

W406 WinCE

User's Manual

First Edition, April 2009

www.moxa.com/product

MOXA®

© 2009 Moxa Inc. All rights reserved.
Reproduction without permission is prohibited.

W406 WinCE 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

Copyright © 2009 Moxa Inc.
All rights reserved.
Reproduction without permission is prohibited.

Trademarks

MOXA is a registered trademark of Moxa Inc.
All other trademarks or registered marks in this manual belong to their respective manufacturers.

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 made periodically to the information in this manual 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: +86-10-6872-3958

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

Table of Contents

Chapter 1	Introduction	1-1
	Overview.....	1-2
	W406 Software	1-2
	Applications Development Environment.....	1-2
	Networking and Communications Capabilities	1-3
	Supported Servers and Daemons	1-3
	Obtaining the Firmware Build Version	1-4
	Memory and File Systems	1-4
	RAM File System	1-4
	Flash File System.....	1-4
	External File System.....	1-4
	Cautions when Using File Systems	1-4
	Using a RAM File System instead of a Flash File System	1-5
	Hive-based Registry.....	1-5
	Connect to Network via Ethernet.....	1-5
	Cellular Networking	1-5
	Inserting an SD Card into the Computer.....	1-6
	Connecting a USB Mass Storage Device to the Computer	1-6
	Serial Ports.....	1-6
	Console Port	1-6
	RS-232/422/485 Serial Ports	1-6
Chapter 2	Getting Started	2-1
	Starting Your W406 Computer.....	2-2
	Resetting Your W406 Computer	2-2
	Boot Loader	2-3
	Operating Your W406 Computer through the Serial Console.....	2-3
	Changing the Network Settings	2-4
	Connecting to a GPRS/EDGE Network.....	2-4
	Disconnecting from a GPRS/EDGE Network	2-6
	GPRS/EDGE Connection Error Detection.....	2-7
	SIM Card Lock Utilities.....	2-7
	Sending and Receiving SMS Messages	2-8
	Sending AT Command	2-10
	Operating Your W406 Computer through a Telnet Client.....	2-11
	User/Group Management.....	2-11
	System Time Management.....	2-12
	Adjusting OS Time Zone	2-12
	SNTP Client.....	2-12
	Starting and Stopping Services	2-13
	Troubleshooting Network Connectivity	2-13
	Simple Network Management Protocol (SNMP).....	2-14
Chapter 3	Web-based Management System	3-1
	Logging onto the Web-based Management System	3-2
	System Information.....	3-2
	Networking/Server Configuration.....	3-3
	Serial Port Configuration	3-3
	Monitoring and Controlling Processes (Threads)	3-4

Launching Processes Automatically	3-4
Monitoring and Controlling Services.....	3-5
Binary/Text File Management	3-5
Appendix A Firmware Upgrade Procedure.....	A-1
Appendix B Application Development.....	B-1
Developing an application with VS2005	B-1
Visual C++ Examples	B-2
Visual C# Examples.....	B-19

The W406 embedded computer has two RS-232/422/485 serial ports, one 10/100 Mbps Ethernet port, an embedded GSM/GPRS/EDGE module, an SD socket interface for storage expansion, one USB 2.0 host port, four digital input channels, and four digital output channels. This combination of features makes the W406 ideal for your wireless embedded applications.

The RISC-based W406 embedded computers come with the Windows® CE operating system pre-installed. Microsoft® Windows® CE 6.0 is an open, scalable, 32-bit operating system that allows users to build a wide range of innovative, small footprint devices. A typical Windows® CE-based device is designed for a specific use, and often runs disconnected from other computers, or distributed as a front-end to a centralized host. Examples include enterprise tools, such as industrial controllers, communications hubs, point-of-sale terminals, and display devices, such as HMI, advertisement appliances, and interactive panels.

The following topics are covered in this chapter:

- ❑ **Overview**
- ❑ **W406 Software**
 - Applications Development Environment
 - Networking and Communications Capabilities
 - Supported Servers and Daemons
- ❑ **Obtaining the Firmware Build Version**
- ❑ **Memory and File Systems**
 - RAM File System
 - Flash File System
 - External File System
 - Cautions when Using File Systems
 - Using a RAM File System instead of a Flash File System
- ❑ **Hive-based Registry**
- ❑ **Connect to Network via Ethernet**
- ❑ **Cellular Networking**
- ❑ **Inserting an SD Card into the Computer**
- ❑ **Connecting a USB Mass Storage Device to the Computer**
- ❑ **Serial Ports**
 - Console Port
 - RS-232/422/485 Serial Ports

Overview

The W406 is an embedded Linux or WinCE computer that features 2 software selectable RS-232/422/485 ports, 1 Ethernet port, and quad-band GSM/GPRS/EDGE 900/1800/850/1900 MHz for cellular communication. It also comes with an SD socket, USB host, and 4 digital input and 4 digital output channels, making it the ideal computer for a variety of industrial applications such as data acquisition, data processing, protocol conversion, and remote device control and monitoring via wireless communication. The W406 comes pre-installed with either Linux or WinCE 6.0, and offers a reliable and powerful computing platform for industrial environments. Programmers will find that the W406 provides a convenient programming environment for producing bug-free industrial applications at a lower cost.

W406 Software

The W406 embedded computer is a ready-to-run, RISC-based, “headless” computer with a robust and network-centric design. It uses the Microsoft® Windows® CE 6.0 operating system. Developers of embedded communication applications will find that the open programming environment makes the W406 well-suited for both new system development and legacy system migration.

