Moxa Proactive Monitoring User's Manual

Edition 1.0, September 2015

www.moxa.com/product

Moxa Proactive Monitoring User's Manual

Copyright Notice

Trademarks

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.

Moxa provides this document as is, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Technical Support Contact Information

www.moxa.com/support

Moxa Americas
Toll-free: 1-888-669-2872
Tel: +1-714-528-6777
Fax: +1-714-528-6778

Moxa China (Shanghai office)
Toll-free: 800-820-5036
Tel: +86-21-5258-9955
Fax: +86-21-5258-5505

Moxa Europe
Tel: +49-89-3 70 03 99-0
Fax: +49-89-3 70 03 99-99

Moxa Asia-Pacific
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

Moxa India
Tel: +91-80-4172-9088
Fax: +91-80-4132-1045

Table of Contents

1

Installing and Using Moxa Proactive Monitoring

The following topics are covered in this chapter:

- **Installing Moxa Proactive Monitoring**
  - Installation Steps
  - Starting or Stopping the Moxa Proactive Monitoring Daemon
- **Monitoring System Status**
  - How to display the Moxa Proactive Monitoring UI
  - How to use the Moxa Proactive Monitoring UI
  - Customizing Your Own Monitoring Items
- **Setting the System Alarm**
  - Changing Alarm Settings
  - Performing an Alarm Action
  - Stopping an Alarm Action
  - Testing an Alarm Action

Installing Moxa Proactive Monitoring

Installation Steps

1. Upload **mxpromon_1.0.0_amd64.deb** to the target machine.
2. Type the following command to install Moxa Proactive Monitoring:

```
root@Moxa:~# dpkg -i mxpromon_1.0.0_amd64.deb
```

Starting or Stopping the Moxa Proactive Monitoring Daemon

The Moxa Proactive Monitoring daemon **pro_mond** executes in the background when the system boots up, and then continuously monitors the status of the target machine. It logs alarm messages and performs alarm actions, if these features are activated. Take the following steps to stop or start the daemon.

How to stop the daemon

Type the following command to stop the daemon:

```
root@Moxa:~# /etc/init.d/pro_mond stop
```

How to start the daemon

Type the following command to start the daemon:

```
root@Moxa:~ # /etc/init.d/pro_mond start
```

How to restart the daemon

Type the following command to restart the daemon:

```
root@Moxa:~ # /etc/init.d/pro_mond restart
```

How to prevent the daemon from executing on boot up

If you don’t want to execute the daemon when the system boots up, use the following command to turn it off:

```
root@Moxa:~ # insserv -r pro_mond
```

How to force the daemon to execute on boot up

If you want to execute the daemon when the system boots up, use the following command to turn it on:

```
root@Moxa:~ # insserv pro_mond
```

Monitoring System Status

NOTE We suggest using full screen viewing to display the Moxa Proactive Monitoring user interface. **Note:** You cannot execute more than one instance of the Moxa Proactive Monitoring application at the same time.

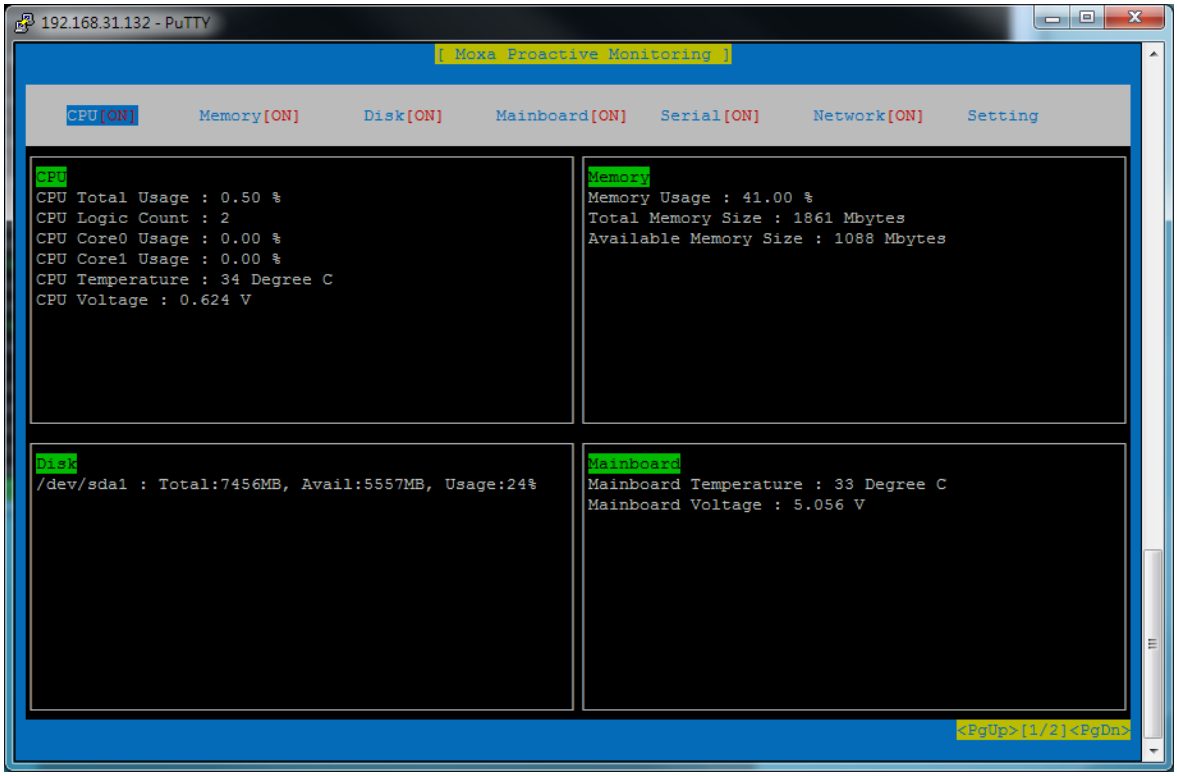
How to display the Moxa Proactive Monitoring UI

Use the following command to display the Moxa Proactive Monitoring user interface:

