1
3	IN3
5	IN4
7	GND
9	IN7
11	IN9
13	IN10
15	GND
17	IN13
19	IN15

Pin	Signal
2	IN2
4	+5V
6	IN5
8	IN6
10	IN8
12	+5V
14	IN11
16	IN12
18	IN14
20	IN16

Temp 1

Pin	Signal
1	VREF
2	Temp3
3	VREF
4	Temp2
5	VREF
6	Temp1

Temp 2

Pin	Signal
1	VREF
2	Temp6
3	VREF
4	Temp5
5	VREF
6	Temp4

OUTPUTS

Pin	Signal
1	OUT1
3	OUT3
5	OUT4
7	GND
9	OUT7
11	OUT9
13	OUT10
15	GND
17	OUT13
19	OUT15

Pin	Signal
2	OUT2
4	+5V
6	OUT5
8	OUT6
10	OUT8
12	+5V
14	OUT11
16	OUT12
18	OUT14
20	OUT16

I2C_EXT

Pin	Signal
1	SDA_2
2	SCL_2
3	SYS_RST
4	INT_2
5	+5V
6	GND

Digital Temp

Pin	Signal
1	+5V
2	SDA_3
3	GND
4	SCL_3

I2C_3V

Pin	Signal
1	SCL_5
2	3.3V
3	SDA_5
4	GND
5	SDA_4
6	SCL_4

Table 12: Pin Assignment

7.3 Dimensions PCB of fix mounted version 041-039

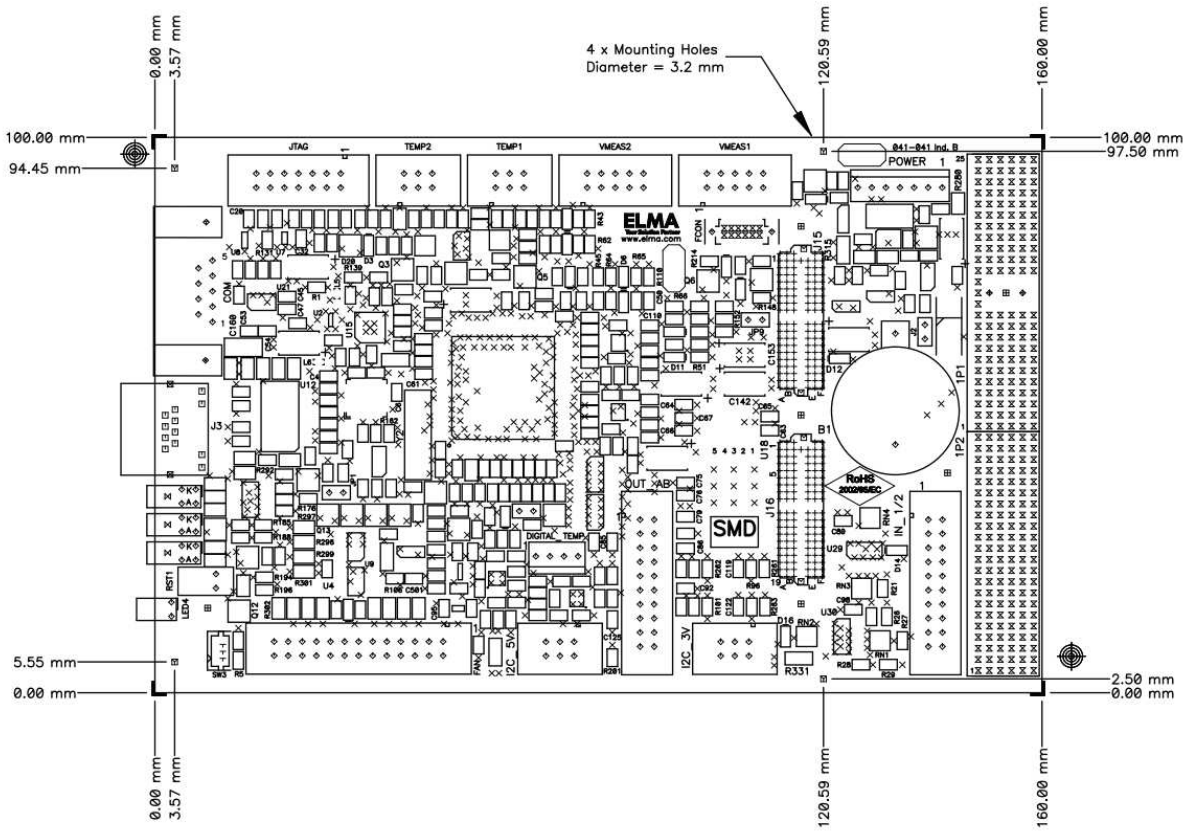

Figure 4: PCB Dimensions

7.4 Layout / position of the connectors

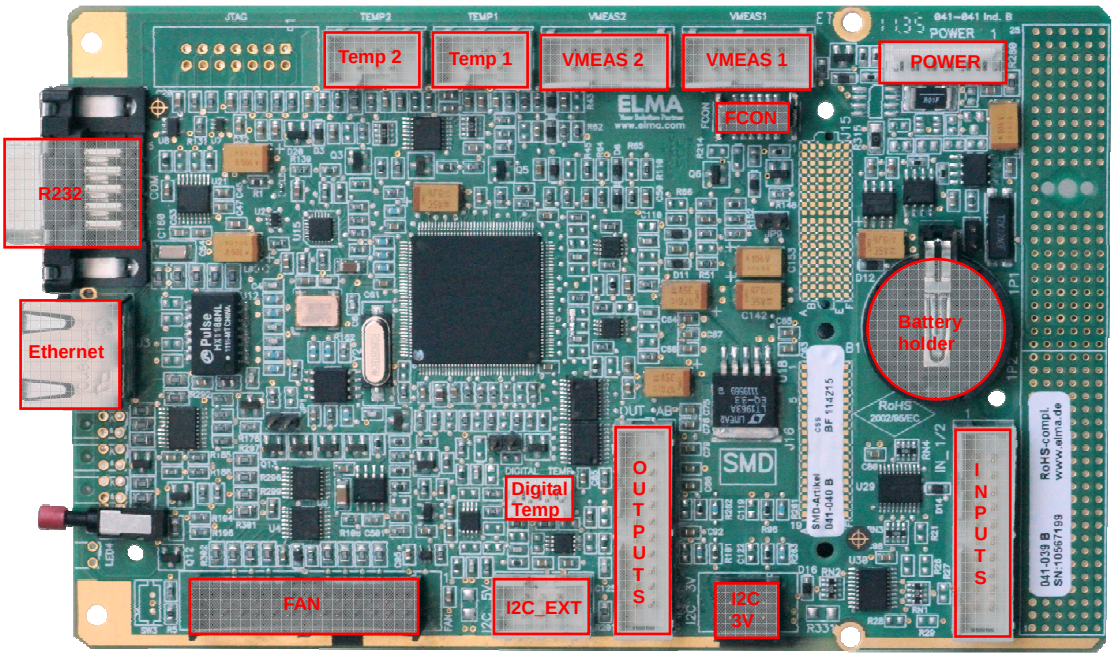

Figure 5: Connector Positions

7.5 Versions and accessories

Order number	Description
041-039	Fix mounted version
042-641	Pluggable version with front panel. (a suitable mating connector slot for the Sysmon must be available on the backplane!)
024-927	LED display complete with 0.8 m cable and mating connector (front plate not included in delivery)
030-160	LCD display
041-855	FanController add-on module for controlling 12V fans without PWM signal input
042-751	Adapter board for pluggable version 042-641
044-003	Set consisting of pluggable version 042-641 and adapter board 042-751
041-970	Hall-Sensor board (5V/50A 3.3V/50A 12V/20A -12V/20A) for analogue current measurement
xxx-xxx *	ADC extension board for digital (I2C) current and voltage measurement

Table 13: Versions

Ruggedized version (extended temperature range, conformal acrylic coating) available on request.

*For all available versions of the ADC extension board please contact us.

Software and Updates

