Moxa Proactive Monitoring Windows Software User's Manual

Edition 1.0, November 2015

www.moxa.com/product

Moxa Proactive Monitoring Windows Software User's Manual

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

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

Installation and Usage

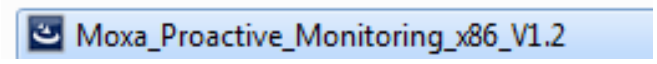
The following topics are covered in this chapter:

- **Installing Moxa Proactive Monitoring**
- **Monitoring System Status**
- **Customizing Your Own Monitoring Items**
- **Setting System Alerts**
- **Clearing Alert Output**
- **Changing the Device Configuration**

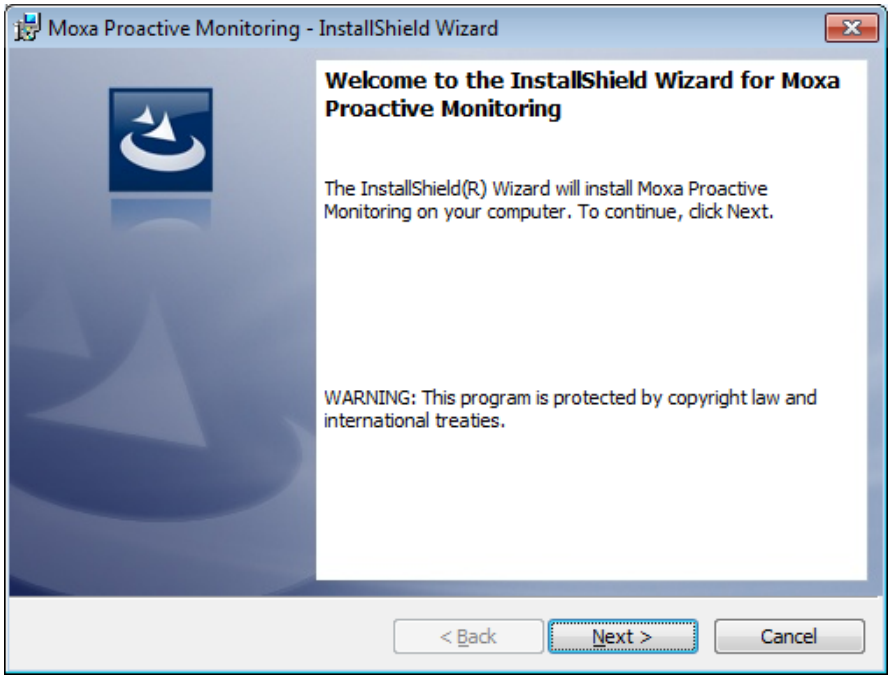
Installing Moxa Proactive Monitoring

In this section, we describe how to install the Moxa Proactive Monitoring software.

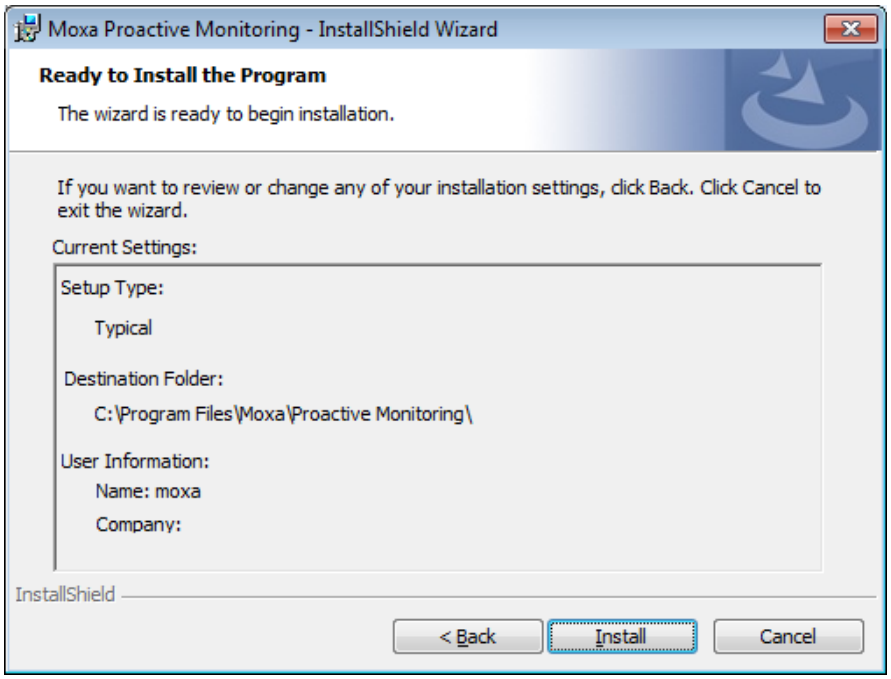
1. Run the exe file provided in the setup package.



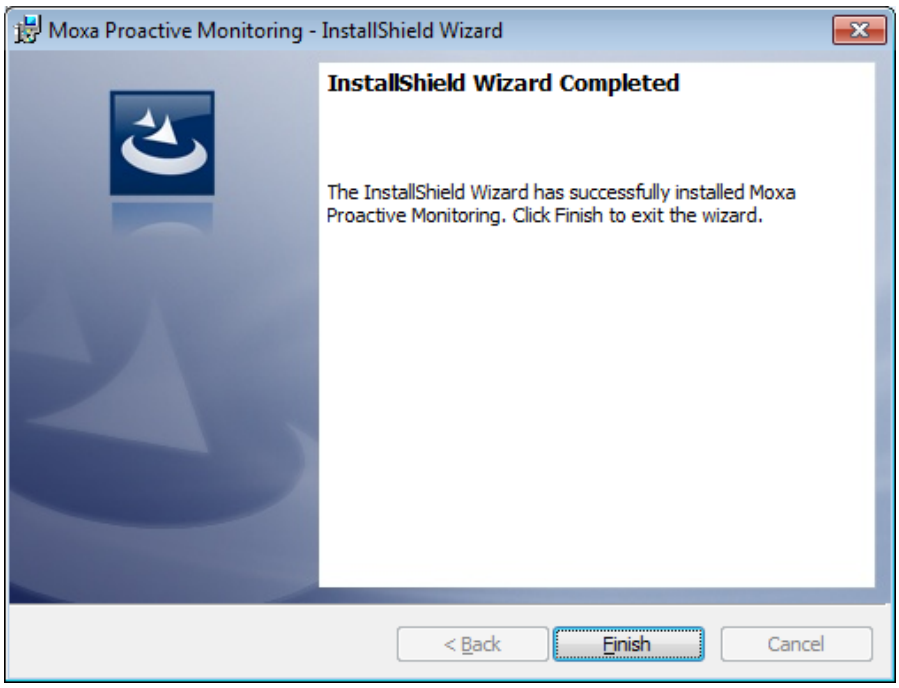
2. Click **Next**.