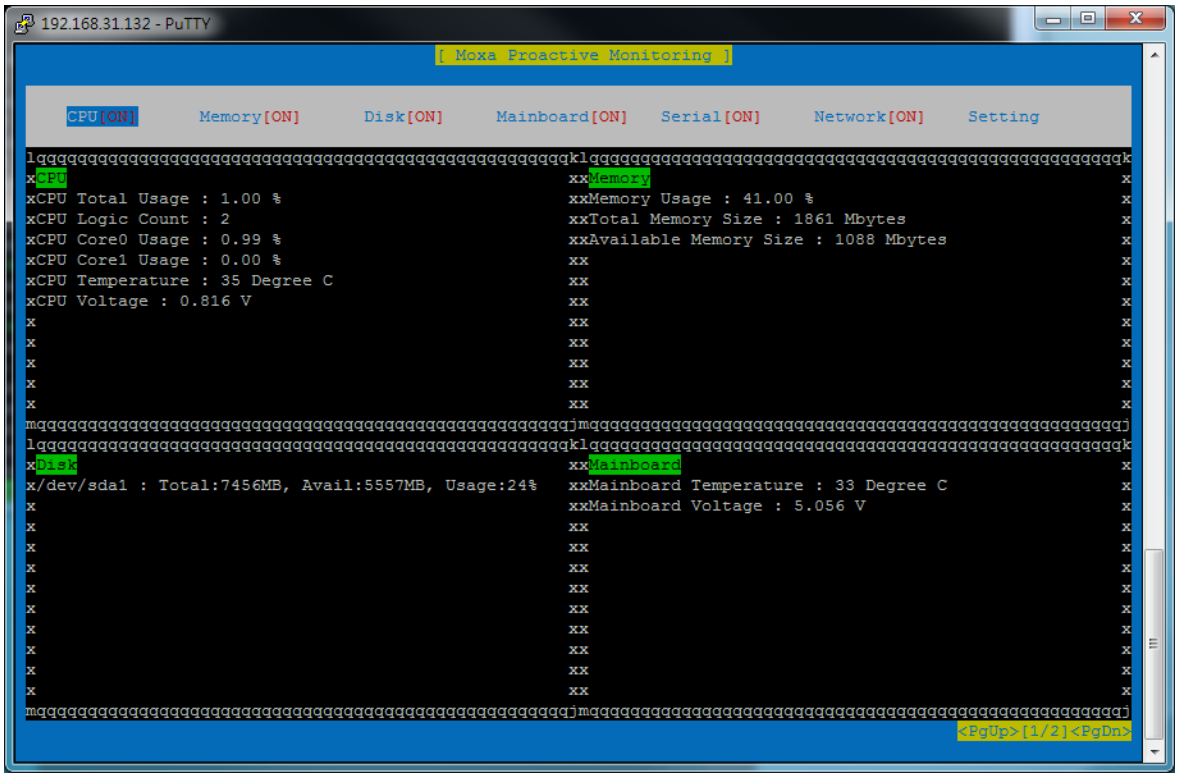
```
root@Moxa:~ # pro_mon
```

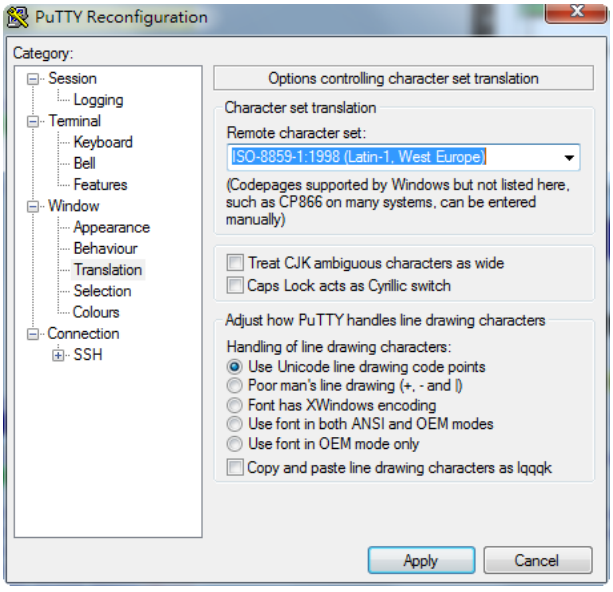
How to use the Moxa Proactive Monitoring UI

Four system status items will be displayed at the same time. To display other system status items, use your keyboard's **<Page Up>** or **<Page Down>** buttons to scroll up or down to the next page of status items.



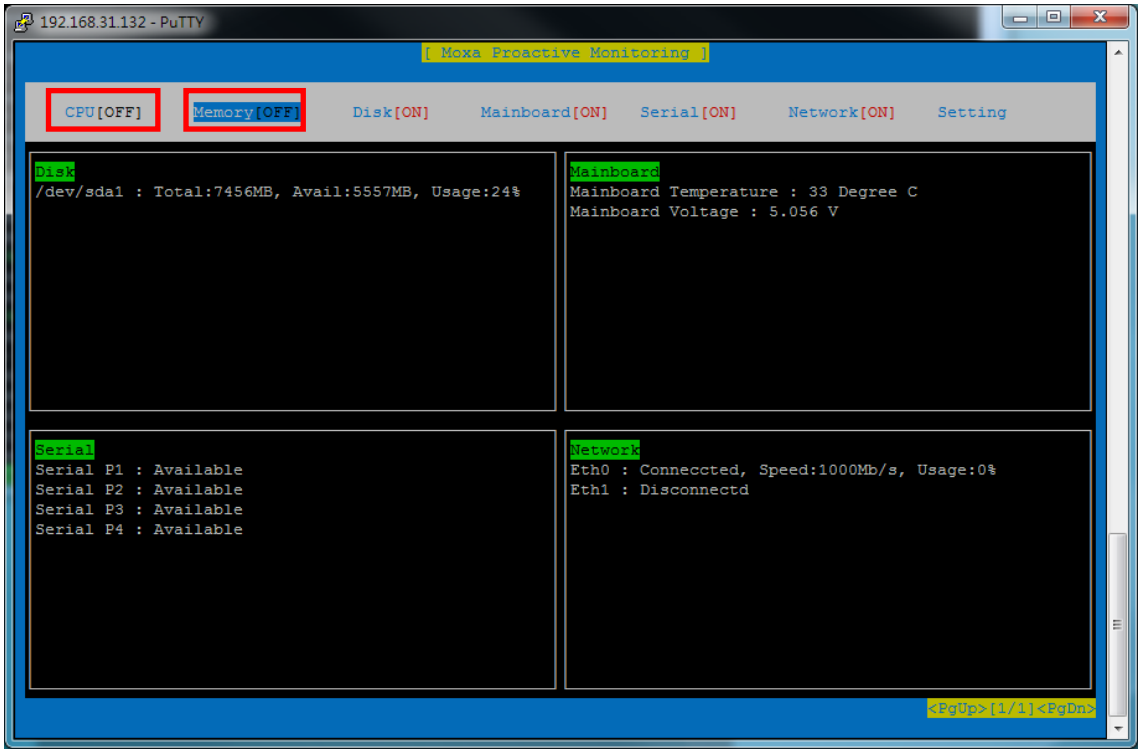
NOTE If garbage characters appear in the UI screen, as illustrated in the following screenshot, you may be able to rectify the situation by changing your computer's "Remote character set" from UTF-8 to a different encoding (ISO-8859-1, for example), and then restarting **pro_mon** (see the previous section).