Current Firmware, Configuration-Tool and a detailed user manual can be downloaded from our FTP-Server:

ftp://ftp.elma.de/ELMA_Sysmon3Gen
User: download
Password: elma

For more information please contact us

ELMA Electronic AG
Hofstrasse 93, Postfach
CH-8620 Wetzikon
Phone +41 44 933 41 11
Fax +41 44 933 42 15
sales@elma.ch

ELMA Electronic France SASU
ZA du Buisson Rond
F-38460 Villemoirieu
Phone +33 4 37 06 21 10
Fax +33 4 37 06 21 19
sales@elma-electronic.fr

ELMA Electronic Inc.
44350 Grimmer Blvd.
Fremont, CA 94538
Phone +1 510 656 3400
Fax +1 510 656 3783
sales@elma.com

ELMA Bustronic Corp.
44350 Grimmer Blvd.
Fremont, CA 94538
Phone +1 510 490 7388
Fax +1 510 490 1853
sales@elmabustronic.com

Optima Electronic Packaging Systems
1775 MacLeod Drive
Lawrenceville, GA 30043
Phone +1 770 496 4000
Fax +1 770 496 4041
sales@optimaeps.com

ELMA Electronic Inc.
760 Veterans Circle
Warminster, PA 18974-3531
Phone +1 215 956 1200
Fax +1 215 956 1201

ELMA Electronic GmbH
Stuttgarter Str. 11
D-75179 Pforzheim
Phone +49 7231 97 34 0
Fax +49 7231 97 34 97
info@elma.de

ELMA Electronic Israel Ltd.
34, Modi'in St.
Sgula I.Z., Petach-Tikva, 49271 IL
Phone +972 3 930 5025
Fax +972 3 931 3134
sales@elma.co.il

ELMA Electronic UK Limited
Solutions House
Fraser Road, Priory Business Park
Bedford, MK44 3BF
Phone +44 1234 838 822
Fax +44 1234 836 650
sales@elma-uk.com

ELMA Electronic Romania SRL
Calea Buziasului Nr. 11 A
300699 Timisoara
Phone +40 256 222 290
Fax +40 256 222 490

ELMA Electronic (China) Co., Ltd.
8F, 355, Fu Te Road (West 1)
Wai Gao Qiao Free Trade Zone
Pudong District
Shanghai, 200131, China
Phone +86 21 5866 5908
Fax +86 21 5866 5918
sales@elmachina.com

Elma Asia Pacific Pte Ltd
115-A Commonwealth Drive
#03-14 Tanglin Halt Industrial Estate
149596 Singapore
Phone +65 6479 8552
Fax +65 6479 8662

Your local solution partner:

PEFC

8/2012

ELMA
Your Solution Partner