3. Click **Install**.



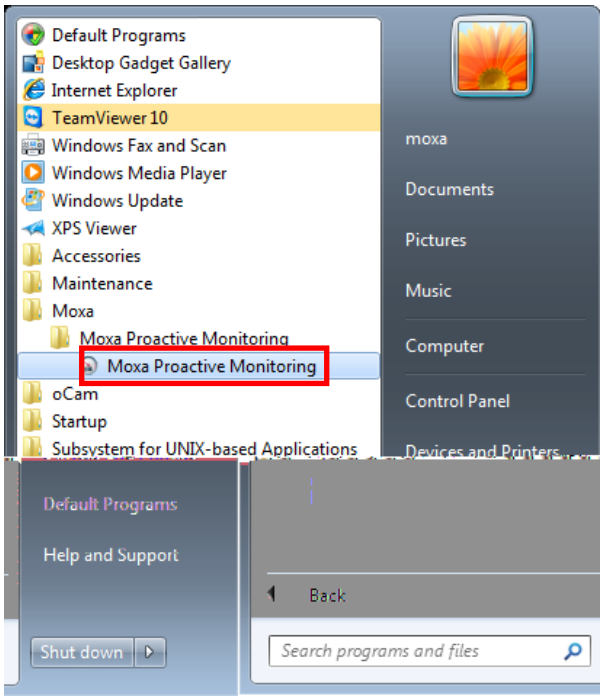
4. Click **Finish**.



Monitoring System Status

Take the following steps to monitor system status:

1. Execute **Moxa Proactive Monitoring** from **All Programs** → **Moxa** → **Moxa Proactive Monitoring**.



2. The system status can be determined by viewing the dashboard.

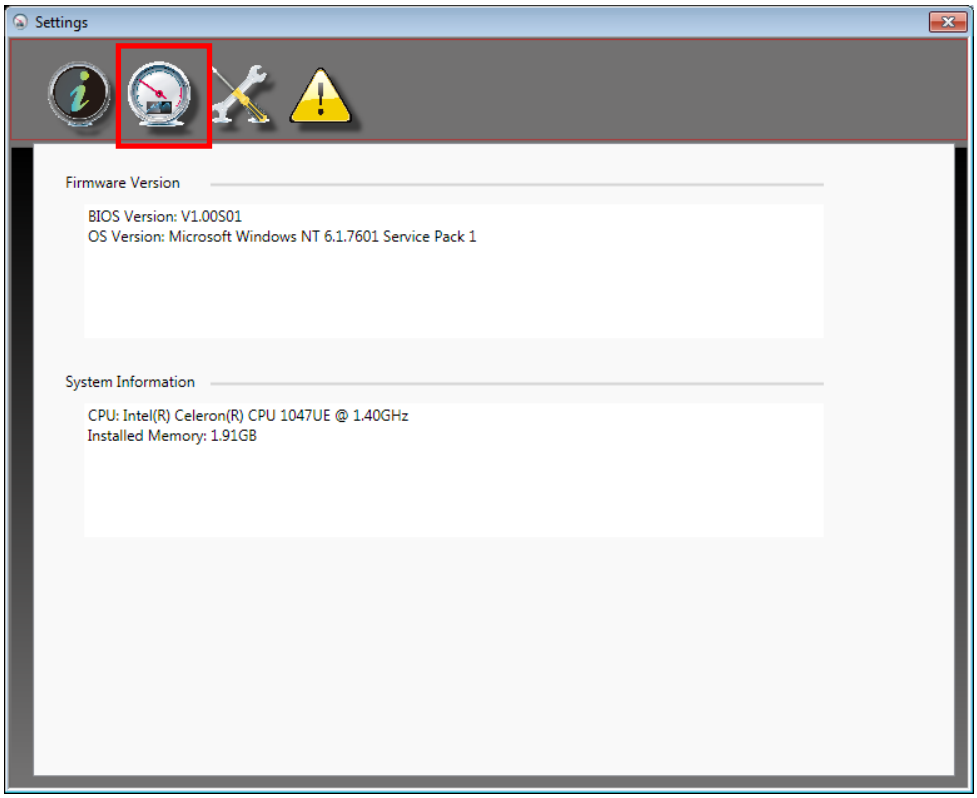
3. The dashboard displays four kinds of system status at the same time. To display other system status items, click the previous button on the left, or the next button on the right, to change which items are displayed.