Applications Development Environment

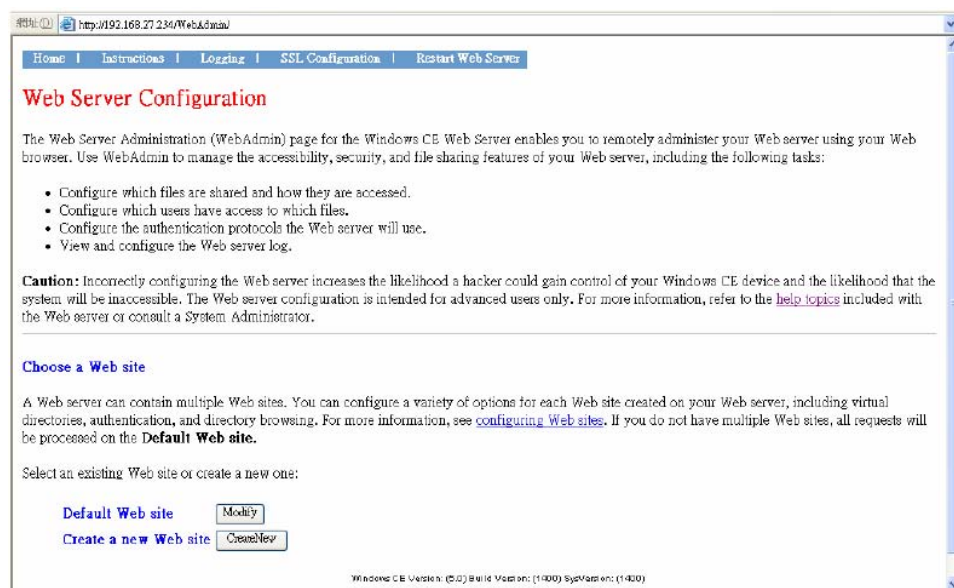
- **C Libraries and Run-times**—Compared to the C libraries and run-times used on a desktop PC running Windows®, the C libraries and run-times on a W406 WinCE are a subset of the WIN32 APIs. The system supports a full ANSI C run-time, standard input/output library, standard input/output ASCII library, and standard ASCII string functions. In addition, C++ compiler exception handling and Run-Time Type Information (RTTI) equivalent to desktop C++ compilers are supported.
- **Active Template Library**—Active Template Library (ATL) for Windows CE is a C++ template library designed to help create small, fast Microsoft® ActiveX® servers. An ActiveX server is a dynamic-link library (DLL) or executable (.exe) that contains one or more Component Object Model (COM) components. COM components can be anything from a simple dialog box to a full ActiveX control.
- **Component Services (COM)**—The Common Object Model (COM) is an operating system-independent, object-oriented system for creating binary software components that can interact with other COM-based components in the same process space, in other processes, or on remote machines.
- **Microsoft® .NET Compact Framework 2.0 with service pack 2**—It offers a choice of languages, initially Microsoft® Visual Basic® and Microsoft® Visual C#, and eliminates the common problems faced with language interoperability.
- **XML**—support for XML Query Language (XQL)
- **Winsock 2.2**—Provides enhanced capabilities over Winsock 1.1, including installable service providers for additional third-party protocols, and Media sense.

Networking and Communications Capabilities

- **Simple Network Management Protocol (SNMP)**—Monitors remote connections to the network.
- **Simple Network Time Protocol (SNTP) Client**—Provides support for synchronizing the device's system time with an SNTP server, and supports Daylight Savings Time.
- **Serial Communications**—In addition to the 16550 UART driver bound to a debug port and the console port, a special driver for 2 additional Moxa serial ports is also included.
- **Network Utilities (IpConfig, Ping, Route)**—Utilities for troubleshooting various network problems.
- **TCP/IP**—Includes IP, Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), Internet Group Membership Protocol (IGMP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), name resolution and registration, and DHCP.

Supported Servers and Daemons

- **Telnet Server**—A sample server that allows remote administration through a standard Telnet client.
- **FTP Server**—A sample server used for transferring files to and from remote computer systems over a network using TCP/IP.
- **File Server**—The File Server functionality in Microsoft® Windows® CE enables clients to access files and other resources over the network.
- **Dial-up Networking**—Consists of RAS client API and the Point to Point Protocol (PPP). RAS and PPP support Extensible Authentication Protocol (EAP) and RAS scripting.
- **Watchdog**—A CPU Hardware function for reset CPU in a user specified time interval. You must call the MOXA library function to trigger it.
- **Web Server (HTTPD)**—Includes ASP, ISAPI Secure Socket Layer support, SSL 2, SSL 3, Transport Layer Security (TLS/SSL 3.1) public key-based protocols, and Web Administration ISAPI Extensions.



Obtaining the Firmware Build Version

There are two ways to obtain the firmware version of W406 embedded computers. This information is particularly important for identifying the features supported by the computer.

- Examine the welcome message after you log on to the computer.
- Log on to the web-based management system (described in a later chapter) to view the system information.

Memory and File Systems

The 32 MB of SDRAM is divided into two main parts. The main memory, which houses the operating system and user applications, has a capacity of about 20 MB. The kernel image occupies the remainder of the memory space.

RAM File System

The internal file system in the W406 computer controls access to flash and also provides file storage in the object store, which is in the RAM. The root directory is a RAM file system of size 4 MB. Child directories such as “Windows,” “Temp,” “My Documents,” “Network,” and “Program Files” are under the root directory. They can be used for storing temporary files for your applications. However, do not place persistent files or applications in these directories because they will be deleted when the system is shut down. Instead, place them in the “NORFlash” directory.

Flash File System

The Flash file system provides persistent storage for applications and related data, even when the main power supply is lost. The system integrates the read-only files that are stored on the Flash with the read/write files from both applications and users. A child directory named “NORFlash” is created under the root; the size of the directory is 3 MB.

External File System

The additional file systems from USB and SD storage devices are placed in the root of the internal file system. If you intend to use these devices to port data between your PC and the W406 computer, you should format them using the FAT file system on your PC.

Cautions when Using File Systems

We recommend storing your programs only in the on-board NOR Flash. Please store the log data generated by your programs in an external storage device such as an SD card or Network File System. Note that a Network File System will generally provide more storage space than the SD card. In addition, it is easier to replace a full or damaged Compact Flash than an on-board NOR Flash.

A NOR Flash has a life cycle of 100,000 write operations in block (128 KB) level, and does not support BBM (Bad Block Management). For this reason, the FAT file system would not know when a flash block has reached its cycle, and would try to scan the block repeatedly.

FAT sequentially searches for free memory space for write operations. After deleting many files, the memory space could become fragmented, making it more difficult to search for free space. If your program updates (deletes and then creates) a file frequently, it is quite possible that the program writes data to the same flash area. In the long run, FAT would be blocked when scanning the area and would cause the operating system to hang.

An SD card has its own life cycle. Since most SD cards are made from a NAND Flash, their hardware controllers implement BBM. This feature allows FAT to skip bad blocks if they exist. Furthermore, the memory space of an SD card is much larger than that of the NOR Flash. Using this space cautiously will ensure that its life cycle is not exceeded. When creating a file for storing log data, we suggest creating a large empty file (e.g., 30 MB), and then writing data evenly to that space. When the space is used up, the program rewinds the write operations. As a result, the number of write operations to each block is reduced.

Using a RAM File System instead of a Flash File System

Even though data in the RAM file system will be deleted after shutting off the power, using the RAM file system has several advantages over using the Flash file system, including faster read/write access and not needing to deal with the life cycle issue.

For important applications that relay data back to the host directly, you should write the necessary log data to the RAM file system. After the host accesses the data, the application erases the data, freeing up the memory space for further use.

The embedded computer has limited resources, and designers should decide if storing data in a file system is really necessary. If it is necessary, be sure to choose the most appropriate file system.

Hive-based Registry

The registry for the W406 is a hive-based registry instead of a RAM-based registry. The hive-based registry stores registry data inside files, or hives, which can be stored in any file system. This eliminates the need for performing backup and restoring power.

The registry data will be automatically flushed by default. If **AutoFlush** is disabled, the **RegFlushKey** function is needed when you change the registry value in your applications. The **RegAutoFlush.exe** utility is provided for users to change the auto-flush setting.