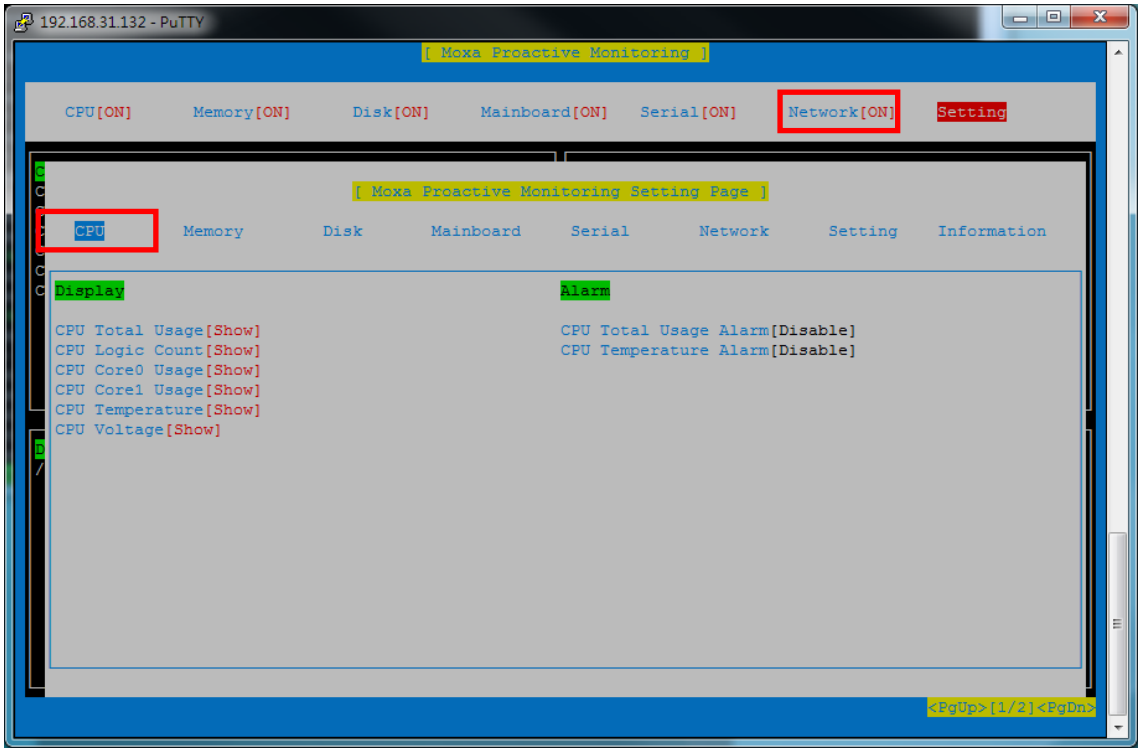


Customizing Your Own Monitoring Items

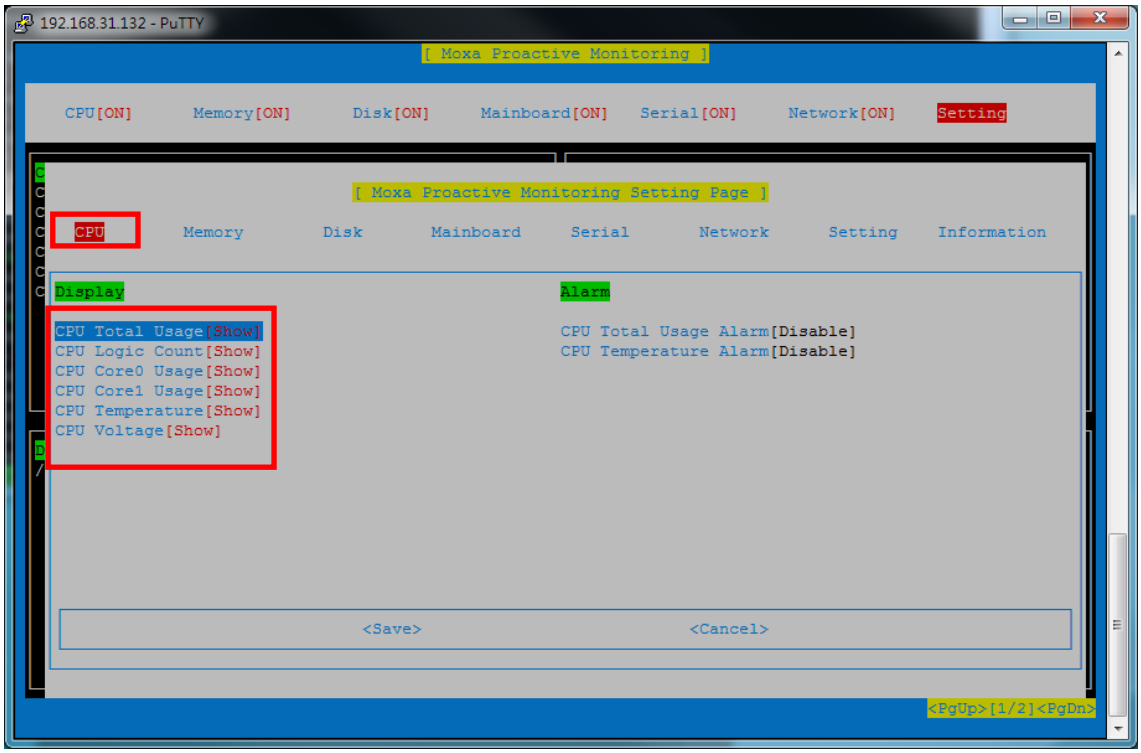
1. You can select your own system status to display in the UI window by turning on or off the status of different monitoring items. For example, if you don't want to show the CPU and memory status, you can turn off these features by pressing **<space>** or **<Enter>** on your computer's keyboard.



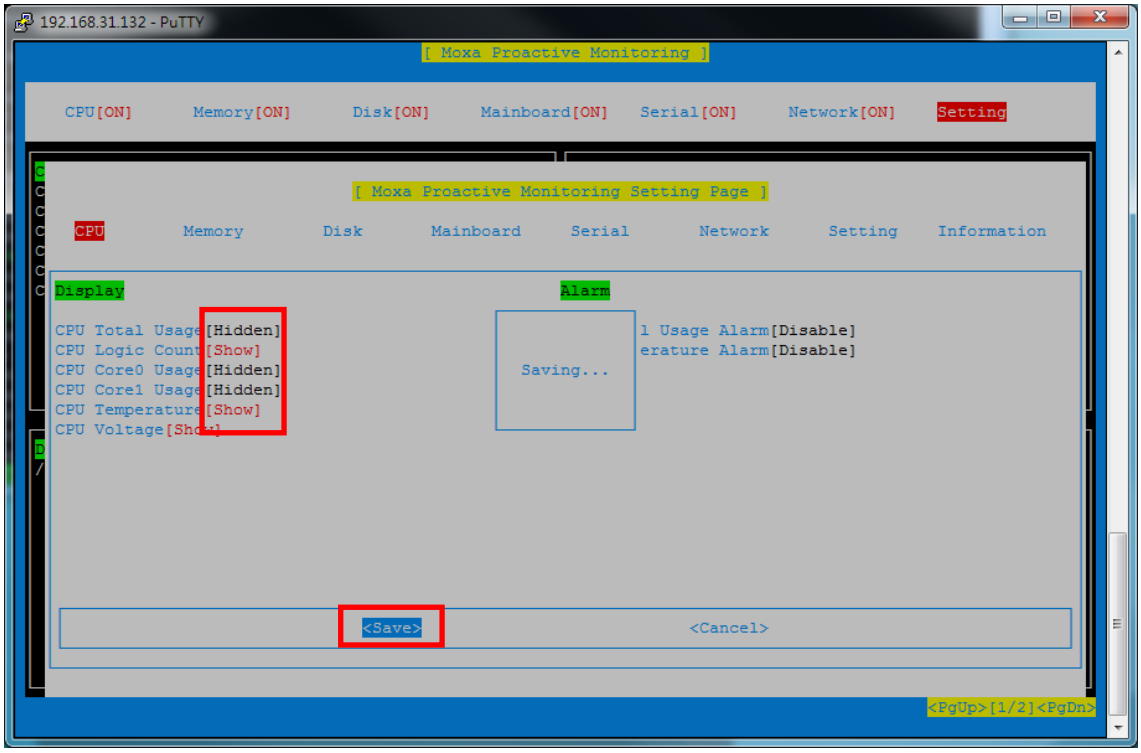
2. Furthermore, you can customize which items in each monitoring status category will be displayed. For example, if you don't want to see the CPU usage of each core, you can turn it off. To achieve this, select **Setting** and then press **<Space>** or **<Enter>** to enter the settings page**.**