Customizing Your Own Monitoring Items

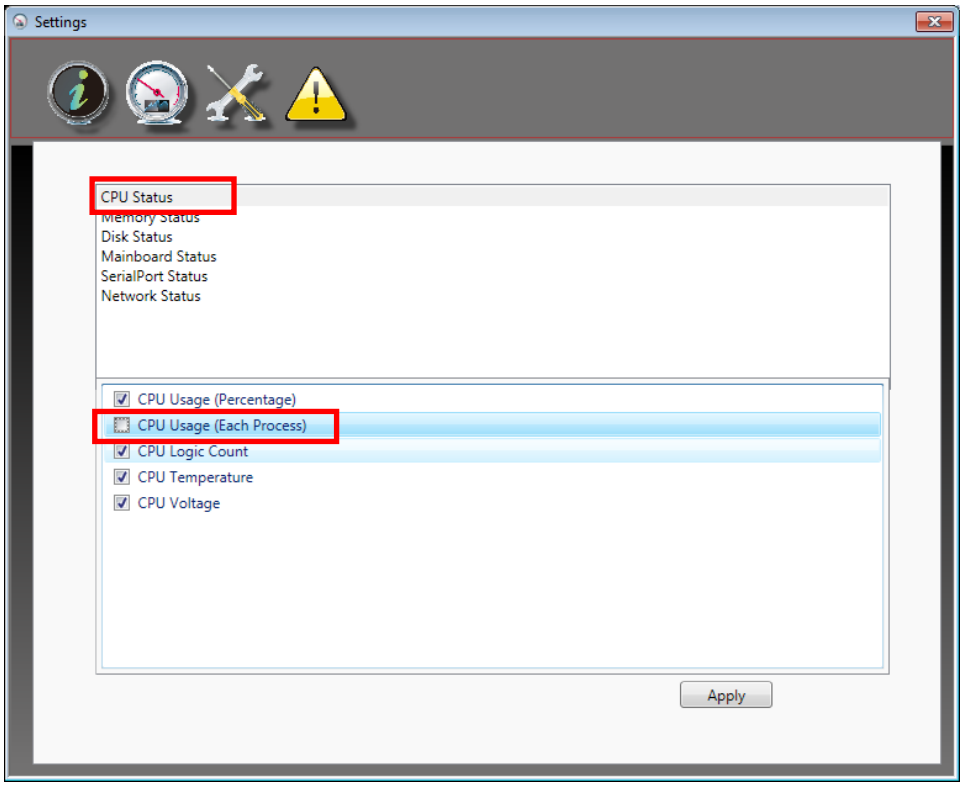
1. You can select your own system status to display on the dashboard by turning on or off the monitor status. For example, if you don't want to monitor the CPU and memory status, you can turn off these features by clicking the buttons under the CPU and Memory figures. The dashboard will be updated based on the selection.

2. Furthermore, you can customize which items are displayed. For example, if you don't want to show the CPU usage of each core, you can turn off that item by clicking the **Settings** button.

3. When the settings page opens, select the second icon to change to the member item selection page.



4. Select **CPU Status** in the top section of the window and then deselect **CPU Usage (Each Process)** in the bottom section of the window.



5. The CPU status shown on the dashboard will be updated based on your selection.

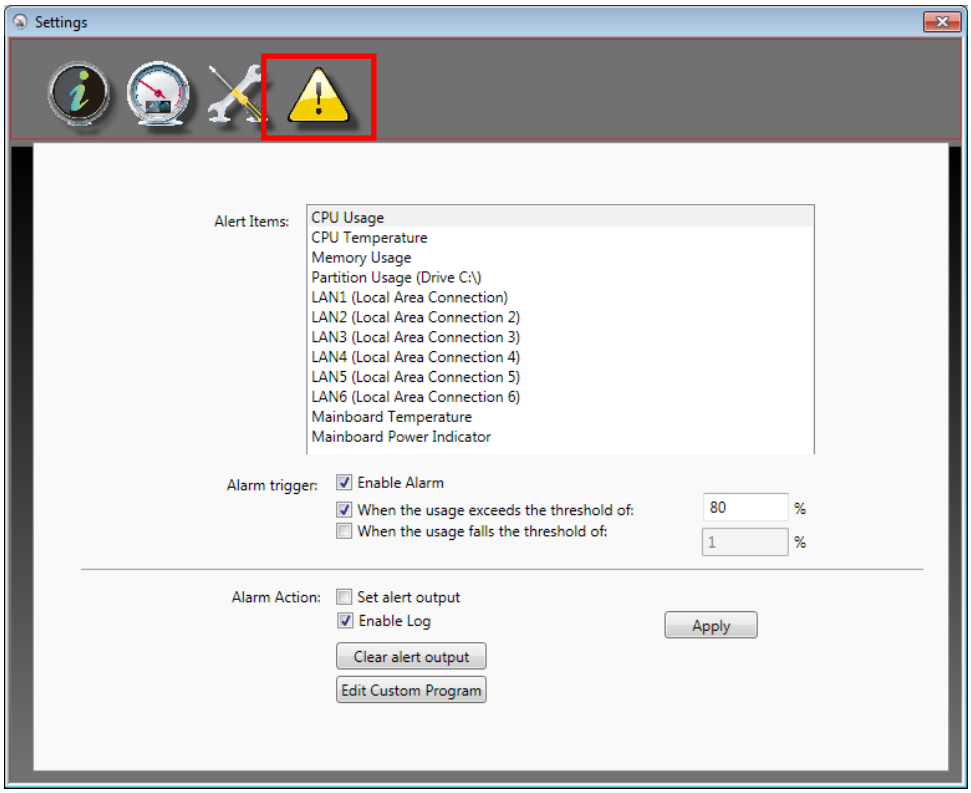
Setting System Alerts

Take the following steps to set system alerts:

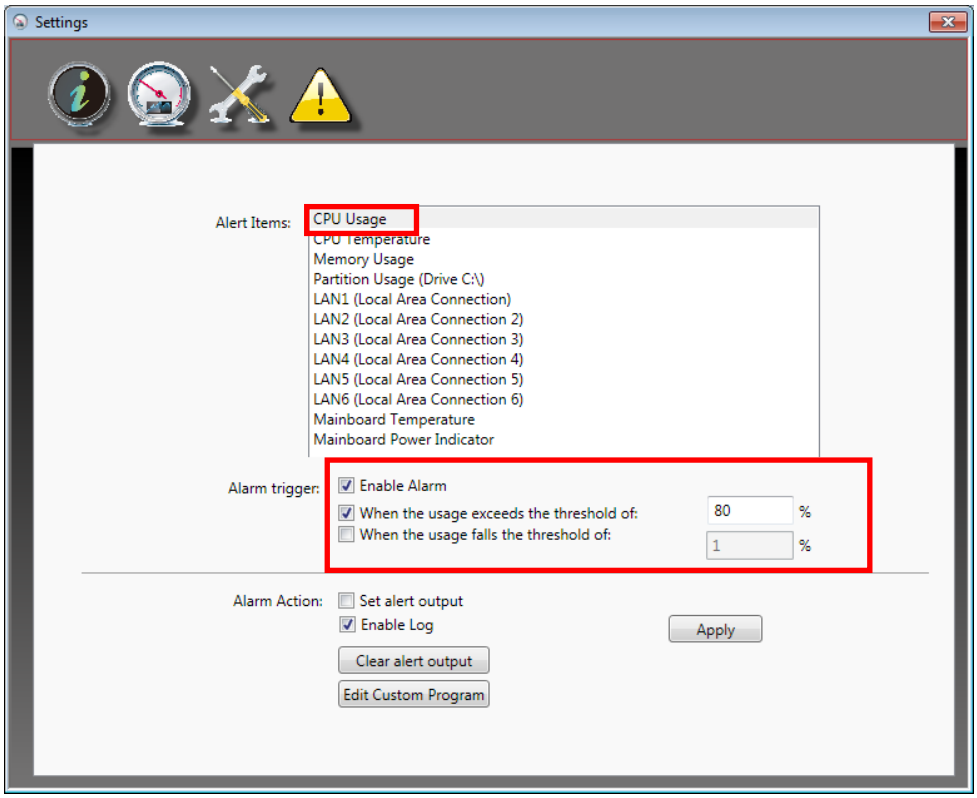
1. Click **Settings**.