The **RegAutoFlush.exe** utility can be found at the utility_tools directory on CD

Type **RegAutoFlush -d** to disable auto-flush setting.

```
\> RegAutoFlush -d
```

OR type **RegAutoFlush -e** to enable auto-flush setting.

```
\> RegAutoFlush -e
```

Connect to Network via Ethernet

The W406 computer offers a 10/100M RJ-45 Ethernet port for network redundancy.

Cellular Networking

The W406 computer has been embedded with a GSM/GPRS/EDGE cellular module for a reliable and stable wireless communication. It supports quad-band GSM/GPRS/EDGE 850/900/1800/1900 MHz, and GPRS/EDGE Class 10. Meanwhile, it offers SMS tunnel mode for auto configuration and SIM card authentication.

Inserting an SD Card into the Computer

The W406 is equipped with an SD slot. When an empty SD card is inserted into the slot, the computer automatically formats it to the FAT system. This process takes a few minutes to complete. After an SD card is inserted, the embedded computer will create a directory named "StorageDisk" under the root directory. The "StorageDisk" directory controls access to the SD storage space. The embedded computer will create a directory called "StorageDisk2" if another USB storage device is plugged in at a later time.

Connecting a USB Mass Storage Device to the Computer

The USB mass storage device is considered to be highly portable between your PC and a computer that does not support the TFAT system. We suggest that you format your devices with the FAT format. When the first USB storage device is plugged into the slot on the back of the computer, a directory named "USBDisk" under the root directory is created in the internal file system as a link to the storage device. The embedded computer will create a directory called "StorageDisk2" if another SD storage device is plugged in at a later time.

The following table lists USB mass storage devices that have been tested successfully for compatibility.

Vendor	Device Name	Size
CRUZER	mini	128 MB
Intel	Flash memory	128 MB
Abocom		128 MB
PQI		256 MB
Transcend	JetFlash	512 MB
Transcend	JetFlash	1 GB



ATTENTION

Some USB storage devices may not be detected by the system. We suggest that you use one of the devices listed in the above table, since these USB mass storage devices have been tested successfully for compatibility.

Serial Ports

Console Port

The serial console port, located on the bottom panel is a 4-pin pin-header RS-232 port. It is designed for serial console terminals, which are useful for viewing boot-up messages, system configuration and develop applications.

RS-232/422/485 Serial Ports

The W406 computer contains two software-selectable RS-232/422/485 ports for reliable and stable serial communication. They support non-standard baudrates settings up to 921,600 bps.

2

Getting Started

In this chapter, we explain how to use a PC to operate a W406 embedded computer. We will refer to the PC that connects to the W406 as a *development workstation*, and the W406 embedded computer will be called a *target computer*.

development workstation = PC used to operate the embedded computer
target computer = W406 embedded computer

In addition, we describe the steps you should follow to carry out certain operations, such as setting the system time and troubleshooting network connectivity. Some of these operations can be carried out using system commands after gaining access to the target computer, and others can be carried out using a web-based management system. The web-based management system is described in a later chapter.

The following topics are covered in this chapter:

- Starting Your W406 Computer**
- Resetting Your W406 Computer**
- Operating Your W406 Computer through the Serial Console**
- Changing the Network Settings**
- Operating Your W406 Computer through a Telnet Client**
- User/Group Management**
- System Time Management**
- Starting and Stopping Services**
- Troubleshooting Network Connectivity**
- Simple Network Management Protocol (SNMP)**

Starting Your W406 Computer

Connect the SG wire to the shielded contact located in the upper left corner of the W406 computer, and then power it up by connecting it to the power adaptor. It takes about 30 to 60 seconds for the system to boot up. Once the system is ready, the “Ready” LED will light up. The light will stay lit until you shut down the computer.

Resetting Your W406 Computer

When the target computer stops responding, or an application locks up, or the target computer fails to work normally, you may need to restart or reset the target computer's operating system. We provide four ways to restart or reset the operating system.

Warm-Start: When the computer is powered on, insert a pin into the “Reset” hole next to the serial console, and hold for 1 to 2 seconds. The computer will reboot automatically.

Cold-Start: Unplug the power line and then plug it back in again. The computer will reboot automatically.

Resetting to Factory Defaults: If the computer is not working properly, and you want to reset it back to factory default settings, press and hold the “Reset” button for at least 5 seconds. The buzzer will sound while the factory default settings are being loaded. After the factory defaults have been loaded, the computer will reboot automatically.

Resetting the System: In rare circumstances, the FAT file system may be damaged by executing improper applications, or due to an unstable power supply. In this case, the computer may fail to boot up if the FAT table crashes. In order to get the system back up and running, you will need to format the flash disk and reset the operating system. Note that all user files and configurations will be erased. The following steps show how to format the flash disk through boot loader utilities.

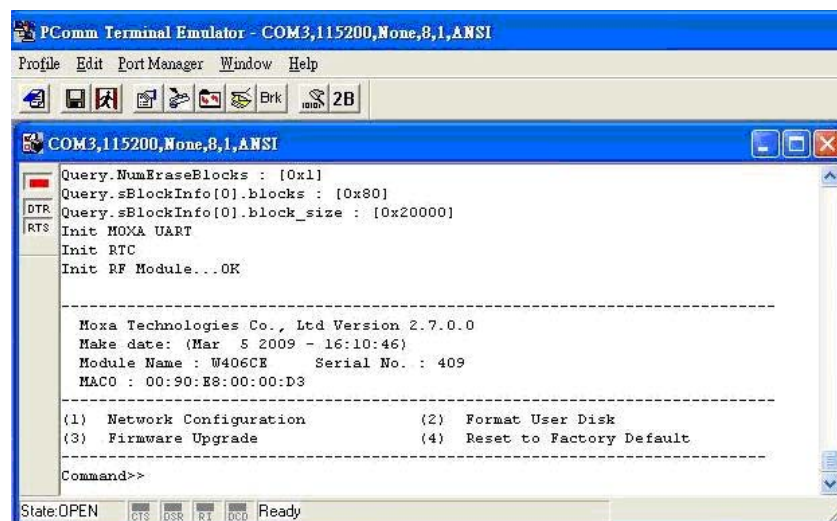
Step 1: Power off the W406 embedded computer.

Step 2: Connect the W406 to your PC using the console port cable.

Step 3: Start a terminal program with the settings: Baudrate = 115200, no hardware flow control, 8 N 1, character set VT100.

Step 4: Hold in the “DEL” key on your PC.