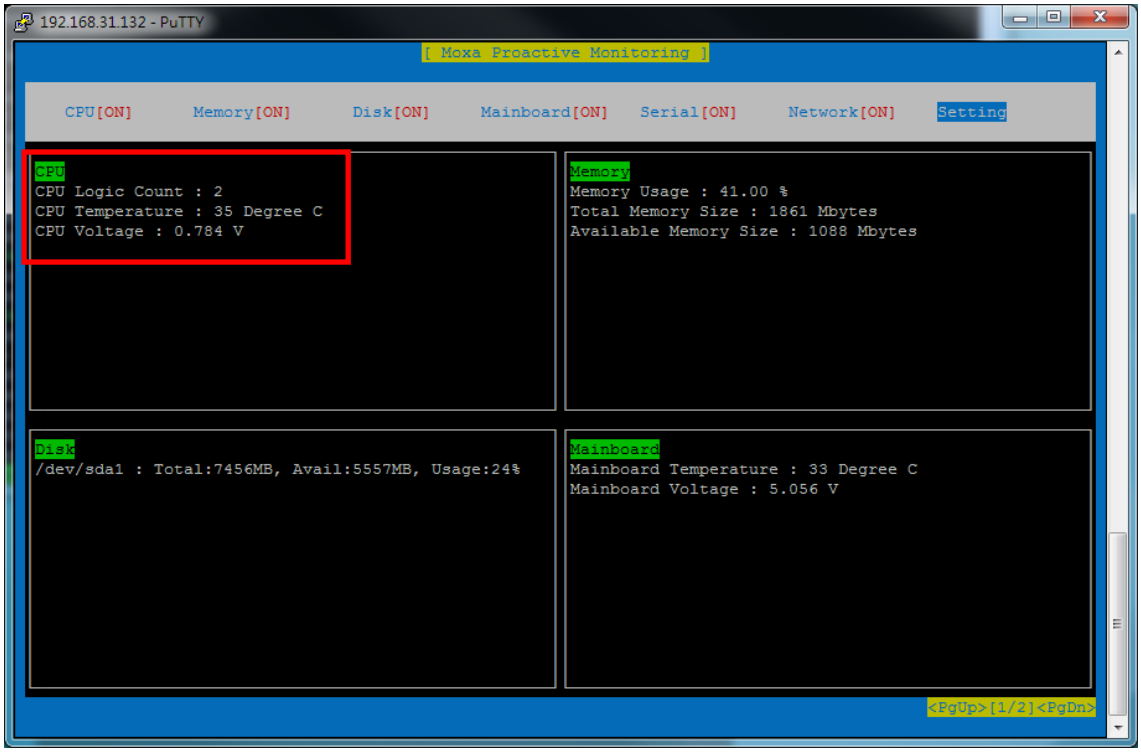
3. Choose which CPU items will be displayed, and which will be hidden.



4. For example, as shown below, you can turn off **CPU Usage,** and then press **<Tab>** to switch to the save menu to save the setting.



5. Press **<Esc>** to return to the previous page. The CPU status will be updated in accordance with your selection.

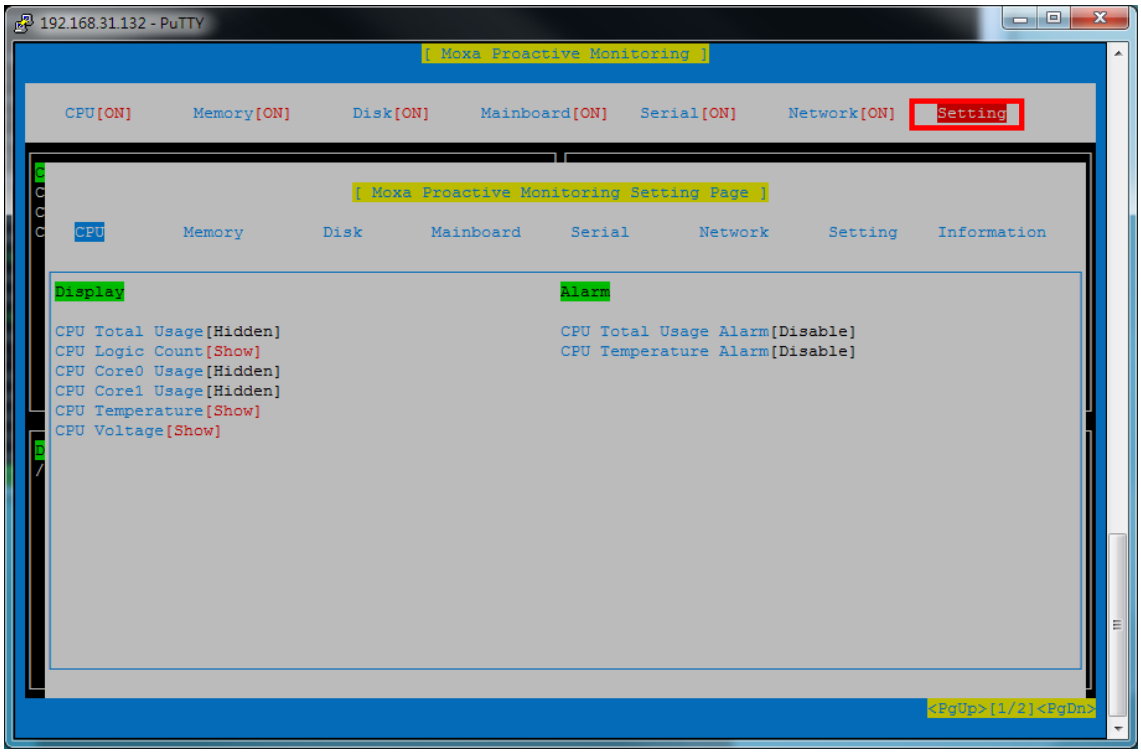


Setting the System Alarm

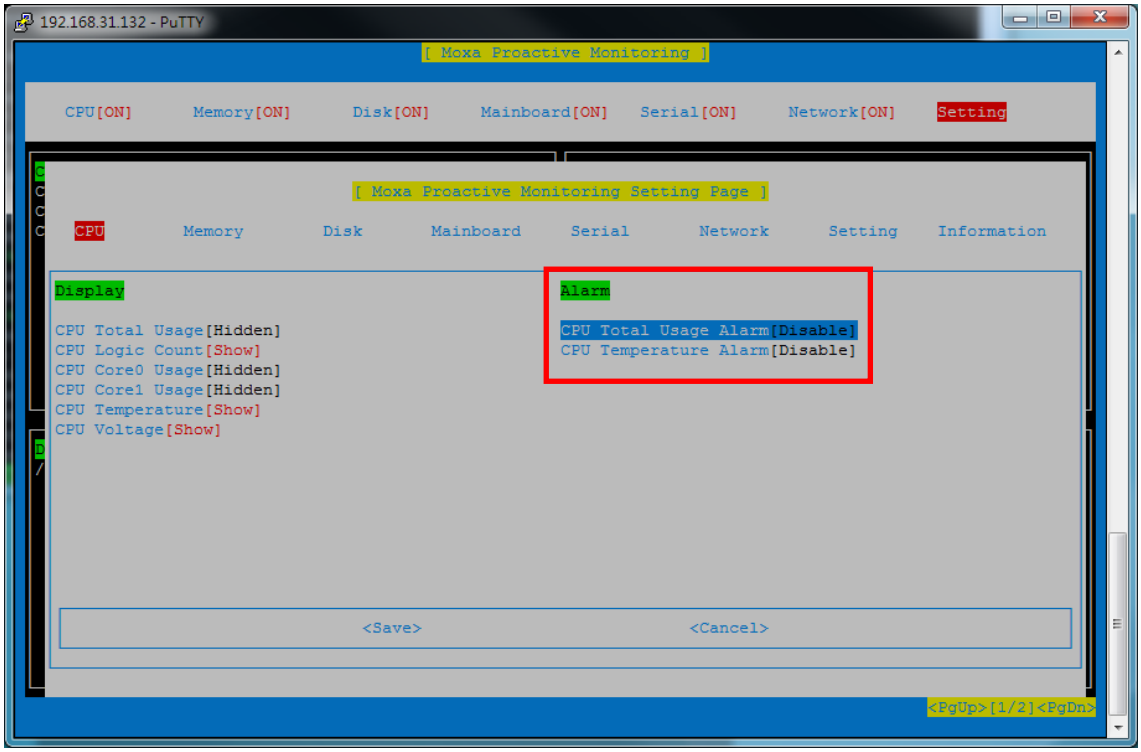
Take the following steps to configure the system alarm settings.