2. Click the mouse button to change to the alert setting pages; you can configure the alert settings here to monitor system status and send alarms to relay output when the system status is over the threshold value of the alert settings.

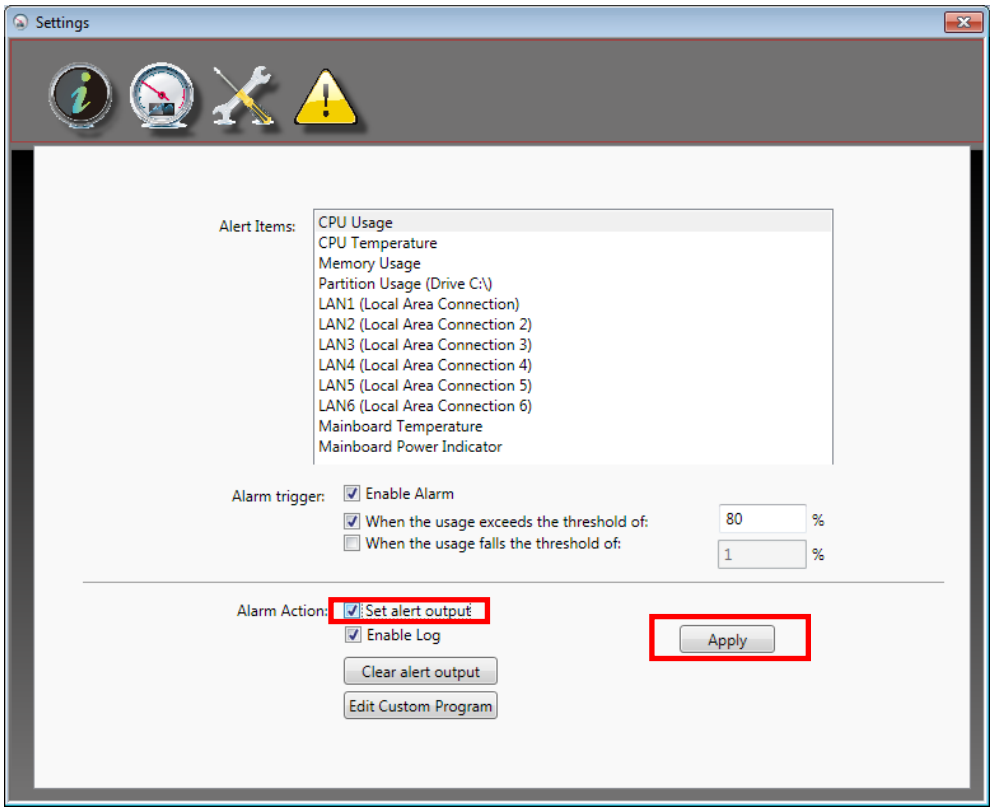
NOTE Only some Moxa computers are equipped with a relay output.



3. For example, suppose alert settings for CPU usage is enabled and the threshold value of CPU usage is 80%. In this case, when CPU usage is over 80%, the icon on the dashboard will change to red, and the alert will be logged in the log file.



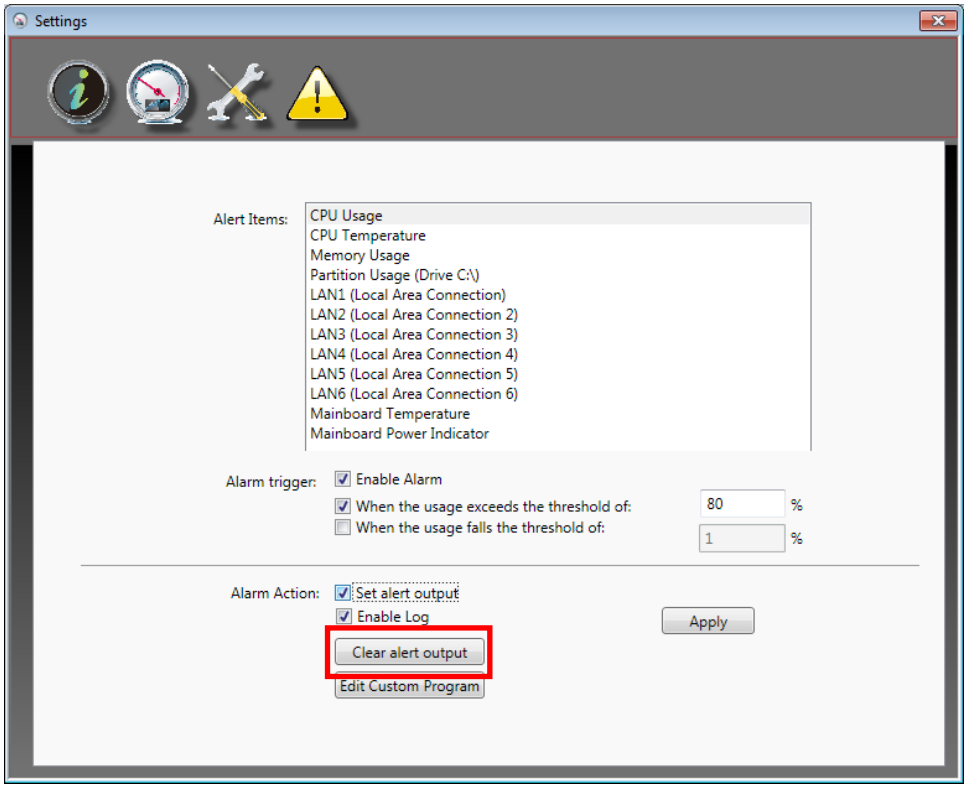
4. If you want to set the relay output when the alert occurs, select the **Set alert output** option and then click **Apply**.



