Moxa UC-8410A Proactive Monitoring User's Manual

Edition 1.0, May 2016

www.moxa.com/product

Moxa UC-8410A 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 1-1
Installing Moxa Proactive Monitoring 1-2
Installation Steps 1-2
Starting or Stopping the Moxa Proactive Monitoring Daemon 1-2
Monitoring System Status 1-3
How to display the Moxa Proactive Monitoring UI 1-3
How to Use the Moxa Proactive Monitoring UI 1-3
Customizing the Monitoring Views 1-5
Setting the System Alarm 1-7
Changing Alarm Settings 1-8
Performing an Alarm Action 1-9
Stopping an Alarm Action 1-10
Testing an Alarm Action 1-11
2. Moxa Proactive Monitoring API 2-1
API Functions 2-2
API Return Value Table 2-5

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 the Monitoring Views
- **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. Download the **mxpromon_1.0.1_armhf.deb** utility from the Moxa website on to the target machine.
2. Type the following command to install Moxa Proactive Monitoring utility:

```
root@Moxa:~# dpkg -i mxpromon_1.0.1_armhf.deb
```

Starting or Stopping the Moxa Proactive Monitoring Daemon

The Moxa Proactive Monitoring daemon ***pro_mond*** runs in the background when the system boots up and continuously monitors the status of the system. The daemon can log alarm messages and trigger actions based on the alarms, if these features are activated.

Starting/Stopping the Daemon

Type the following command to start the **pro_mond** daemon:

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

Type the following command to stop the **pro_mond** daemon:

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

Restarting the Daemon

Type the following command to restart the **pro_mond** daemon:

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

How to Prevent the Daemon from Starting When the System Boots Up

Use the following command to turn off the **pro_mond** daemon and prevent it from starting up when the system boots up:

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

How to Configure the Daemon to Start at System Boot Up

Use the following command to turn the **pro_mond** daemon on if you want it to start when the system boots up:

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

Monitoring System Status

NOTE We suggest using the full screen display option to be able to optimally view the Moxa Proactive Monitoring user interface. You can only run one instance of the Moxa Proactive Monitoring application at a time.

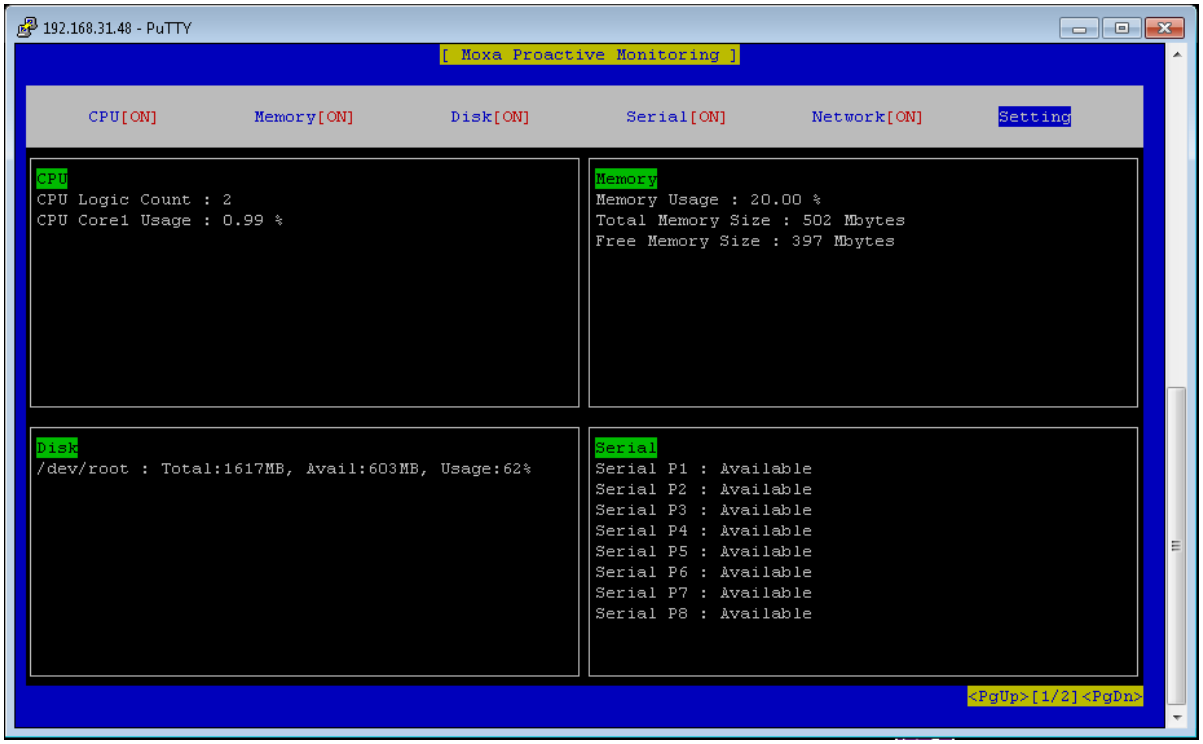
How to display the Moxa Proactive Monitoring UI