Step 5: Power on the W406. You will be guided to boot loader utility menu.



```
PCComm Terminal Emulator - COM3,115200,None,8,1,ANSI
Profile Edit Port Manager Window Help
[Icons] Erk 2B
COM3,115200,None,8,1,ANSI
Query.NumEraseBlocks : [0x1]
Query.sBlockInfo[0].blocks : [0x80]
Query.sBlockInfo[0].block_size : [0x20000]
Init MOXA UART
Init RTC
Init RF Module...OK

-----
Moxa Technologies Co., Ltd Version 2.7.0.0
Make date: (Mar 5 2009 - 16:10:46)
Module Name : W406CE Serial No. : 409
MACO : 00:90:E8:00:00:D3
-----
(1) Network Configuration          (2) Format User Disk
(3) Firmware Upgrade              (4) Reset to Factory Default
-----
Command>>
```

Step 6: Type "2" for "Format User Disk", and then press **Enter**.

Step 7: After a few seconds, you will see the bootloader menu again. Unplug the power line and then plug it back again. It takes about 3 minutes to reset the operating system.

Boot Loader

For the OS stability issue, we provide an easy and useful function with Boot Loader for the following tasks:

1. Reset to default: you can set the flag to reset WinCE 6.0 to factory default settings.
2. Format storage flash: The CE 6.0 file system is an FAT that can be damaged by unstable power or improper application execution. If the FAT table crashes, do not start up the OS. You can, however, format the file system and reboot your W406.

At startup, the W406 will check the file system and re-partition it if the file system is empty.

3. Firmware upgrade: If you find a new firmware from Moxa's website, you can upgrade the firmware with this function (details in appendix A).

Go to the boot loader menu from serial console:

Step 1: Power off your W406 device.

Step 2: Make sure the serial console wire is connected to your PC correctly.

Step 3: Go to [Start] → [Programs] → [Accessories] → [Communication] → [Terminal] to create a new terminal communication with the following settings: baudrate of 115200, no hardware flow control, 8 N 1, and character set VT100.

Step 4: Activate this terminal window on your PC.

Step 5: Hold "DEL" key continuously.

Step 6: Power on the W406 device.

Operating Your W406 Computer through the Serial Console

The serial console port (located on the bottom panel) gives users a convenient way of connecting the development workstation to the console utility of the target computer. This method is particularly useful when using the computer for the first time.

After you have wired a serial cable, go back to the development workstation and start a terminal program (e.g., HyperTerminal) by using the settings shown below for the serial console port.

Baudrate	115200 bps
Parity	None
Data bits	8
Stop bits	1
Flow Control	None
Terminal	ANSI

After a successful connection, type the login name and password as requested to log on to the computer. The default values are both **admin**.

Login: admin

Password: admin

Changing the Network Settings

The W406 computer has one network interface. There is no default IP address; the default network setting is configured as DHCP. If you have a DHCP server, simply connect the W406 to your network using an Ethernet cable. An IP address will automatically be appointed to the W406. If you would like to use a fixed IP address, you should use **netconfig** command.

The **netconfig** command is a utility that is used to complete the task. Before changing the IP addresses, type **netconfig -h** to list the help for this command.

\> **netconfig -h**

```
Usage: netconfig -n <AdapterName | Alias> [-EnableDHCP] [-i <IP address>] [-m <etmask>]
[-g <gateway>] [-d <DNS server>] [-w <WINS Server>] [-noask] e.g.: netconfig -n
IXP425ETHNPE1 -i 192.168.10.101 -g 192.168.10.254: netconfig -n CS89501 -i
192.168.12.101 -m 255.255.0.0: netconfig -n PCIRTL81391 -EnableDHCP
```

Alias: LAN1=CS89501

For example, if your development workstation has a LAN port at 192.168.1.1, and the IP address of the Domain Name Server (DNS) is 192.168.2.6, execute the following command.

\> **netconfig -n LAN1 -i 192.168.1.5 -m 255.255.255.0 -g 192.168.1.254 -d 192.168.2.6**

Use the **netconfig** command to view the updated settings.

\> **netconfig**

```
LAN1 Interface Configuration:
IP Address:      192.168.1.5
SubNet Mask:    255.255.255.0
Gateway:        192.168.1.254
DNS:            192.168.2.6
```

Connecting to a GPRS/EDGE Network

Before connecting to a GPRS/EDGE network, make sure the SIM card is properly installed and the antenna is connected. (Please refer to W406 Hardware User's Manual for installation details.) Note that the SIM card must be installed when the embedded computer is powered off. The LED indicators on the top panel can be used to check the signal strength.

The W406 provides two programs to help users connect to a network via cellular communication. When the SIM card is inserted, **GPRSAutodetect.exe** will detect the configuration and create the entry automatically if the **-I** parameter is entered. It will log the connection process into the **GPRSLog.txt** file. After the entry has been created, you can use **GPRSConnect.exe** to connect to the Internet via cellular module. The **GPRSConnect.exe** file also supports an automatic reconnection function to retry the connection if the connection is not established.

NOTE: Do not execute the **-d** option when the reconnection function is activated and the connection has not been established, or else the reconnection function will be invalid.

1. To create an entry for connection, use the following command:

```
/> gprsautodetect
```



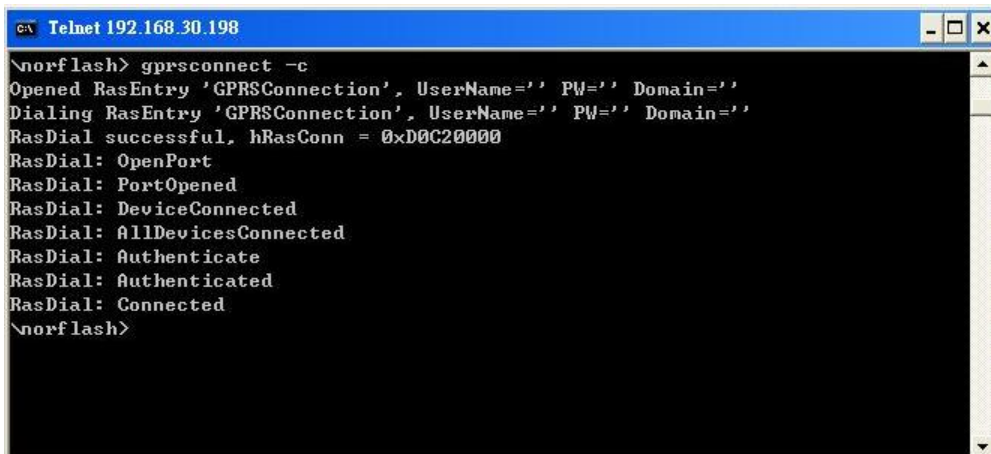
```

Telnet 192.168.30.198
\norflash> autodetect
Detecting baudrate...OK
    Current baudrate : 115200
Checking SIM Card status...OK
    PIN Authentication Success!!
Searching for APN...OK
    Operator : Chunghwa Teleco
    APN : Internet
SetBaudRate...OK
Create Entry "GPRSConnection"...OK
\norflash> _

```

2. To make a connection, use the following command:

```
/> gprsconnect -c
```



```

Telnet 192.168.30.198
\norflash> gprsconnect -c
Opened RasEntry 'GPRSConnection', UserName='', PW='', Domain=''
Dialing RasEntry 'GPRSConnection', UserName='', PW='', Domain=''
RasDial successful, hRasConn = 0xD0C20000
RasDial: OpenPort
RasDial: PortOpened
RasDial: DeviceConnected
RasDial: AllDevicesConnected
RasDial: Authenticate
RasDial: Authenticated
RasDial: Connected
\norflash>

```

If you want to activate the reconnection function and log the process, use **-r [seconds]** to reconnect every few seconds and use **-l** to log the process.