Clearing Alert Output

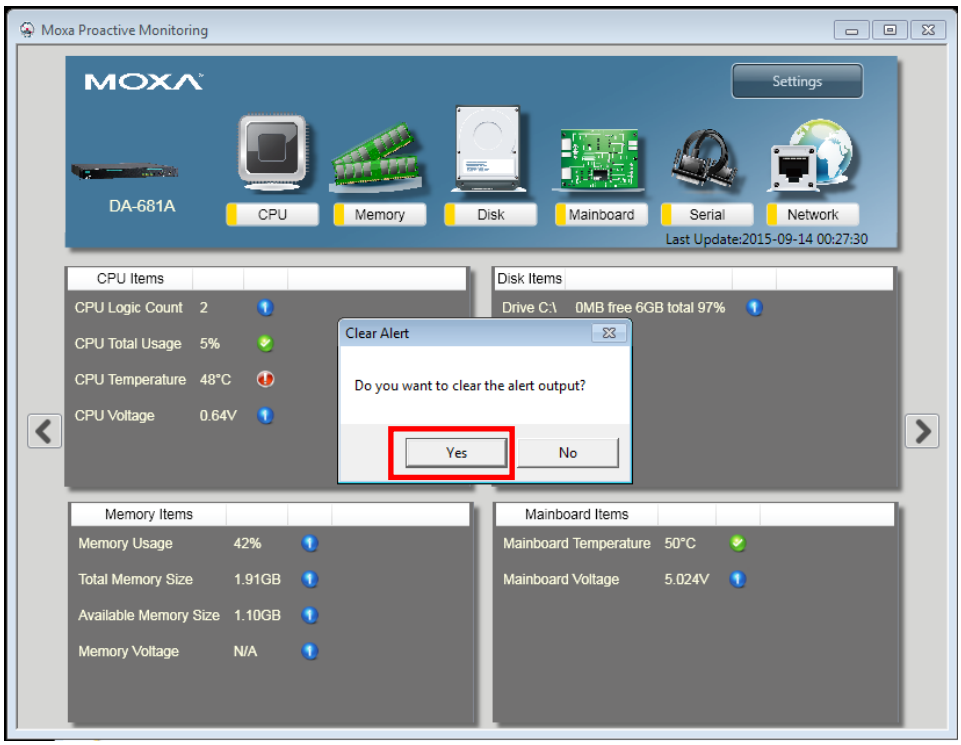
1. When the alert occurs, the relay output signal will be kept until it is cleared. There are two ways to clear the alert output:

   (a) Click the **Clear alert output** button.



   (b) Double-click the alert item on the dashboard.

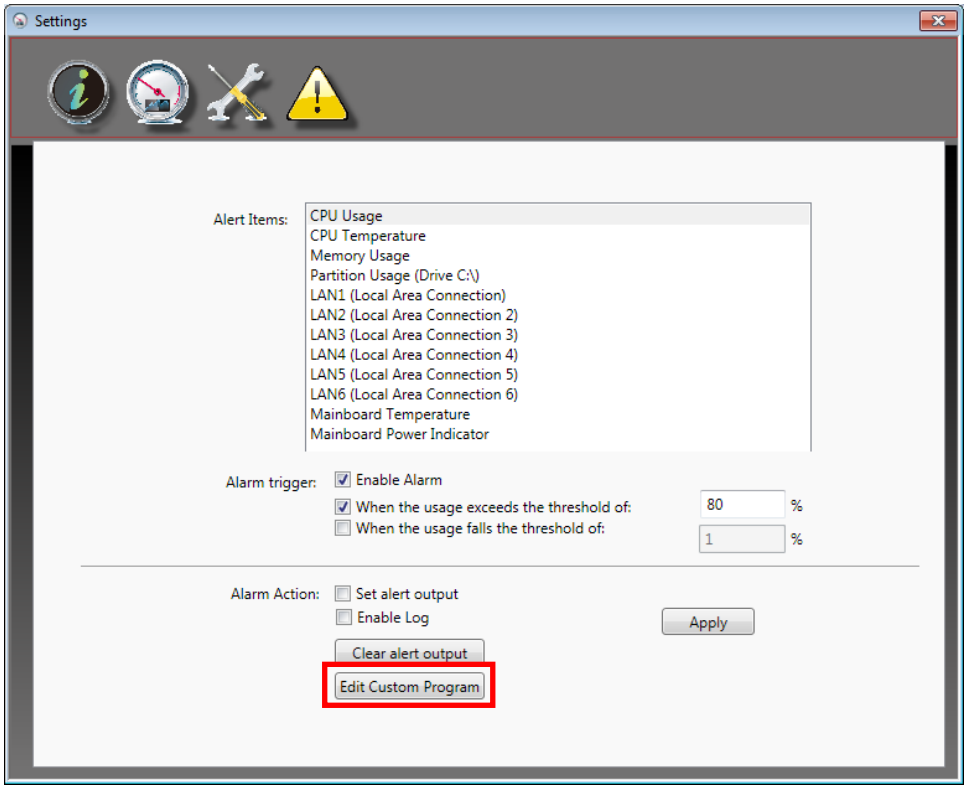
And then select **Yes** to clear the relay output.