Changing Alarm Settings

1. Click **Setting**.

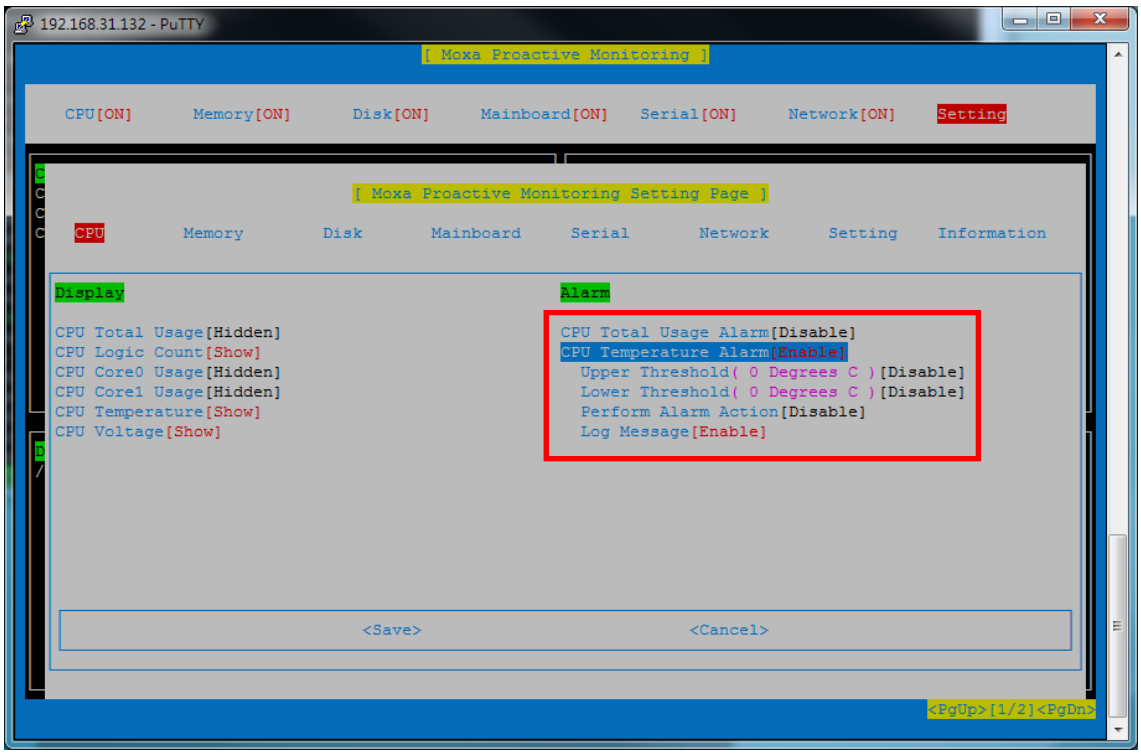


2. Click the CPU item under **[Moxa Proactive Monitoring Setting Page]** under **Alarm**, located near the middle of the page, and then configure the alarm settings here to monitor the system status and perform the alarm action when the system status is over the threshold value of the alarm settings.

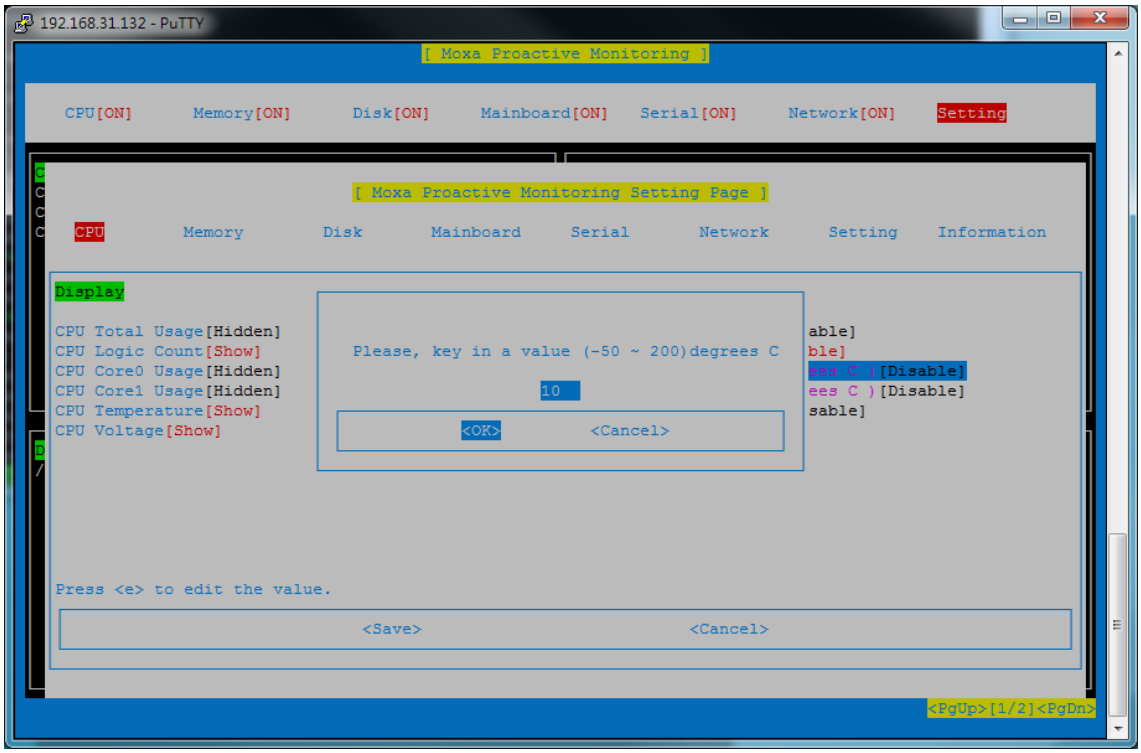


3. To configure the alarm settings, press **<Space>** or **<Enter>** to turn on the alarm. The following example shows the CPU Temperature alarm setting.