NOTE: If the **[seconds]** parameter needs to be changed, execute **gprsConnect -r -d** to deactivate the reconnection function and then restart the reconnection function.

3. Connecting by GPRS/EDGE takes only a few seconds. While still connecting, you should see the following message from the command shell:

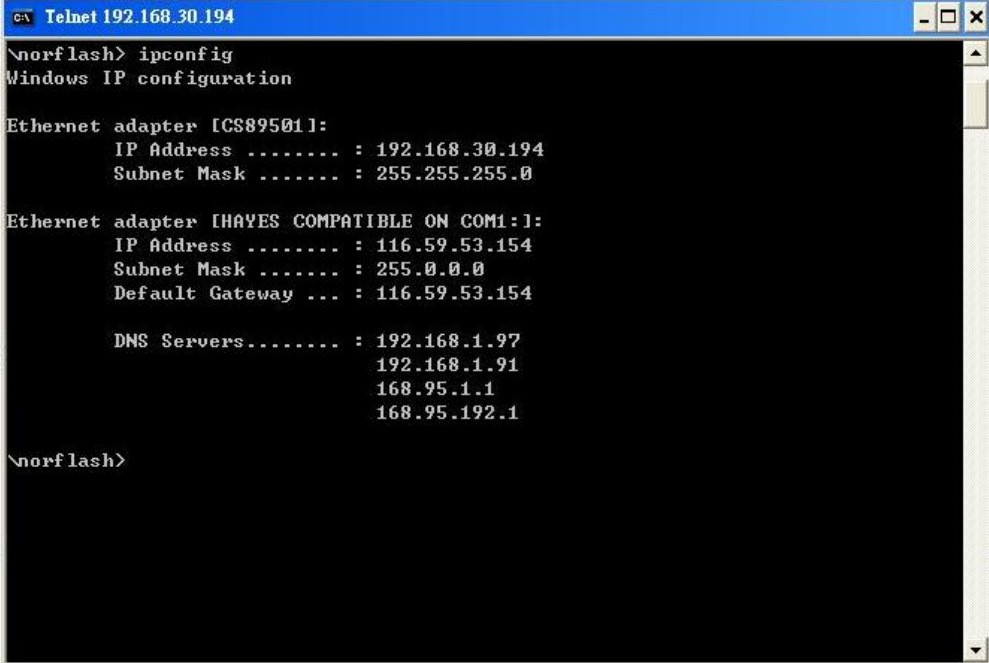
Connecting... Please wait a few seconds.

4. When the embedded computer has connected successfully to the GPRS/EDGE network, you should see the message **Connected**.

5. To verify the connection, type:

```
/>ipconfig
```

The ppp network interface should show up as follows



```

c:\ Telnet 192.168.30.194
\norflash> ipconfig
Windows IP configuration

Ethernet adapter [CS89501]:
    IP Address . . . . . : 192.168.30.194
    Subnet Mask . . . . . : 255.255.255.0

Ethernet adapter [HAYES COMPATIBLE ON COM1:1]:
    IP Address . . . . . : 116.59.53.154
    Subnet Mask . . . . . : 255.0.0.0
    Default Gateway . . . : 116.59.53.154

    DNS Servers . . . . . : 192.168.1.97
                          192.168.1.91
                          168.95.1.1
                          168.95.192.1

\norflash>
```

Disconnecting from a GPRS/EDGE Network

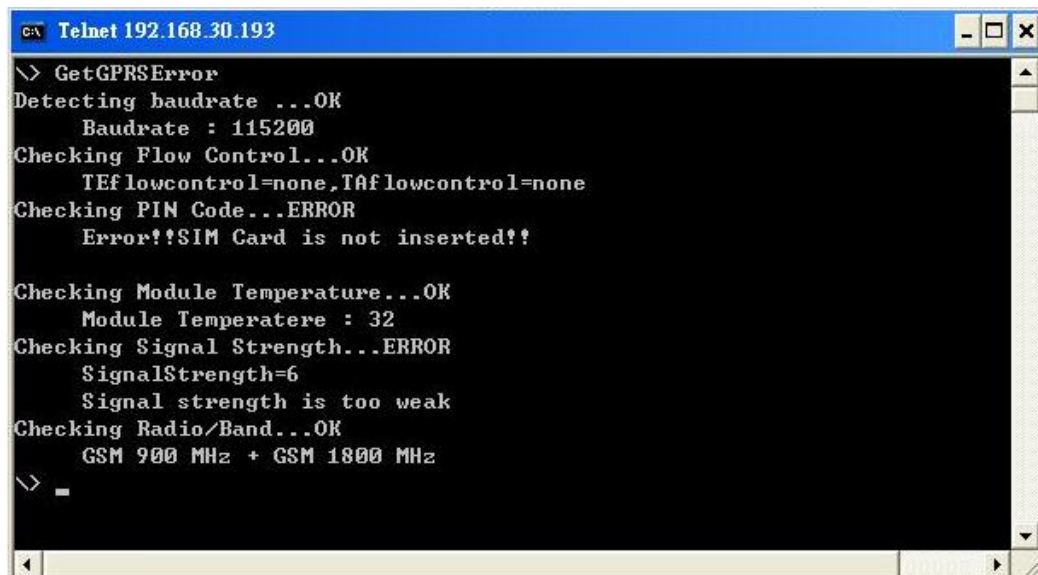
To disconnect from the GPRS/EDGE network, type

```
/>gprsconnect -d
```

After a few seconds, the embedded computer will disconnect from the GPRS/EDGE/EDGE network. A notification message will NOT be shown.