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

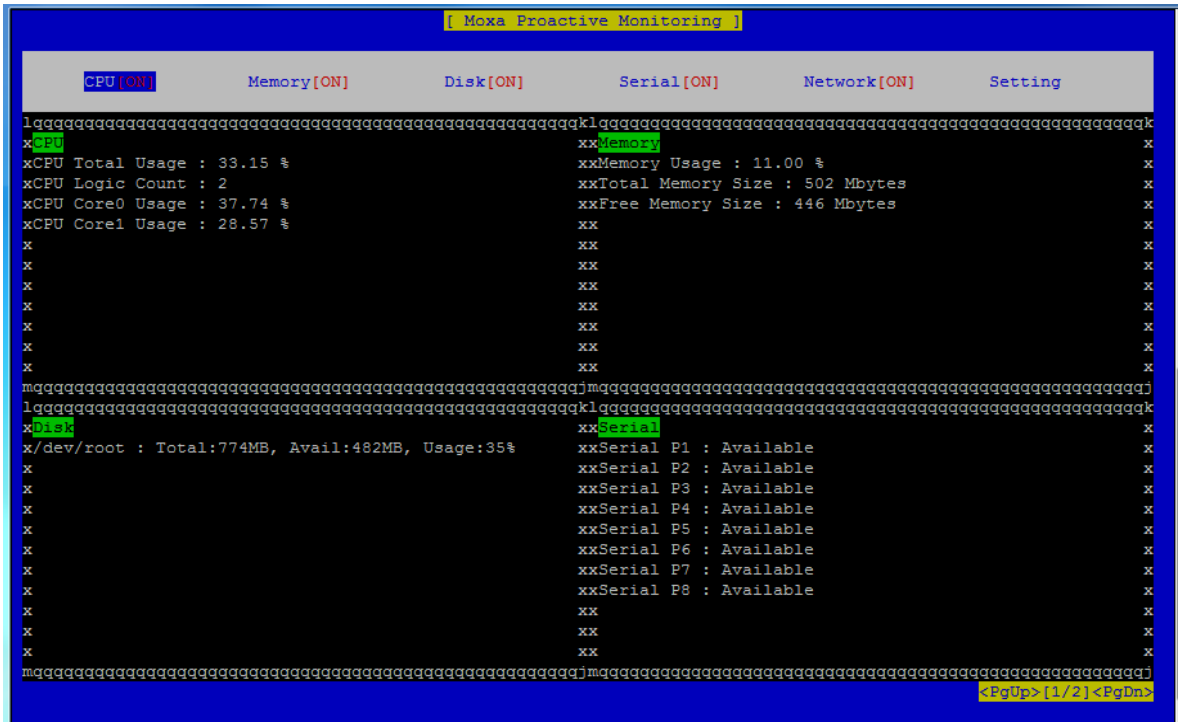
```
root@Moxa:~ # pro_mon
```

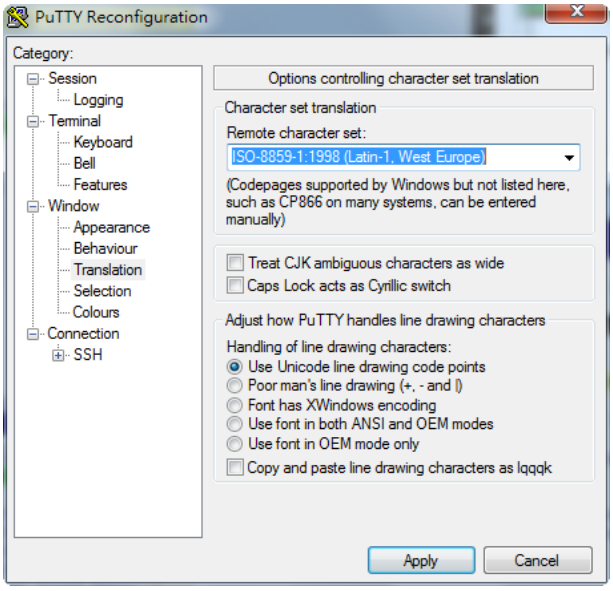
How to Use the Moxa Proactive Monitoring UI

The UI displays four system status menus at the same time. Use **<PgUp>** or **<PgDn>** keys to scroll up or down to the next page to display other menus.