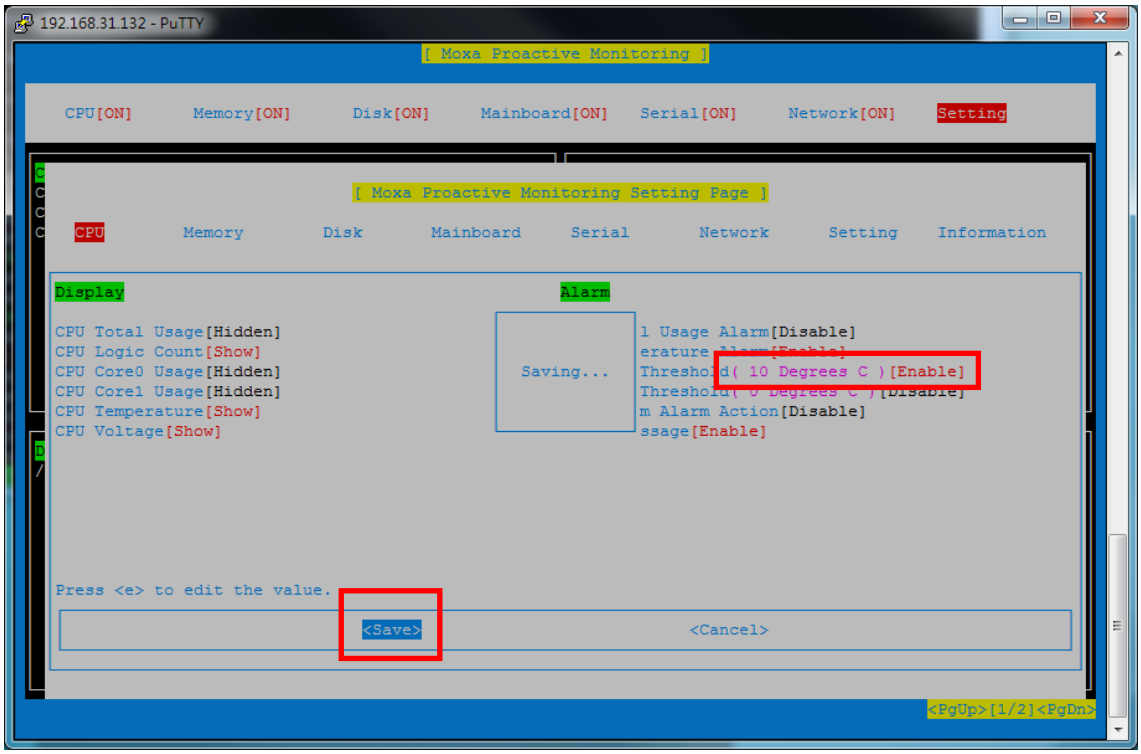
NOTE When you turn on the alarm option, by default the log message option will also turn on. If you don't want to turn on the log message option, you can turn it off yourself.



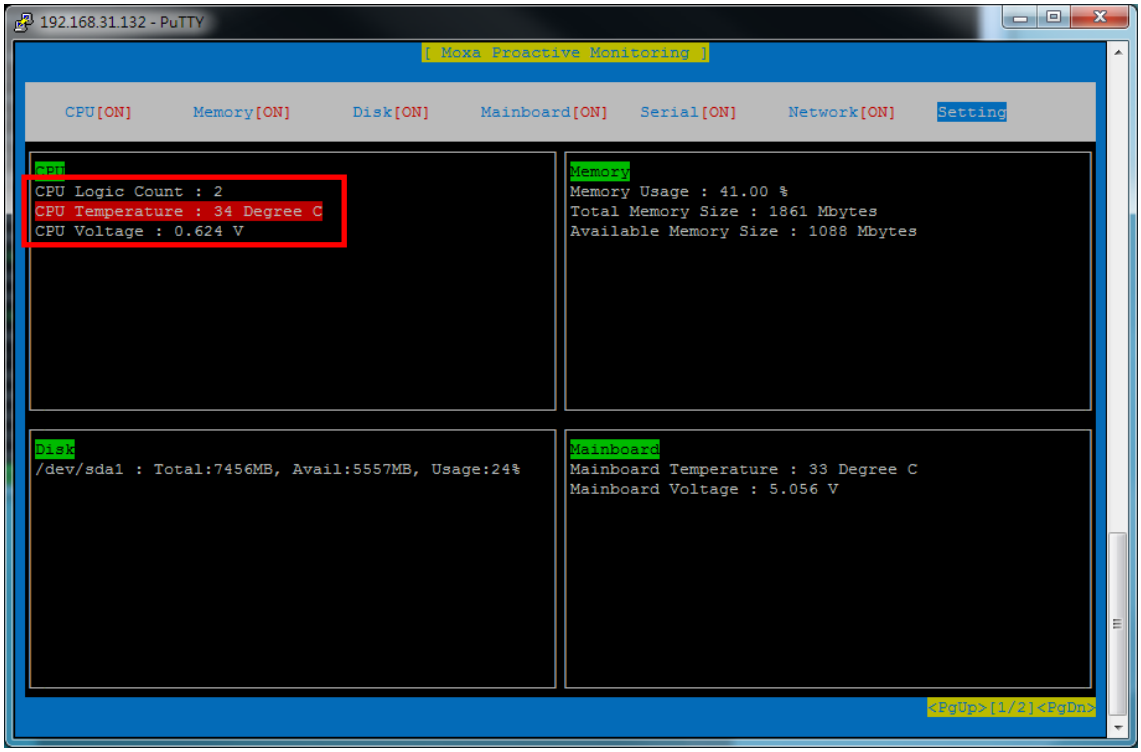
4. After turning on the CPU Temperature alarm, you need to set the threshold value. For example, you can select the upper threshold item and press **<e>** to edit the upper threshold value of CPU Temperature as 10°C and then press **<Space>** or **<Enter>** to enable this option. In this case, when the CPU Temperature exceeds 10°C, the info on the window will change to red, and the alarm will be logged into the log file.



5. Next, press **<Tab>**, and then press **<Space>** or **<Enter>** to save your setting.



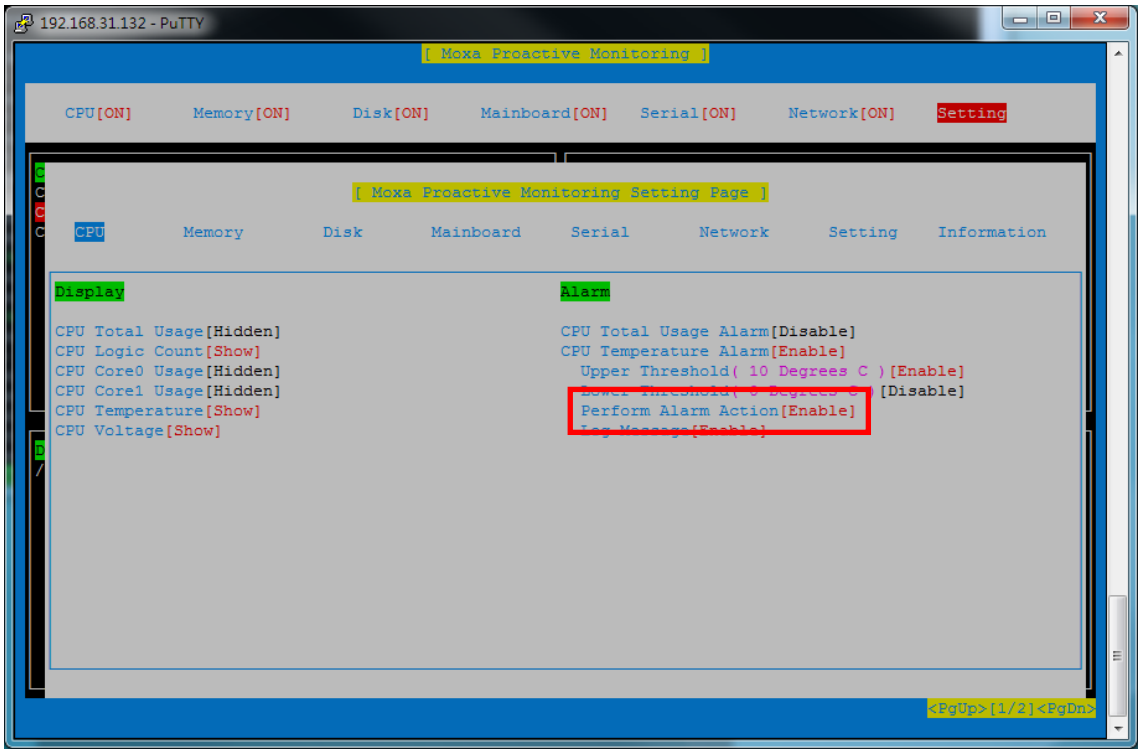
6. Continuing with the previous example, if the CPU temperature exceeds 10°C, the CPU Temperature will be highlighted in red, and the alarm will be logged in the log file. The log file path is:
   /var/log/pro_mon/[date]_pro_mon_log




```
root@Moxa:~# tail /var/log/pro_mon/20150902_pro_mon_log
Wed Sep  2 18:15:21 CST 2015 : CPU Temperature (36 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:15:31 CST 2015 : CPU Temperature (35 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:15:41 CST 2015 : CPU Temperature (34 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:15:51 CST 2015 : CPU Temperature (34 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:01 CST 2015 : CPU Temperature (34 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:11 CST 2015 : CPU Temperature (34 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:21 CST 2015 : CPU Temperature (33 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:31 CST 2015 : CPU Temperature (33 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:41 CST 2015 : CPU Temperature (34 degrees C) is over the upper threshold (10 degrees C)
Wed Sep  2 18:16:51 CST 2015 : CPU Temperature (36 degrees C) is over the upper threshold (10 degrees C)
root@Moxa:~#
```