GPRS/EDGE Connection Error Detection

To detect connection problems, **GetGPRSError.exe** is provided for users to diagnose problems during the network connection process. This utility will execute a series of steps to check whether the configuration is correct or not, and diagnostic information will be logged in the **GPRSLog.txt** text file. You can check the text file to determine the problem that caused the connection problem, or you can send this file to us.



```

c:\> Telnet 192.168.30.193
\> GetGPRSError
Detecting baudrate ...OK
    Baudrate : 115200
Checking Flow Control...OK
    Teflowcontrol=none,Taf flowcontrol=none
Checking PIN Code...ERROR
    Error!!SIM Card is not inserted!!

Checking Module Temperature...OK
    Module Temperatere : 32
Checking Signal Strength...ERROR
    SignalStrength=6
    Signal strength is too weak
Checking Radio/Band...OK
    GSM 900 MHz + GSM 1800 MHz
\> =

```

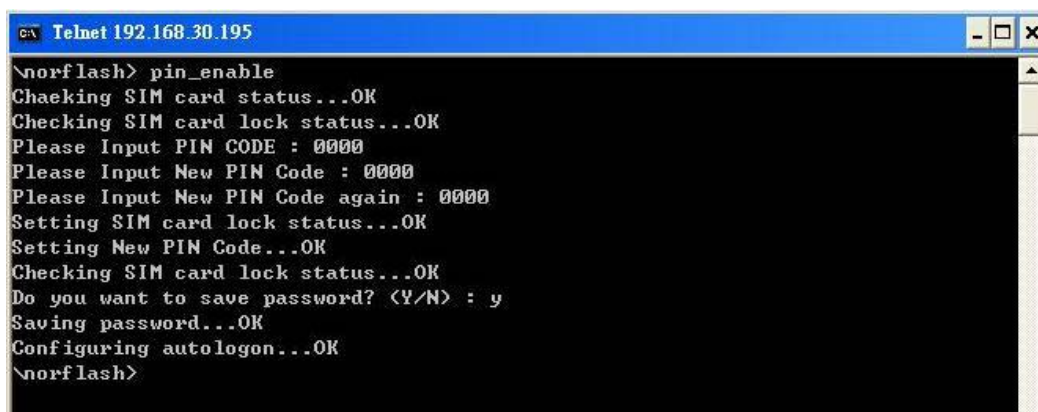
SIM Card Lock Utilities

The W406 provides three utilities, **PIN_Authentication.exe**, **PIN_Enable.exe**, and **PIN_Disable.exe**, for users to enable, disable, and authenticate the SIM card lock. During the process of authentication, you may need to wait a few seconds.

NOTE: Make sure your PIN code is correct. After three failed attempts, the SIM card will be locked and the PUK code will be required to unlock the SIM card. After ten failed attempts at entering the PUK code, the SIM card will be invalidated and no longer operable.

To enable the SIM card lock utility, type

/>PIN_Enable



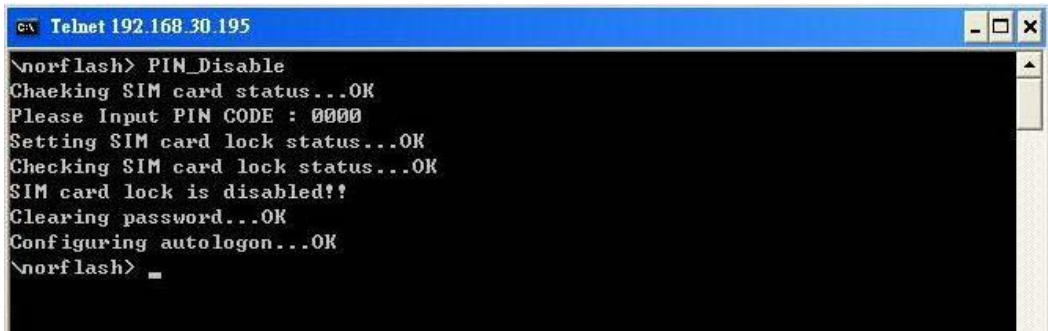
```

c:\> Telnet 192.168.30.195
\norflash> pin_enable
Chaeking SIM card status...OK
Checking SIM card lock status...OK
Please Input PIN CODE : 0000
Please Input New PIN Code : 0000
Please Input New PIN Code again : 0000
Setting SIM card lock status...OK
Setting New PIN Code...OK
Checking SIM card lock status...OK
Do you want to save password? <Y/N> : y
Saving password...OK
Configuring autologon...OK
\norflash>

```

To disable the SIM card lock utility, type

/>PIN_Disable



```
C:\> Telnet 192.168.30.195

\norflash> PIN_Disable
Chaeking SIM card status...OK
Please Input PIN CODE : 0000
Setting SIM card lock status...OK
Checking SIM card lock status...OK
SIM card lock is disabled!!
Clearing password...OK
Configuring autologon...OK
\norflash> _
```

To authenticate the PIN code, type

/>PIN_Authentication



```
C:\> Telnet 192.168.30.195

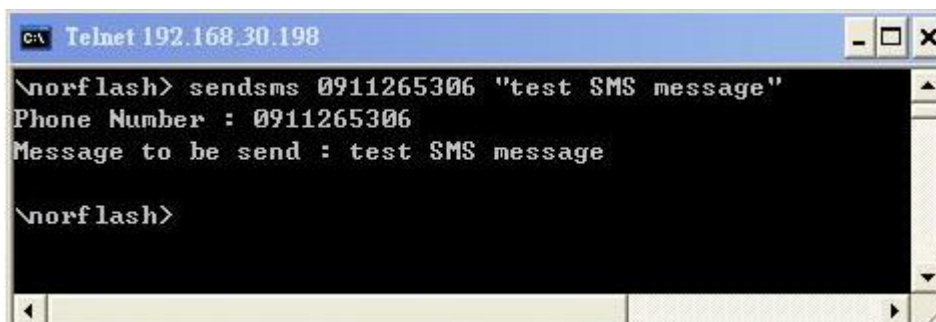
\norflash> PIN_Authentication
Chaeking SIM card status...OK
SIM card is ready to use!!
\norflash>
```

Sending and Receiving SMS Messages

The W406 provides the utilities **SendSMS.exe** and **ReceiveSMS.exe** for users to send and receive SMS messages.

There are two ways to send SMS messages, you can type the message you want to send directly, or store it in a text file and read the message from the text file.

To send an SMS message directly, type “**SendSMS [PhoneNumber] [Message]**”
/>SendSMS 0911265306 “test SMS message”



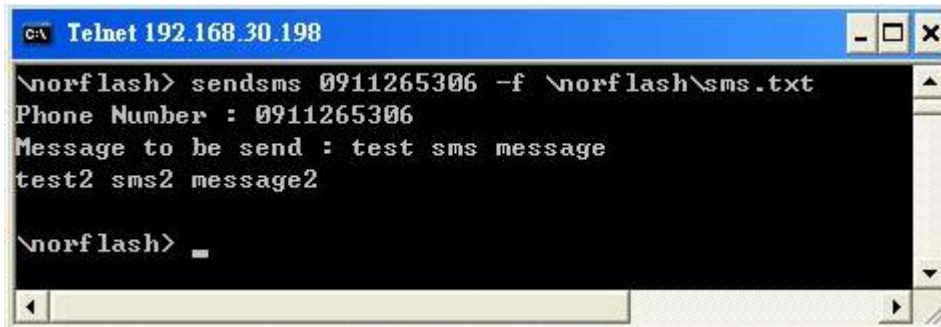
```
C:\> Telnet 192.168.30.198

\norflash> sendsms 0911265306 "test SMS message"
Phone Number : 0911265306
Message to be send : test SMS message

\norflash>
```