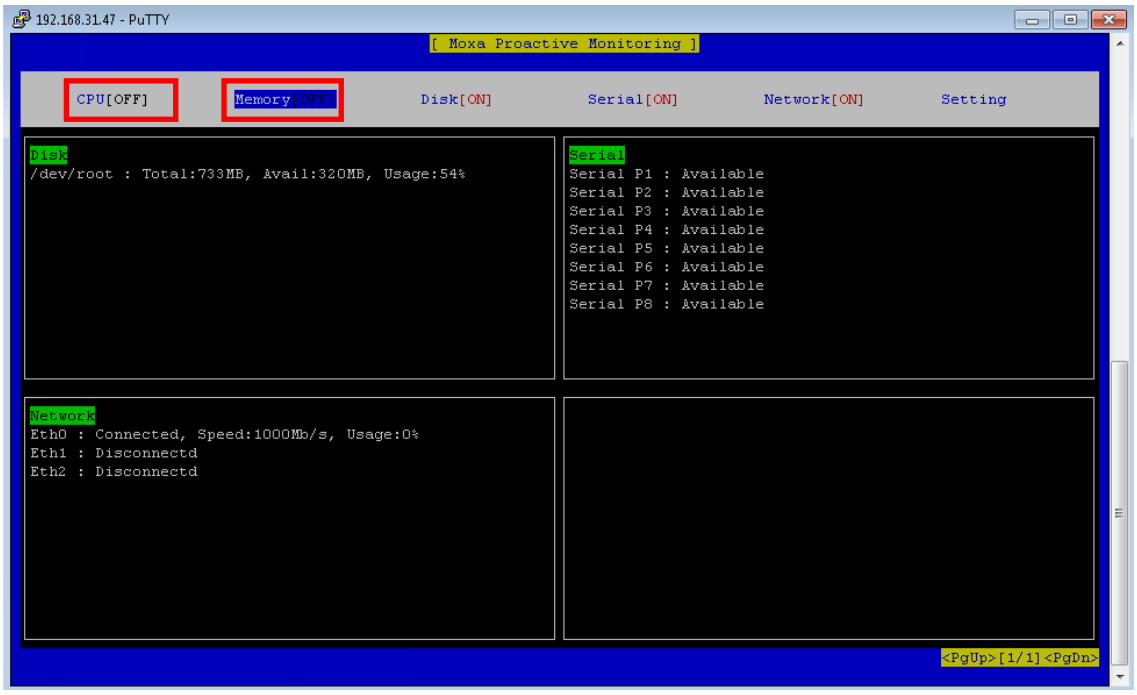
NOTE If you see garbage characters on the UI screen as shown in the following screenshot, change the Remote character set setting in your computer from UTF-8 to a different encoding, for example, ISO-8859-1, and then restart the **pro_mon.**daemon





Customizing the Monitoring Views

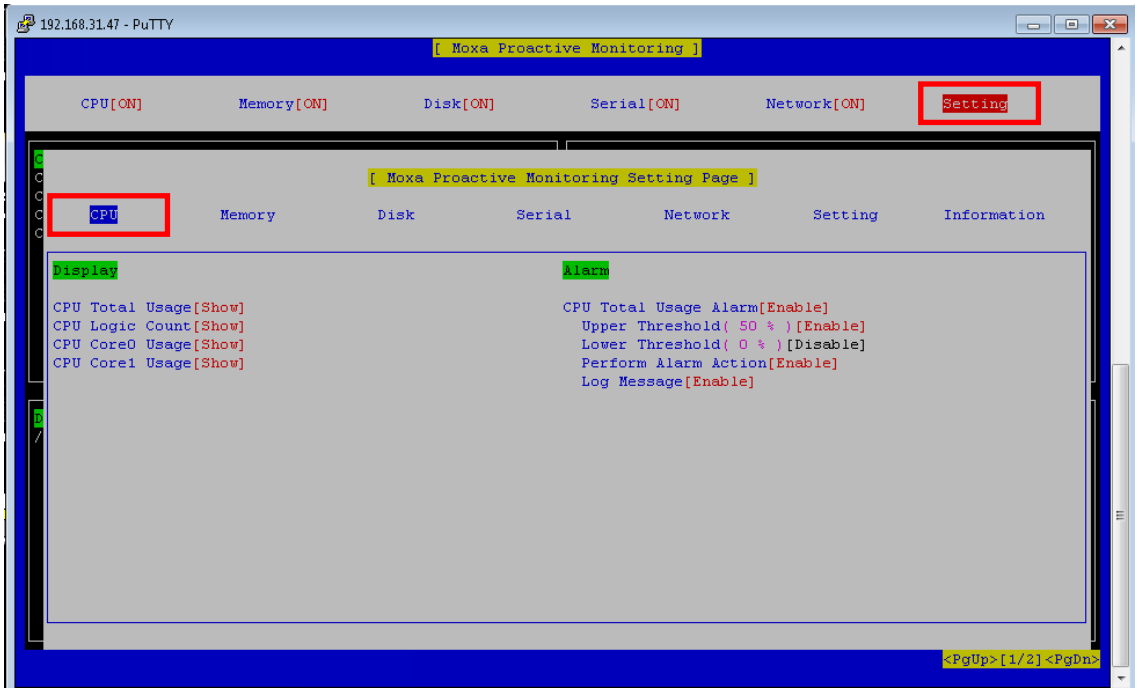
You can select the different items to show in the Proactive Monitoring UI by turning the status of the items to "On" and removing the items that you do not want to display by turning their status value to "Off". For example, if you do not want to show the CPU and memory status, you can turn off these features by first selecting these features and pressing the space bar or the **Enter** button on the computer keyboard.