Performing an Alarm Action

1. If you want to perform the alarm action when the alarm occurs, check the **Perform alarm action** item, turn it on, and then save this setting.



2. After you save the setting, Moxa Proactive Monitoring will perform the alarm action when the alarm occurs. You can edit **/sbin/mx_perform_alarm** to change the alarm action.

   The script will set the first digital output to high in the default setting.

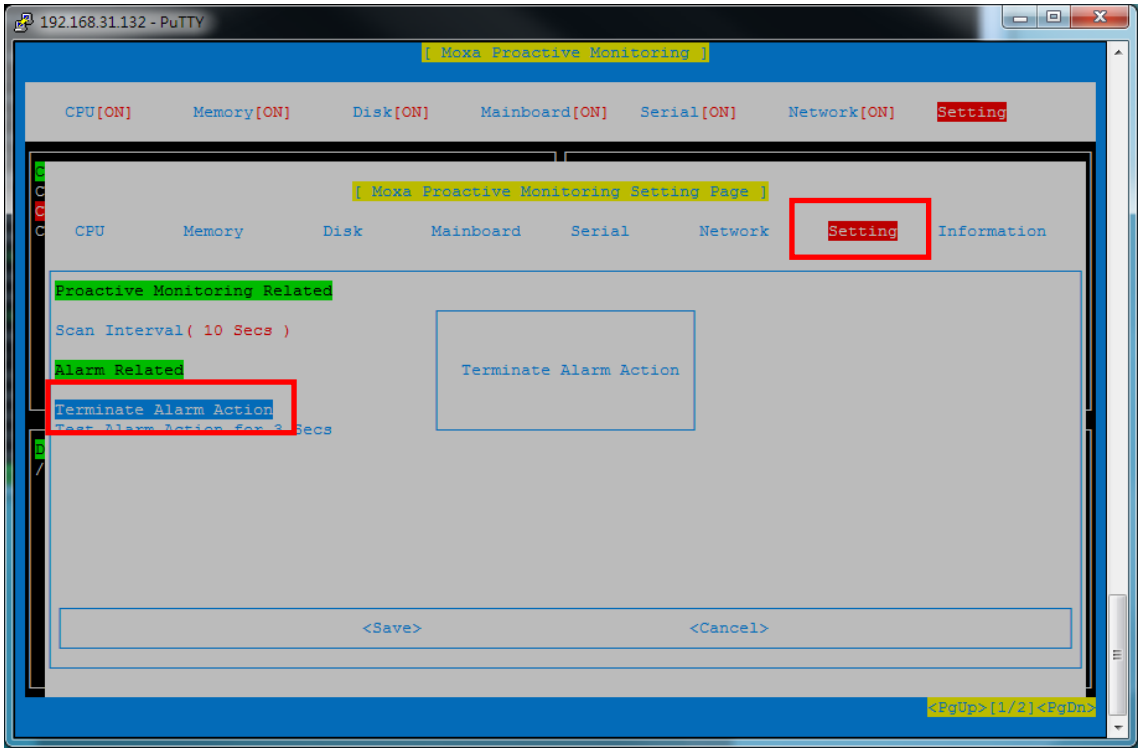
```
root@Moxa:~# vi /sbin/mx_perform_alarm
#!/bin/sh
mx_dio_control -s 1 -n 1
```

NOTE If there is a relay on the device, the script will turn on the relay in the default setting.

Stopping an Alarm Action

1. To stop an alarm action, select setting and then click the terminate alarm action item to stop the alarm.

NOTE If the alarm action is triggered, it will run nonstop until you press the terminate alarm action to stop it, even if the item returns to normal status.



2. Terminating an alarm action will execute the **/sbin/mx_stop_alarm** file. You can edit **/sbin/mx_stop_alarm** to change the stop alarm action.

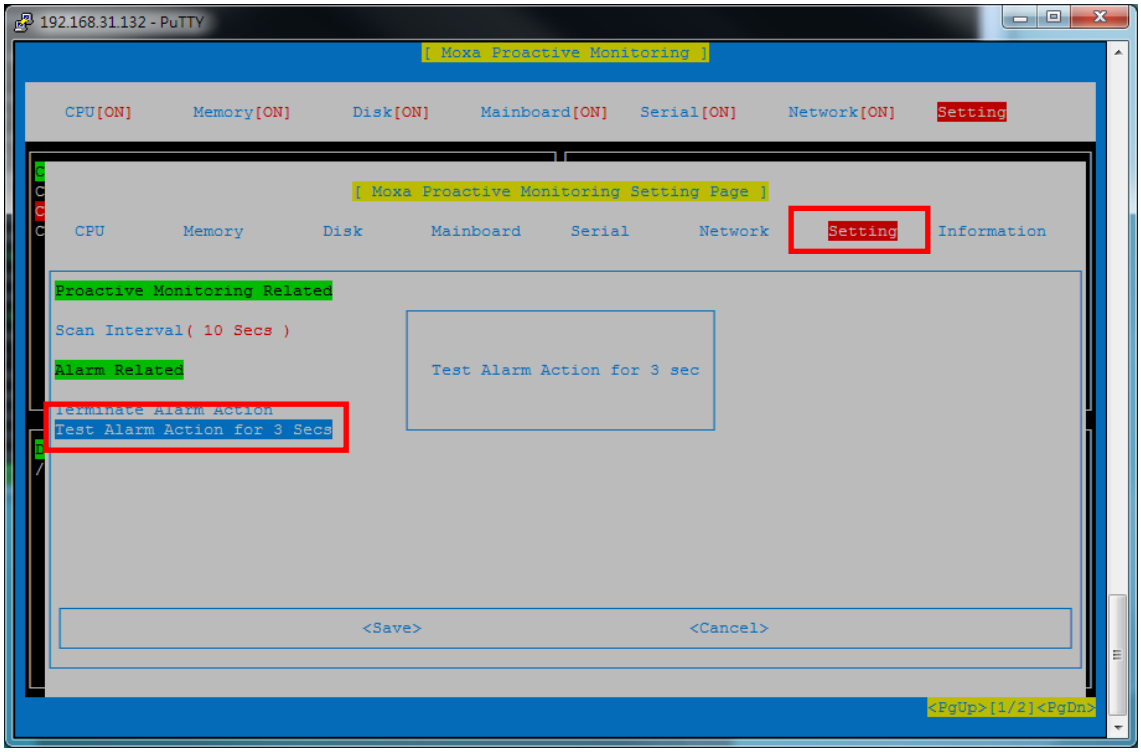
   The script will set the first digital output to low in the default setting.

```
root@Moxa:~# vi mx_stop_alarm
#!/bin/sh
mx_dio_control -s 0 -n 1
```

NOTE If there is a relay on the device, the script will turn off the relay in the default setting.