To send SMS messages via a text file, type “SendSMS [Phone Number] -f [filename]”
/>>SendSMS 0911265306 -f SMS.txt


NOTE: 1.You must specify the absolute path for the filename or this function may fail.
 2. This utility only transmits SMS messages in text mode.



```
C:\ Telnet 192.168.30.198
\norflash> sendsms 0911265306 -f \norflash\sms.txt
Phone Number : 0911265306
Message to be send : test sms message
test2 sms2 message2
\norflash> _
```

To receive SMS messages, type

/>>ReceiveSMS [message number]



```
C:\ Telnet 192.168.30.198
\norflash> receivesms 2
Date : 09/03/20
Time : 16:17:02+32
Phone : 886911265306
Status : REC READ
Message : Cabba
\norflash>
```

Sending AT Command

The W406 provides the **SendAT.exe** utility for users to send AT commands to wireless modules.

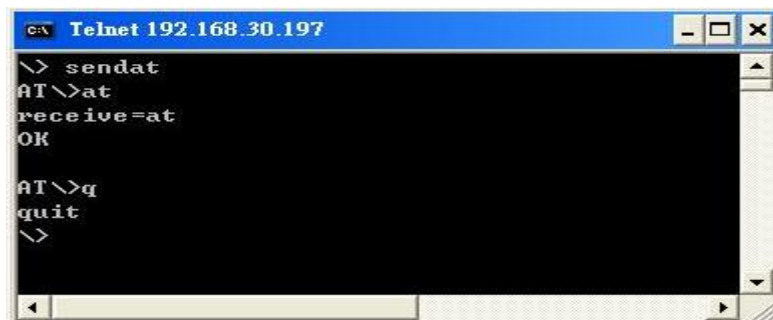
To enter the command mode, type

```
/>SendAT
```

And then you will enter the AT command mode and you can type the AT command. Press enter to send the AT command.

To quit the command mode, type

```
/AT>q
```

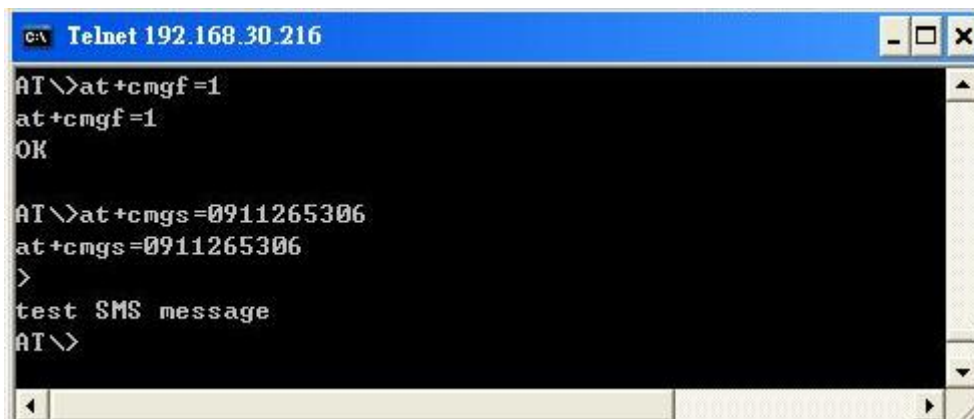


```
C:\> Telnet 192.168.30.197
> sendat
AT>at
receive=at
OK

AT>q
quit
>
```

To send text SMS message via sendat.exe you need to do the following steps:

1. Type **at+cmgf=1** to set message format to text mode.
2. Type **at+cmgs=xxxxxxx**, where **xxxxxxx** is the phone number you want to send SMS messages to.
3. Type the SMS message you want to send and then press **Enter**.
(Note: You don't need to enter ctrl+z to send the message or this message won't be sent.)



```
C:\> Telnet 192.168.30.216
AT>at+cmgf=1
at+cmgf=1
OK

AT>at+cmgs=0911265306
at+cmgs=0911265306
>
test SMS message
AT>
```

Operating Your W406 Computer through a Telnet Client

Use a cross-over Ethernet cable to connect directly from your development workstation to the W406 computer, or use a straight-through Ethernet cable to connect the computer to a LAN hub or switch. Next, use the Telnet client in your development workstation to connect to the Telnet console utility of the W406 computer. After connecting successfully, type the login name and password as requested to log on to the computer.

Login: admin

Password: admin

After logging on through the console port or through a Telnet client, a list of commands are available to operate the computer. Use "HELP" to display all of the commands and type "HELP [command name]" to display extended help for the given command. Some of these commands, such as "DATE" and "TIME" are very useful for managing the computer's system time. Other commands, such as "DIR" and "MKDIR" are good utilities for file management. For example, to inspect the file structure of the root directory, type **DIR**.

```
> dir /b
   Network
   NORFlash
   My Documents Program Files Temp
   Windows
```

User/Group Management

User Group: You should assign specific services, such as ftp and Telnet, to defined user groups so that these services are accessible only by the users within the permissible user group. Three user groups, namely **ftpd**, **telnetd**, and **httpd**, are already created by default for your convenience.

Adding a Group: Use the command **useradd -g <groupName>** to create a user group.

```
\> useradd -g yyyy
   group yyyy has been added.
```

Deleting a Group: To remove a group, use the command **userdel -g <groupName>**.

```
\> userdel -g yyyy
   group yyyy has been removed.
```

Adding a User: Use the command **useradd <newUserID>** to add a user to the system. The user's password, by default, is the same as the user name.

```
\> useradd xxxx
   user xxxx has been added.
```

In addition, you can permit this user to access a particular service by typing **-g** followed by the user group name of the service, i.e., **useradd -g <groupName> <newUserID>**. For example,

```
\> useradd -g telnetd xxxx
user xxxx is existent
group telnetd is existent
user xxxx has been added to group yyyy
```

Deleting a User: Use the command **userdel <userID>** to delete a user from the system. User “admin” **CANNOT** be deleted.

```
\> userdel xxxx
user xxxx has been deleted
```

You can also just remove a user from a user group by using the command **userdel -g <groupName> <newUserID>**. For example,

```
\> userdel -g yyyy xxxx
user xxxx has been removed from group yyyy
```

System Time Management

Setting the System Time Manually: Use the *date* and *time* commands to query the current system date/time, or to set a new system date/time.

```
\> date
The current date is: Tuesday, November 22, 2005
Enter the new date (mm-dd-[yy]yy): 12-23-05
\> date /T
Wednesday, November 23, 2005
\> time
The current time is: 5:27:17 PM
Enter the new time (hh:mm:ss): 16:02:00
\> time /T
4:02:04 PM
```

Adjusting OS Time Zone

In the web manager, you can adjust your current system Time Zone.