Changing the Device Configuration

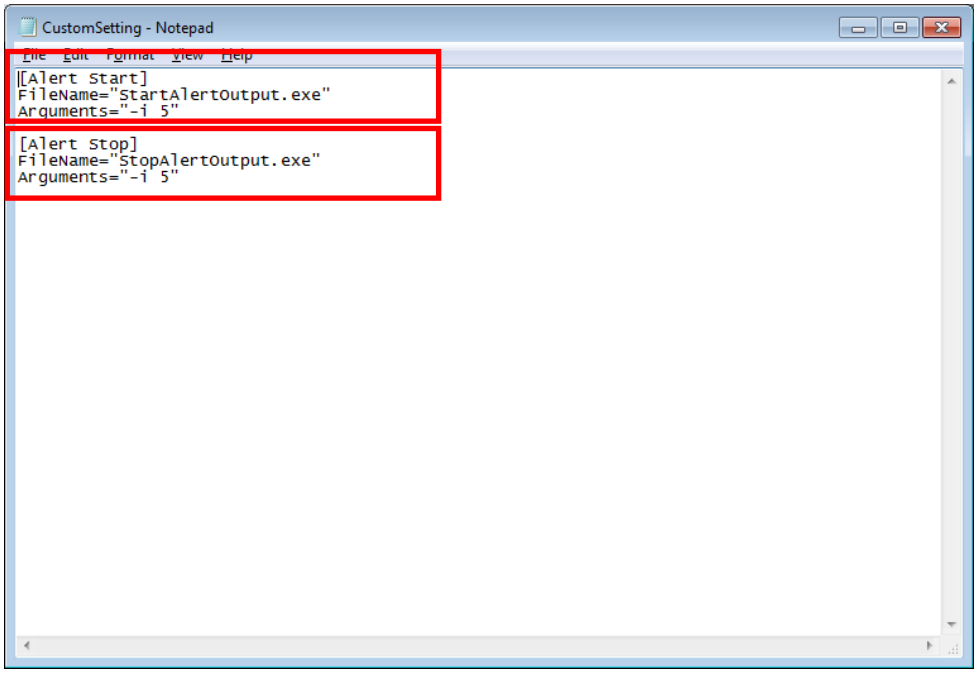
The alert output can be assigned to your programs when the alert is enabled or disabled. Take the following steps to assign the alert output:

1. Click the **Edit Custom Program** button.

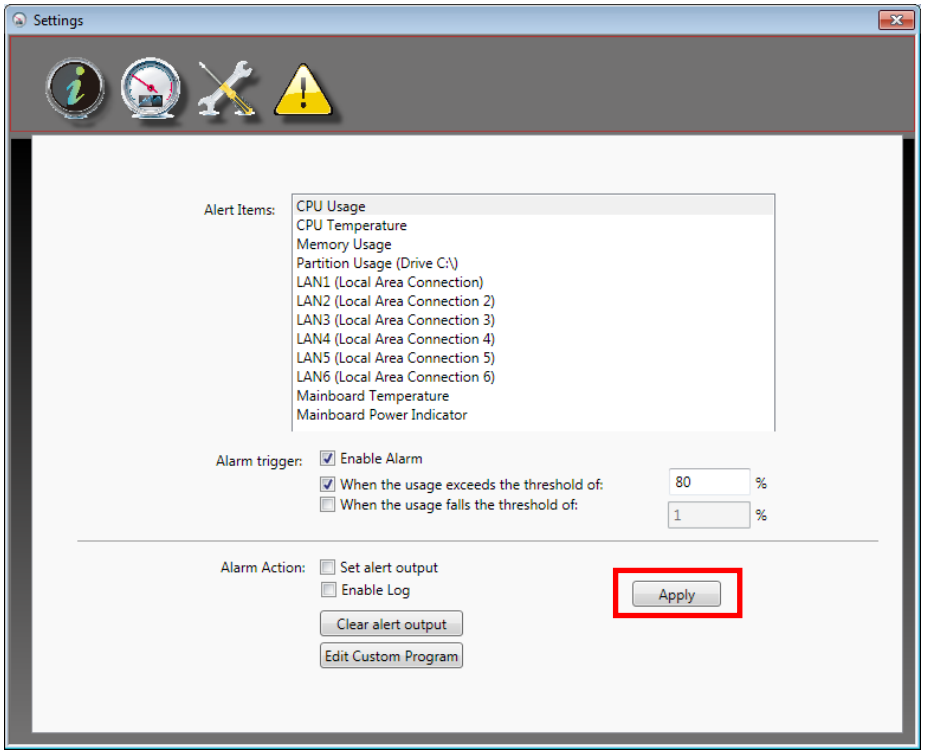


2. Edit the program file name and arguments in the **Alert Start** section and **Alert Stop** section.

NOTE "startAlertoutput.exe" is just an example; you may your own *.exe file name here.



3. Click **Apply** to save the settings.



2

API

The Moxa Proactive Monitoring API provides a set of “C” functions for communicating with hardware devices.

The following topics are covered in this chapter:

- **API Overview**
- **API Function Summary**
  - CPU Information
  - Memory Information
  - UART Information
  - Ethernet Information
  - Disk Information
  - BIOS Information
  - I/O Ports
- **API Details**
  - CPU Information
  - Memory Information
  - UART Information
  - Ethernet Information
  - Disk Information
  - BIOS Information

API Overview

The API provides support for the following tasks:

- Obtaining BIOS information.
- Obtaining CPU information.
- Obtaining the amount of free and used system memory.
- Obtaining disk information.
- Obtaining Ethernet information.
- Set or get IO port status.

API Function Summary

CPU Information

GetAverageCpuUsage	Display average CPU utilization of the system.
GetCpuUsage	Display per core CPU utilization of the system.
GetCpuCount	Display the number of CPU units available.

Memory Information

GetMemUsage	Display the total amount of used disk in the system.
GetMemTotalSize	Display the total amount of disk in the system.
GetMemAvailSize	Show the total amount of available physical memory in the system.

UART Information

GetUartCount	Display the number of UART units available.
GetUartStatus	Display the UART port status.

Ethernet Information

GetEthSpeed	Display specific ethernet port speed.
GetEthLink	Display specific ethernet port link.
GetEthUsage	Display specific ethernet port speed.
GetEthCount	Display the total ethernet port number.

Disk Information

GetDiskUsage	Display specific disk utilization of the system.
GetDiskTotalSize	Display the specific total amount of disk size in the system.
GetDiskAvailSize	Display the specific amount of available disk size.

BIOS Information

GetDeviceName	Display the device name on main board.
GetBiosVer	Display the bios version.
GetSerNum	Display the bios serial number.
GetMilliVolt	Display the sensor voltage in the system.
GetTemperature	Display the sensor temperature in the system.

I/O Ports

GetPwrStatus	Display the power supply status.
SetRelay	Set relay status.