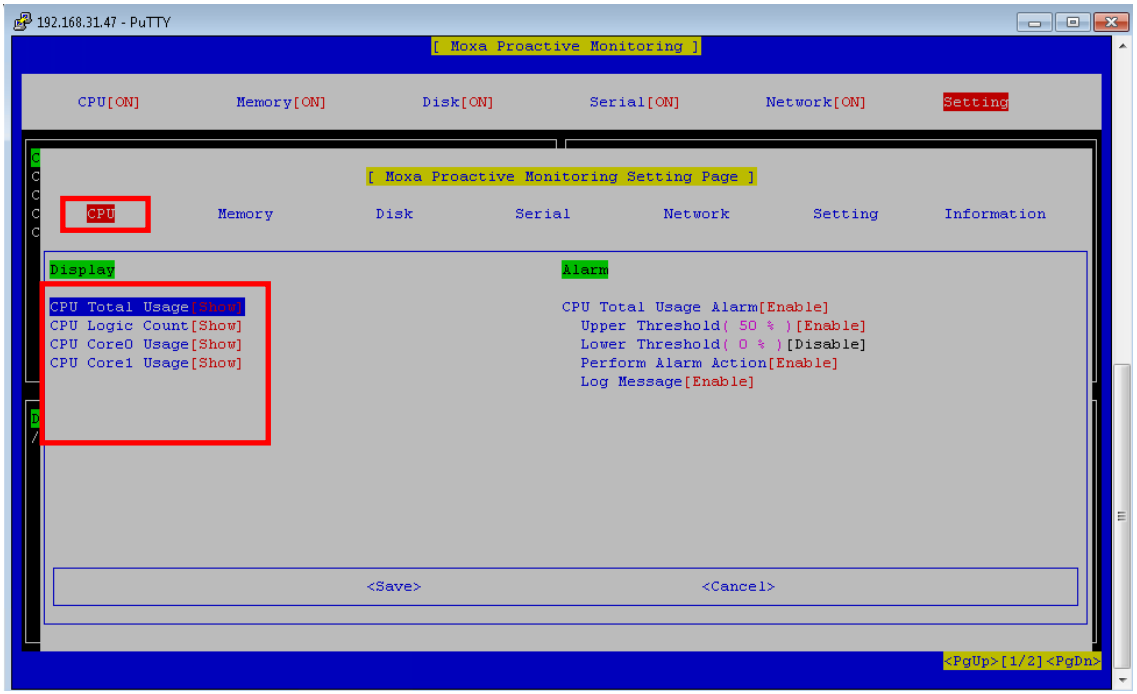
You can further customize the items that you want to display in each monitoring status category.

For example, if you don't want to see the CPU usage of each core, you can turn it off. To achieve this, do the following:

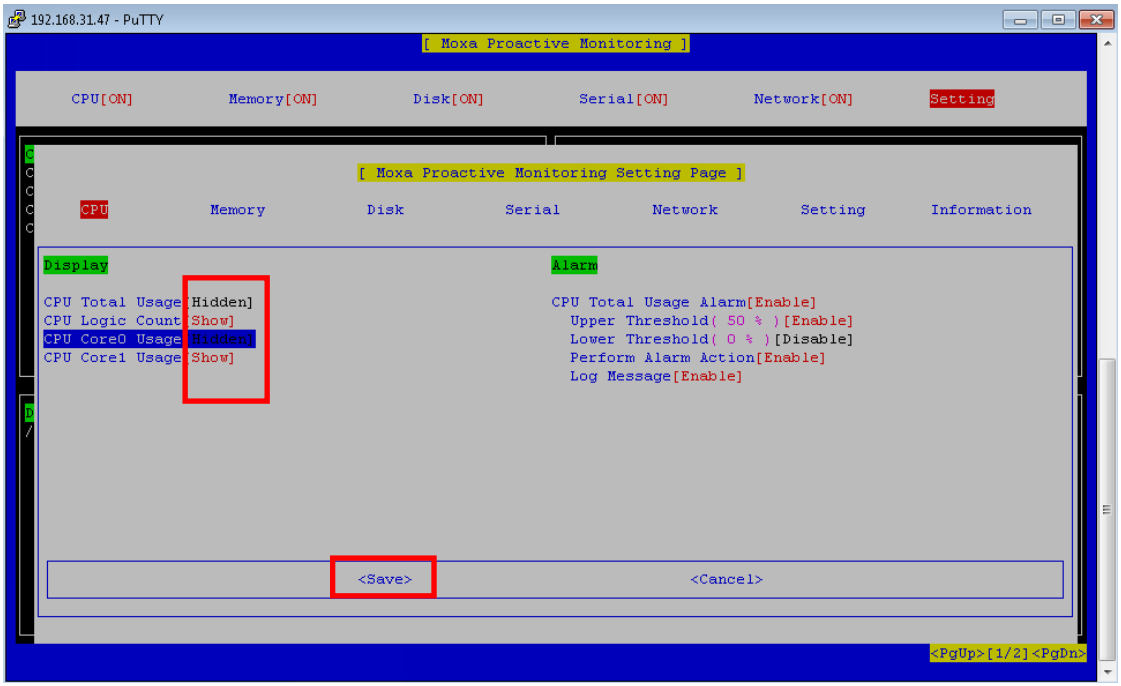
1. Select **Setting** and then press the space bar or the **Enter** button on your keyboard to go to the settings page**.**