Testing an Alarm Action

If you want to test the **/sbin/mx_perform_alarm** and **/sbin/mx_stop_alarm** features, you can click the test alarm action to test it. It will execute **/sbin/mx_perform_alarm** for 3 seconds and then execute **/sbin/mx_stop_alarm**.



2

Moxa Proactive Monitoring API

An example of using the Moxa Proactive Monitoring API is stored in the folder **mx_pro_mon**. Refer to the sample code to learn how to apply the API functions described in this chapter.

The following topics are covered in this chapter:

- **API Functions**
- **API Return Value Table**

API Functions

Function	int get_average_cpu_usage(double *value)
Description	Obtain the average CPU utilization of the system.
Input	<value> The variable used to save the average CPU utilization of the system.
Output	<value> The CPU utilization of the system.
Return	Refer to the return value table.

Function	int get_cpu_count(int *value)
Description	Obtain the number of CPU cores in the system.
Input	<value> The variable used to save the number of CPU cores.
Output	<value> The number of CPU cores.
Return	Refer to the return value table.

Function	int get_cpu_usage(int index, double *value)
Description	Obtain the specified CPU core utilization.
Input	<index> CPU number (0, 1, 2, 3 ...) <value> The variable used to save the CPU core utilization.
Output	<value> The specified CPU core utilization.
Return	Refer to the return value table.

Function	int get_mem_total_size(double *value)
Description	Obtain the system's total memory size.
Input	<value> The variable used to save the total memory size.
Output	<value> The system's total memory size.
Return	Refer to the return value table.

Function	int get_mem_usage(int *value)
Description	Obtain the memory utilization of the system.
Input	<value> The variable used to save the memory utilization.
Output	<value> The memory utilization.
Return	Refer to the return value table.

Function	int get_mem_avail_size(double *value)
Description	Obtain the available memory size in the system.
Input	<value> The variable used to save the available memory size.
Output	<value> The available memory size.
Return	Refer to the return value table.

Function	int get_uart_count(int *value)
Description	Obtain the number of UART in the system.
Input	<value> The variable used to save the number of UARTs.
Output	<value> The number of UARTs.
Return	Refer to the return value table.

Function	int get_uart_status(int index, int *value)
Description	Obtain the specified UART status. (0) UART port is free, (1) in use
Input	<index> UART port (0, 1, 2 …) <value> The variable used to save the specified UART status.
Output	<value> The specified UART status. (0) UART port is free, (1) in use
Return	Refer to the return value table.

Function	**int get_eth_count(int *value)**
Description	Obtain the number of Ethernet ports.
Input	<value> The variable used to save the number of Ethernet ports.
Output	<value> The number of Ethernet ports.
Return	Refer to the return value table.

Function	**int get_eth_speed(int index, int *value)**
Description	Obtain the specified Ethernet port speed.
Input	<index> The specified Ethernet port number (0, 1, 2 …) <value> The variable used to save the specified Ethernet port speed.
Output	<value> The specified Ethernet port speed.
Return	Refer to the return value table.

Function	**int get_eth_link(int index, int *value)**
Description	Obtain the specified Ethernet port link status.
Input	<index> The specified Ethernet port number (0, 1, 2 …) <value> The variable used to save specified Ethernet port link status.
Output	<value> The specified Ethernet port link status.
Return	Refer to the return value table.

Function	**int get_eth_usage(int index, int *value)**
Description	Obtain the specified Ethernet port utilization.
Input	<index> The specified Ethernet port number (0, 1, 2 …) <value> The variable used to save the specified Ethernet port utilization.
Output	<value> The specified Ethernet port utilization.
Return	Refer to the return value table.

Function	**int get_eth_link(int index, int *value)**
Description	Obtain the specified Ethernet port link status.
Input	<index> The specified Ethernet port number (0, 1, 2 …) <value> The variable used to save the specified Ethernet port link status.
Output	<value> The specified Ethernet port link status.
Return	Refer to the return value table.

Function	**int get_disk_total_size(char *diskpath, double *value)**
Description	Obtain the specified disk size.
Input	<diskpath> The specified disk location path <value> The variable used to save the specified disk size.
Output	<value> The specified disk size.
Return	Refer to the return value table.

Function	**int get_disk_avail_size(char * diskpath, double *value)**
Description	Obtain the amount of storage space available on the specified disk.
Input	<diskpath> The specified disk location path <value> The variable used to save the amount of storage space available on the specified disk.
Output	<value> The amount of storage space available on the specified disk.
Return	Refer to the return value table.

Function	**int get_disk_usage(char *diskpath, double *value)**
Description	Obtain the specified disk utilization.
Input	<diskpath > The specified disk location path. <value> The variable used to save the specified disk utilization.
Output	<value> The specified disk utilization.
Return	Refer to the return value table.

Function	**int get_device_name(unsigned char *value)**
Description	Obtain the device name of the platform.
Input	<value> The variable used to save the device name of the platform.
Output	<value> The device name of the platform
Return	Refer to the return value table.

Function	**int get_bios_ver(unsigned char *value)**
Description	Obtain the bios version.
Input	<value> The variable used to save the bios version.
Output	<value> The bios version.
Return	Refer to the return value table.

Function	**int get_ser_num(unsigned char *value)**
Description	The get_ser_num function is used to obtain the device serial number.
Input	<value> The variable used to save the device serial number
Output	<value> The device serial number
Return	Refer to the return value table.

Function	**int get_pwr_status(int port, int *value)**
Description	Obtain the power supply status.
Input	<port> The power supply number (0, 1 …) <value> The variable used to save the status of the power supply.
Output	<value> The status of the power supply.
Return	Refer to the return value table.

Function	**int set_relay(int port, int value)**
Description	Set relay status by the input value (0) off (1) on.
Input	<port> relay port (0, 1 …) <value> set status: (0) off, (1) on
Output	None
Return	Refer to the return value table.

Function	**int get_milli_volt(int index, unsigned int *value)**
Description	Obtain the specified device voltage by index.
Input	< index> The Voltage type: (0) CPU, (1) Memory, (2) Mainboard <value> The variable used to save the specified device voltage.
Output	<value> The specified device voltage.
Return	Refer to the return value table.

Function	**int get_temperature(int index, unsigned int *value)**
Description	Obtain the specified device temperature.
Input	< index> The Voltage type: (0) CPU, (1) System <value> The variable used to save the specified device temperature.
Output	<value> The specified device temperature.
Return	Refer to the return value table.

API Return Value Table

value	Meaning
0	The operation has completed successfully.
-100	The parameter is invalid.
-101	The command **popen** failed to run. Unable to open the process.
-102	The command pclose failed to run. Unable to close the process.
-103	The system cannot open the device node.
-104	The system cannot close the device node.
-105	The IOCTL call made by the application program is not correct.
-106	Could not load the ini file.
-107	The source command was not found in the sh shell.
-108	The system cannot find the ini key.
-200	The system cannot get the device name.
-201	The system cannot get the sensor value.
-202	The system cannot get the BIOS version.
-203	The system cannot get the serial number.
-204	This system does not support relay.
-205	This system does not support power indicator.
-300	Failed to get Ethernet port status.