API Details

CPU Information

GetAverageCpuUsage

int GetAverageCpuUsage(double *value);	
Parameters	value: The average CPU usage.
Return Value	Return value(ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetAverageCpuUsage function is used to obtain the average CPU utilization of the system.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int cpuLogicCount = 0;
    int cpuCoreIndex = 0;
    double cpuUsage = 0;

    /* Get CPU Average Usage */
    cpuCoreIndex = 0;
    GetCpuStatus(cpuCoreIndex, &cpuUsage);
    printf("Average CPU Usage = %d%%\n", cpuUsage);

    return 0;
}
```

GetCpuUsage

int GetCpuUsage(int index, double *value);	
Parameters	index: CPU Index (Start from 1). value: The CPU usage of a specific CPU core.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetCpuUsage function is used to obtain CPU utilization of each core by index. To get the total CPU core count, call the GetCpuCount function.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  int cpuLogicCount = 0;`
	`  int cpuCoreIndex = 0;`
	`  double cpuUsage = 0;`
	
	`  /* Get CPU Usage of each core */`
	`  for (cpuCoreIndex = 1; cpuCoreIndex <= cpuLogicCount; cpuCoreIndex++)`
	`  {`
	`  GetCpuStatus(cpuCoreIndex, &cpuUsage);`
	`  printf("CPU Usage of core%d = %d%%\n", cpuCoreIndex, cpuUsage);`
	`  }`
	
	`  return 0;`
	`}`

GetCpuCount

int GetCpuCount(int *value);	
Parameters	value: The total CPU core count.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success.
Description	The GetCpuCount function is used to obtain the number of CPU cores.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  int cpuLogicCount = 0;`
	`  int cpuCoreIndex = 0;`
	
	`  /* Get CPU Logic Count */`
	`  GetCpuCount(&cpuLogicCount);`
	`  printf("CPU Logic Count = %d\n", cpuLogicCount);`
	
	`  return 0;`
	`}`

Memory Information

GetMemUsage

int GetMemUsage(int *value);	
Parameters	value: Memory usage in percentage.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetMemUsage function is used to obtain the memory usage as a percentage.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  double memoryUsage = 0;`
	
	`  /* Get memory usage */`
	`  GetMemUsage(&memoryUsage);`
	`  printf("memoryUsage = %d%%\n", memoryUsage);`
	`  return 0;`
	`}`

GetMemTotalSize

int GetMemTotalSize(double *value);	
Parameters	value: Total memory size in megabytes.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetMemTotalSize function is used to obtain the total amount of physical memory in the system.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  double memoryTotalSize = 0;`
	`  double memoryAvailableSize = 0;`
	
	`  /* Get total memory size */`
	`  GetMemTotalSize(&memoryTotalSize);`
	`  printf("memoryTotalSize = %I64d\n", memoryTotalSize*1024*1024);`
	
	`  return 0;`
	`}`

GetMemAvailSize

int GetMemAvailSize(double *value);	
Parameters	value: Available memory size in megabytes.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetMemAvailSize function is used to obtain the available physical memory in the system.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  double memoryAvailableSize = 0;`
	
	`  /* Get available memory size */`
	`  GetMemAvailSize(&memoryAvailableSize);`
	`  printf("memoryAvailableSize = %I64d\n",`
	`memoryAvailableSize*1024*1024);`
	
	`  return 0;`
	`}`

UART Information

GetUartCount

int GetUartCount(int *value);	
Parameters	value: Available COM ports.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: The system cannot get device name.
Description	The GetUartCount function is used to obtain the available COM ports in the system.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  int uartStatus = 0;`
	`  int uartCount = 0;`
	
	`  /* Get serial port count */`
	`  GetUartCount(&uartCount);`
	`  printf("Uart count = %d\n", uartCount);`
	`  return 0;`
	`}`

GetUartStatus

int GetUartStatus(int index, int *value);	
Parameters	index: COM port index (starts from 1). value: The buffer to hold the return message.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetUartStatus function is used to obtain the status of a specific COM port by index. value = 1: The UART port is currently in use. value = 0: The UART port is available.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int uartStatus = 0;
    int uartIndex = 1;
    int uartCount = 0;

    /* Get serial port count */
    GetUartCount(&uartCount);
    printf("Uart count = %d\n", uartCount);

    /* Get serial port status */
    uartIndex = 1; //COM1
    GetUartStatus(uartIndex, &uartStatus);

    if (uartStatus == 1)
    {
        printf("COM1 is In Use\n");
    }
    else
    {
        printf("COM1 is available\n");
    }

    uartIndex = 2; //COM2
    GetUartStatus(uartIndex, &uartStatus);
    if (uartStatus == 1)
    {
        printf("COM2 is In Use\n");
    }
    else
    {
        printf("COM2 is available\n");
    }
    return 0;
}
```

Ethernet Information

GetEthCount

int GetEthCount(int *value);	
Parameters	value: Total number of available network adapters.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetEthCount function is used to obtain the total number of network adapters.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include <math.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  int connectionStatus = 0;`
	`  int linkSpeed = 0;`
	`  double utilization = 0;`
	`  int networkAdapterCount = 0;`
	
	`  /* Get number of network adapter */`
	`  GetEthCount(&networkAdapterCount);`
	`  printf("Network adapter count = %d\n", networkAdapterCount);`
	
	`  return 0;`
	`}`

GetEthSpeed

int GetEthSpeed(int index, int *value);	
Parameters	index: Network adapter index (starts from 1). value: Current link speed.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetEthSpeed function is used to obtain the link speed of a specific network adapter.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include <math.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int connectionStatus = 0;
    int linkSpeed = 0;
    double utilization = 0;
    int networkAdapterCount = 0;

    /* Get number of network adapter */
    GetEthCount(&networkAdapterCount);
    printf("Network adapter count = %d\n", networkAdapterCount);

    /* Get network link speed */
    GetEthSpeed(1, &linkSpeed);
    printf("Link speed = %d Mbps\n", linkSpeed);

    printf("Network utilization = %d\n", utilization);

    return 0;
}
```

GetEthLink

int GetEthLink(int index, int *value);	
Parameters	index: Network adapter index (starts from 1). value: Connection status of the network adapter.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetEthLink function is used to obtain the link status of a specific network adapter.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include <math.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int connectionStatus = 0;
    int linkSpeed = 0;
    double utilization = 0;
    int networkAdapterCount = 0;

    /* Get number of network adapter */
    GetEthCount(&networkAdapterCount);
    printf("Network adapter count = %d\n", networkAdapterCount);

    /* Get network connection status */
    GetEthLink(1, &connectionStatus);
    printf("Connection Status = %d\n", connectionStatus);

    return 0;
}
```