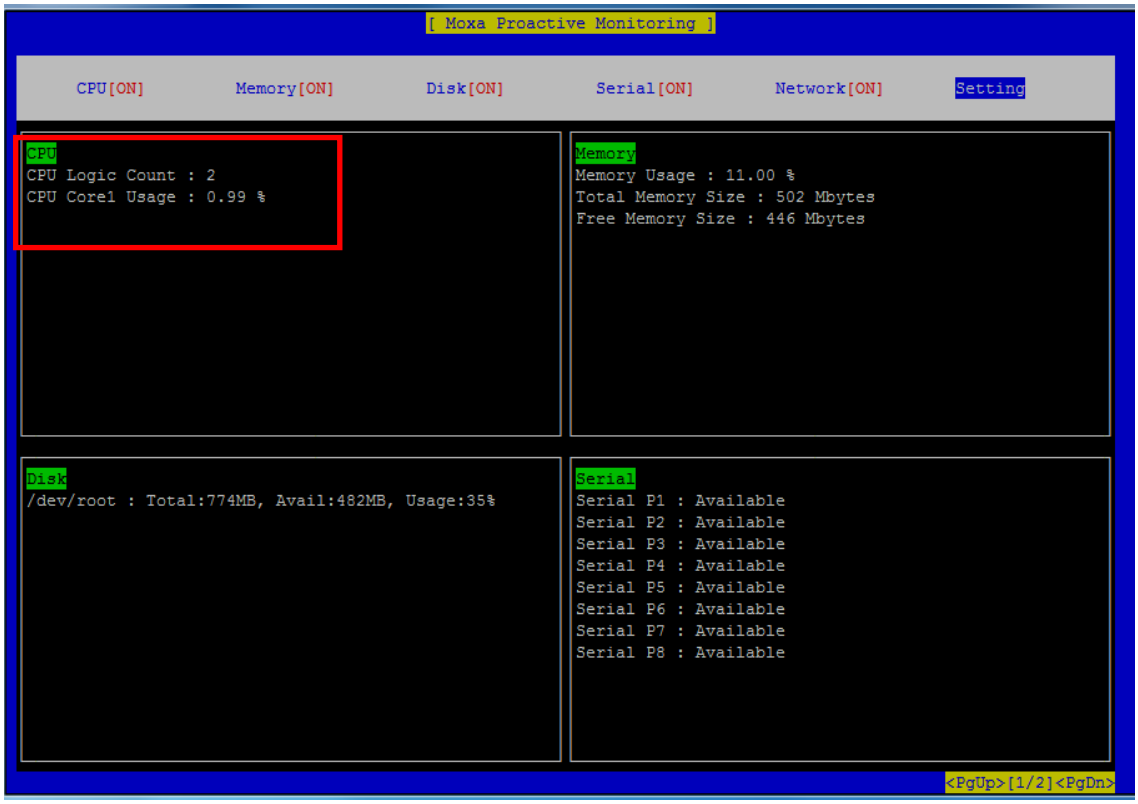
2. Choose the CPU items that you want to display by setting the display option to **Show**, and the ones that you want hidden by changing their display option to **Hidden**.



For example, as shown below, you can hide the **CPU Usage** parameter**,** and then press **Tab** to switch to **<save>** and press **Enter** to save the setting.



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

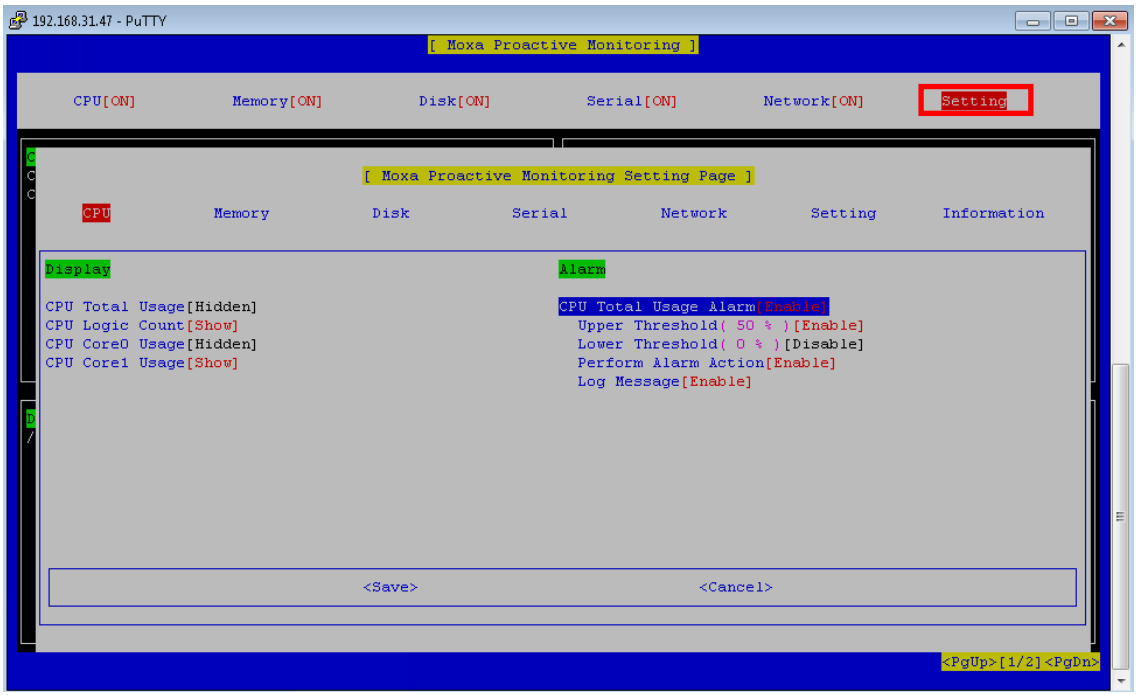


Setting the System Alarm

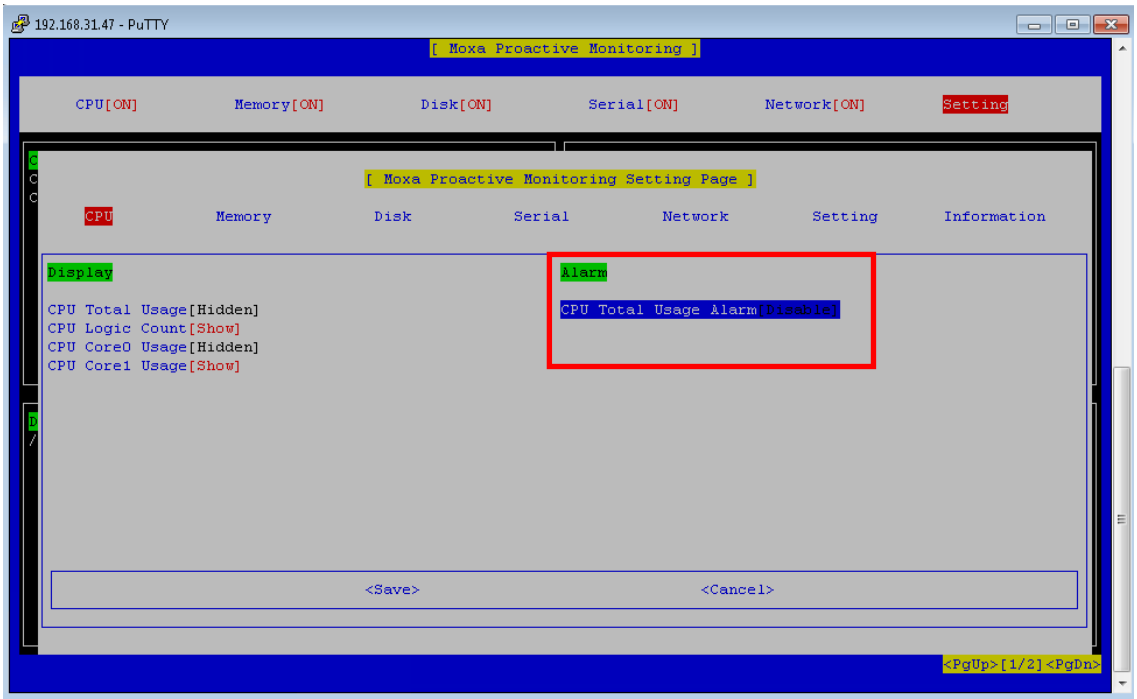
The Proactive Monitoring utility lets you configure system alarms based on the threshold values that you set for system parameters. You can also define and perform actions based on these alarm conditions. The steps to configure and use alarms and trigger alarm actions are explained in the following sections:

Changing Alarm Settings

1. In the **Moxa Proactive Monitoring** utility, select the **Setting** tab.



2. Select the CPU item under **Alarm** as shown in the screenshot and configure the alarm settings to monitor the system status and to perform an alarm action when the system status exceeds the threshold values set.