GetEthUsage

int GetEthUsage(int index, int *value);	
Parameters	index: Network adapter index (Start from 1). value: The network utilization of specific network adapter.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetEthUsage function is used to obtain the network utilization of specific network adapter.
Example	

```
// GetNetworkStatus.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include <windows.h>
#include <math.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int connectionStatus = 0;
    int linkSpeed = 0;
    double utilization = 0;
    int networkAdapterCount = 0;

    /* Get number of network adapter */
    GetEthCount(&networkAdapterCount);
    printf("Network adapter count = %d\n", networkAdapterCount);

    /* Get network utilization */
    GetEthUsage(1, &utilization);

    printf("Network utilization = %d\n", utilization);

    return 0;
}
```

Disk Information

GetDiskUsage

int GetDiskUsage(char *diskpath, double *value);	
Parameters	diskpath: Drive letter. e.g. C:\ value: Disk usage in percentage.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetDiskUsage function is used to obtain the disk usage in percentage.
Example	

```
// GetDiskStatus.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int diskIndex = 0;
    double diskUsage = 0;

    /* Get disk usage */
    GetDiskUsage(diskIndex, &diskUsage);
    printf("Disk Usage = %d %%\n", diskUsage);
    return 0;
}
```

GetDiskTotalSize

int GetDiskTotalSize(char *diskpath, double *value);	
Parameters	diskpath: Drive letter. e.g. C:\ value: Disk total size in megabytes.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetDiskTotalSize function is used to obtain the total disk size in megabytes.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int diskIndex = 0;
    double diskTotalSize = 0;

    /* Get disk total size */
    GetDiskTotalSize(diskIndex, &diskTotalSize);
    printf("Disk Total Size = %d MB\n", diskTotalSize);

    return 0;
}
```

GetDiskAvailSize

int GetDiskAvailSize(char * diskpath, double *value);	
Parameters	diskpath: Drive letter. e.g. C:\ value: Available disk size in megabytes.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetDiskAvailSize function is used to obtain the available disk size in megabytes.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int diskIndex = 0;

    double diskAvailableSize = 0;

    /* Get disk available size */
    GetDiskAvailSize(diskIndex, &diskAvailableSize);
    printf("Disk Avaliable Size = %d MB\n", diskAvailableSize);

    return 0;
}
```

BIOS Information

GetDeviceName

int GetDeviceName(unsigned char *value);	
Parameters	value: Model name. e.g. DA-681A.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0 : Success. ret = -101: Unable to open shared memory.
Description	The GetDeviceName function is used to obtain model name on target device.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    unsigned char deviceName[MAX_PATH];

    /* Get Device Name */
    GetDevName(deviceName);
    printf("Device Name = %s\n", deviceName);

    return 0;
}
```

GetBiosVer

int GetBiosVer(unsigned char *value);	
Parameters	value: BIOS version string.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetBiosVer function is used to obtain BIOS version string.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    unsigned char biosBersion[MAX_PATH];

    /* Get BIOS Version */
    GetBiosVer(biosBersion);
    printf("BIOS Version = %s\n", biosBersion);

    return 0;
}
```

GetSerNum

int GetSerNum(unsigned char *value);	
Parameters	value: Serial number string.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetSerNum function is used to obtain the serial number string.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    unsigned char serialNumber[MAX_PATH];

    /* Get Serial Number */
    GetSerNum(serialNumber);
    printf("Serial Number = %s\n", serialNumber);

    return 0;
}
```

GetPwrStatus

int GetPwrStatus(int port, int *value);	
Parameters	port: Power module index (Start from 1). value: Power module status.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetPwrStatus function is used to obtain the power module status in the system.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    double mainboardPowerStatus = 0;

    /* Get mainboard power status */
    GetPwrStatus (1, & mainboardPowerStatus);
    printf("Mainboard Power Status = %d\n", mainboardPowerStatus);

    return 0;
}
```

NOTE Some MOXA boards may not support power supply indicator. Please refer to the board's manual for information.

SetRelay

int SetRelay(int port, int value);	
Parameters	port: Relay index (Start from 1), reserved. value: 1 = on 0 = off
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	This function can also be used to setting relay status, if you relay set 1, relay is on, on the other hand, set 0, relay is off. **Note:** Only support on DA-820 and DA-681A
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  int powerModuleIndex = 1;`
	`  int relayOutput = 1;`
	
	`  /* Set local relay output */`
	`  SetRelay(powerModuleIndex, &relayOutput);`
	
	`  return 0;`
	`}`

GetMilliVolt

int GetMilliVolt(int index, unsigned int *value);	
Parameters	index: Voltage type. 0 = CPU 1 = DRAM 2 = 5V value: Voltage value.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetMilliVolt function is used to obtain the sensor voltage in the system.
Example	

	`#include "stdafx.h"`
	`#include <windows.h>`
	`#include "MxProMonApi.h"`
	`int _tmain(int argc, _TCHAR* argv[])`
	`{`
	`  double mainboardVoltage = 0;`
	
	`  /* Get mainboard voltage */`
	`  GetMilliVolt(3, &mainboardVoltage);`
	`  printf("Mainboard voltage = %d mV\n", mainboardVoltage);`
	
	`  return 0;`
	`}`

NOTE Some MOXA boards may not support all voltage type. Please refer to the board's manual for information.

GetTemperature

int GetTemperature(int index, unsigned int *value);	
Parameters	index: Temperature type. 0 = cpu 1 = system value: Temperature value.
Return Value	Return value (ret) is zero on success, a negative error code on failure. ret = 0: Success. ret = -101: Unable to open shared memory.
Description	The GetTemperature function is used to obtain the sensor temperature in the system.
Example	

```
#include "stdafx.h"
#include <windows.h>
#include "MxProMonApi.h"

int _tmain(int argc, _TCHAR* argv[])
{
    double mainboardTemperature = 0;

    /* Get mainboard temperature */
    GetTemperature(2, &mainboardTemperature);
    printf("Mainboard Temperature = %d%cC\n", mainboardTemperature,
248);

    return 0;
}
```