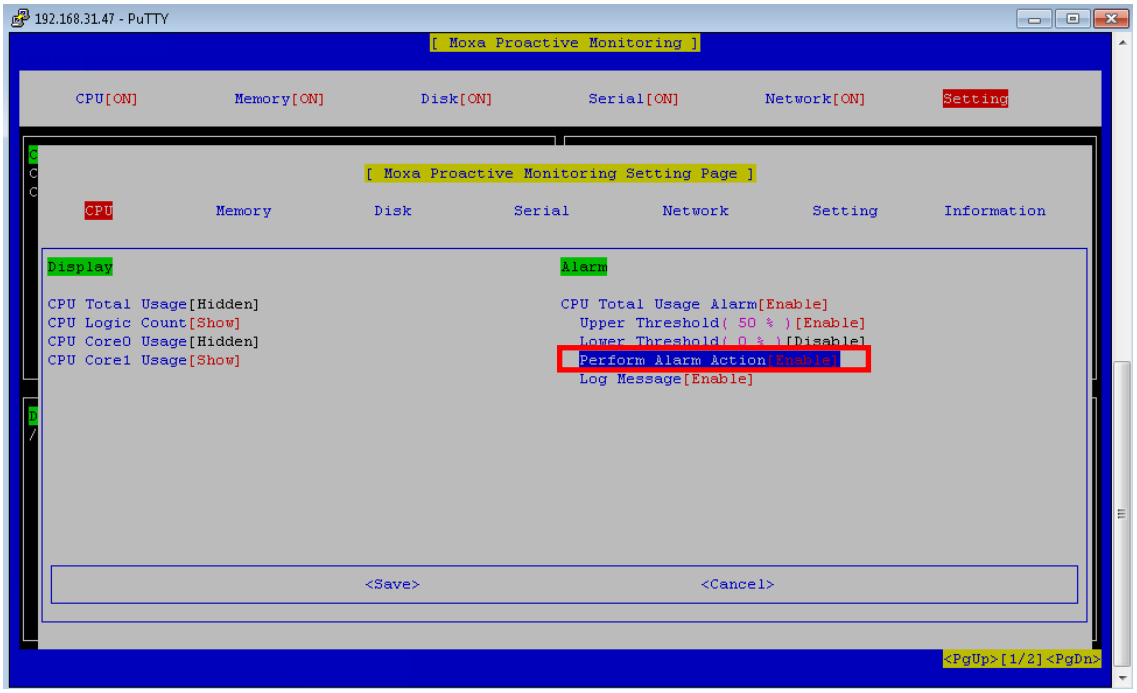


3. To turn on the alarm setting, press the space bar or the **Enter** key on your keyboard.

NOTE When you enable the alarm setting, the log message option will also get enabled by default. You can disable the log message option if you do not need it.

Performing an Alarm Action

1. If you want to trigger an action when the alarm occurs, select **Perform Alarm Action** and press the space bar or the **Enter** button to enable it, and then use the Tab key to move to the <save> option and press **Enter** to save the setting.



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

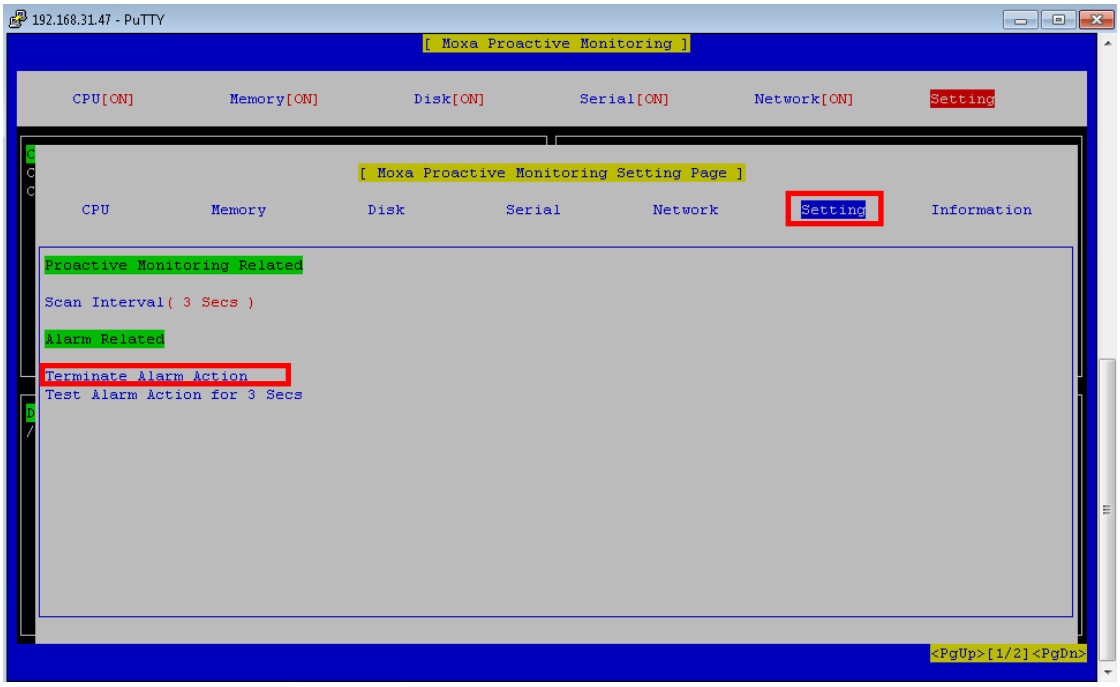
```
root@Moxa:~# vi /sbin/mx_perform_alarm
#!/bin/sh
mx_dio_control -s 1 -n 1
```

NOTE When you trigger an alarm action, the script in the **/sbin/mx_perform_alarm** file will set the first digital output to high by default. If there is a relay on the device, the script will turn on the relay by default.

Stopping an Alarm Action

1. To stop an alarm action, select **Setting** and then select the **Terminate Alarm Action** option under the **Alarm Related** section to stop the alarm.

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



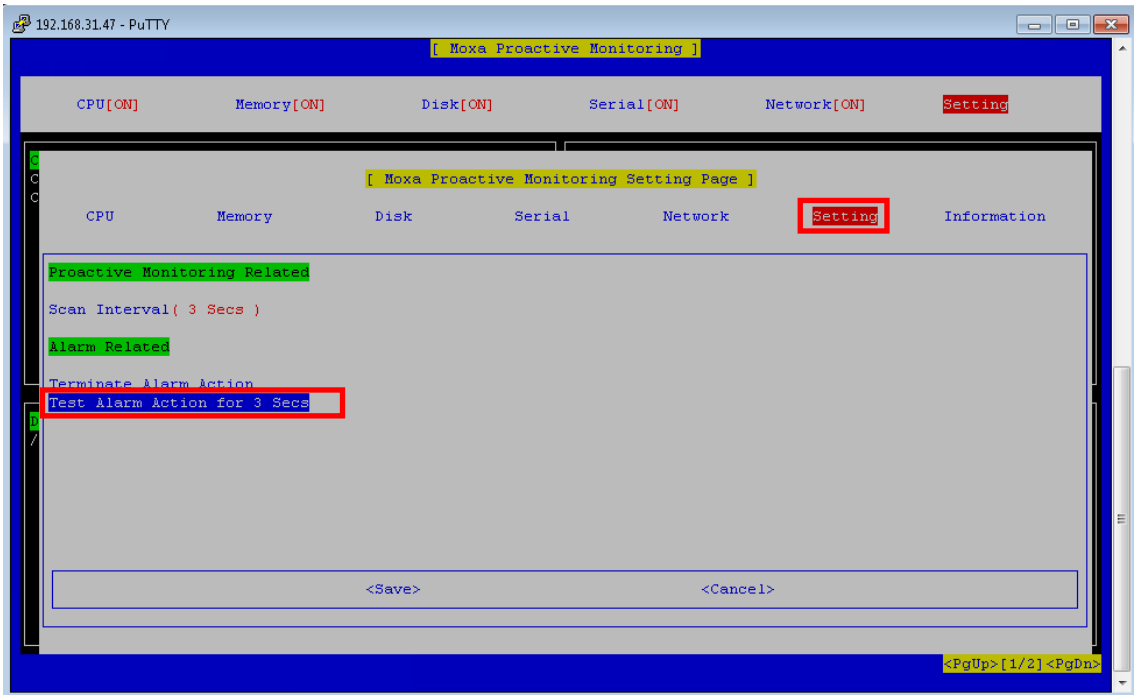
2. Terminating an alarm action will execute the **/sbin/mx_stop_alarm** file. You can edit the file to change the Terminate Alarm Action.

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

NOTE When you terminate an alarm action, the script in the **/sbin/mx_stop_alarm** file will set the first digital output to low by default. If there is a relay on the device, the script will turn off the relay by default.

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 for 3 Secs** option under **Setting**. The test alarm action will run the **/sbin/mx_perform_alarm** script for 3 seconds and then run the **/sbin/mx_stop_alarm**, which will stop the alarm action.



2

Moxa Proactive Monitoring API

An example for using the Moxa Proactive Monitoring API is stored in the **mx_pro_mon** folder. 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	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	<value>: Refer to the **API Return Value Table** for details.

Function	int get_cpu_count(int *value)
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	int get_cpu_usage(int index, double *value)
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	int get_mem_total_size(double *value)
Description	The total memory size of the system.
Input	<value> The variable used to save the total memory size.
Output	<value> The system's total memory size.
Return	<value>: Refer to the **API Return Value Table** for details.

Function	int get_mem_usage(int *value)
Description	The memory utilization of the system.
Input	<value> The variable used to save the memory utilization.
Output	<value> The memory utilization.
Return	<value>: Refer to the **API Return Value Table** for details.

Function	int get_mem_avail_size(double *value)
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	int get_uart_count(int *value)
Description	The number of UART (serial) interfaces in the system.
Input	<value> The variable used to save the number of UART (serial) interfaces.
Output	<value> The number of UART (serial) interfaces.
Return	<value>: Refer to the **API Return Value Table** for details.

Function	int get_uart_status(int index, int *value)
Description	The specified UART status: (0) UART port is free; (1)UART port is 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) UART port is in use
Return	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_eth_count(int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_eth_speed(int index, int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_eth_link(int index, int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_eth_usage(int index, int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_eth_link(int index, int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_disk_total_size(char *diskpath, double *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_disk_avail_size(char * diskpath, double *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_disk_usage(char *diskpath, double *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_device_name(unsigned char *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_bios_ver(unsigned char *value)**
Description	The bios version.
Input	<value> The variable used to save the bios version.
Output	<value> The bios version.
Return	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_ser_num(unsigned char *value)**
Description	The device serial number.
Input	<value> The variable used to save the device serial number
Output	<value> The device serial number
Return	<value>: Refer to the **API Return Value Table** for details.

Function	**int get_pwr_status(int port, int *value)**
Description	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	<value>: Refer to the **API Return Value Table** for details.

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.