After the time zone has been changed, the system time will not be updated automatically; you must reboot your system to take effect.

SNTP Client

In the web manager, you can adjust your SNTP setting.

After the SNTP is enabled, the system time will automatically update so you must reboot your system for updates to take effect.

Starting and Stopping Services

After booting up, the W406 computer runs several services continuously to serve requests from users or other programs. Some important services are telnet (“TEL0:”), console (“CON0:”), world wide web HTTP (“HTP0:”), and file transfer FTP (“FTP0:”). If you rarely use these services, you can still start up or stop a service with its associated name by using the **services** command. For example:

Start the FTP service with the following command:

```
\> services start FTP0:
```

Stop the FTP service with the following command:

```
\> services stop FTP0:
```

Troubleshooting Network Connectivity

The **ipconfig** tool prints the TCP/IP-related configuration data of a host, including the IP addresses, gateway, and DNS servers.

```
\> ipconfig /all
```

```
Windows IP configuration
Ethernet adapter Local Area Connection:
IP Address: 192.168.30.127
Subnet Mask: 255.255.255.0
Adapter Name: CS89501
Description: CS89501
Adapter Index: 2
Address: 00 90 e8 00 d1 23
DHCP Enabled: NO

Host name: W406
Domain Name: moxa.com
DNS Servers: 192.168.1.99
             192.168.1.98
NODETYPE: 8
Routing Enabled: NO
Proxy Enabled: NO
```

To troubleshoot network connectivity or name resolution, use the **ping** command. This command verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The corresponding return Echo Reply messages are displayed, along with round-trip times. For more information, type **ping** without parameters.

```
\> ping www.moxa.com
```

```
Pinging Host www.moxa.com [192.168.1.16]
Reply from 192.168.1.16: Echo size=32 time<1ms TTL=126
Reply from 192.168.1.16: Echo size=32 time<1ms TTL=126
Reply from 192.168.1.16: Echo size=32 time<1ms TTL=126
```

The **route** utility allows you to view or modify network routing tables. Type this command without parameters to view a list of functions.

```
\> route
```

To view current routing items in the tables, type

```
\> route PRINT
```

To add a routing item on network interface 1, type

```
\> route ADD 192.168.0.0 MASK 255.255.0.0 192.168.15.254 IF 2
```

To delete a routing item, type
`\> route DELETE 192.168.0.0`

Simple Network Management Protocol (SNMP)

SNMP is the standard Internet protocol for network management, and belongs to the TCP/IP protocol suite. SNMP was developed to monitor and manage networks. It uses a distributed architecture that consists of agents and managers:

- The SNMP agent is an SNMP application that monitors network traffic and responds to queries from SNMP manager applications. The agent also notifies the manager when significant events occur by sending a trap.
- An SNMP manager is an SNMP application that generates queries to SNMP-agent applications and receives traps from SNMP-agent applications.

The W406 computer installs an SNMP agent to serve as an SNMP device. You should install the SNMP manager on the workstation computer (for example, a Linux system) that monitors the network. After installing the nodes, you need to configure the SNMP manager and agent.

To check SNMP agent capabilities on a target W406 computer (e.g., network IP at 192.168.3.127), log on to the workstation computer on which the SNMP manager resides, and type:

```
\> snmpwalk -v 2c -c public 192.168.3.127 system
SNMPv2-MIB::sysDescr.0 Microsoft Windows CE Version 6.0 (Build 1400)
SNMPv2-MIB::sysObjectID.0 SNMPv2-SMI::enterprises.8691.13.406
SNMPv2-MIB::sysUpTime.0 1282929
SNMPv2-MIB::sysContact.0 Your System Contact Here
SNMPv2-MIB::sysName.0 WindowsCE
```

You will see a series of messages from the SNMP agent on the W406 computer. You may then proceed to monitor and manage the computer.

Web-based Management System

Note: You must use Internet Explorer 5.5 or above to access the web-based management system.

The W406-CE ready-to-run embedded computer is a network-centric platform designed to be used as front-end computers for data acquisition and industrial control. Due to the distributed characteristics of the devices that these computers control, they often reside in harsh environments away from the system administrator. To manage these computers, operations such as networking/server configuration, file management, and process (thread) monitoring/control are critical.

The following topics are covered in this chapter:

- Logging onto the Web-based Management System**
- System Information**
- Networking/Server Configuration**
- Serial Port Configuration**
- Monitoring and Controlling Processes (Threads)**
- Launching Processes Automatically**
- Monitoring and Controlling Services**
- Binary/Text File Management**

Logging onto the Web-based Management System

The web-based management system installed in the W406 computer incorporates often-used features into CGI pages, and categorizes them on a menu bar.

Before attempting to connect to the management system, make sure the network connection from your PC to the target computer active, and you are able to use the PC's Internet browser. The following steps describe how to log on to the web-based system.

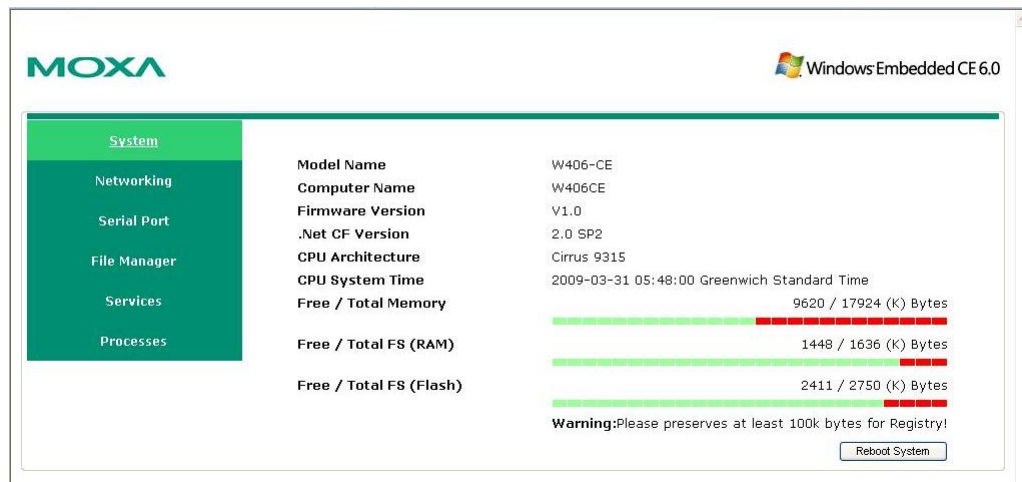
1. Type the IP address of the target computer in the browser's address box. When the main page appears, click on **Web-Based Management**.
2. Enter your user ID and password in the corresponding fields (both are case sensitive) and then press enter to request access to the management system. The system checks your data with the users previously defined in the computer and then determines the validity of your logon.

The default User ID and Password are as follows:

User ID: admin
Password: admin

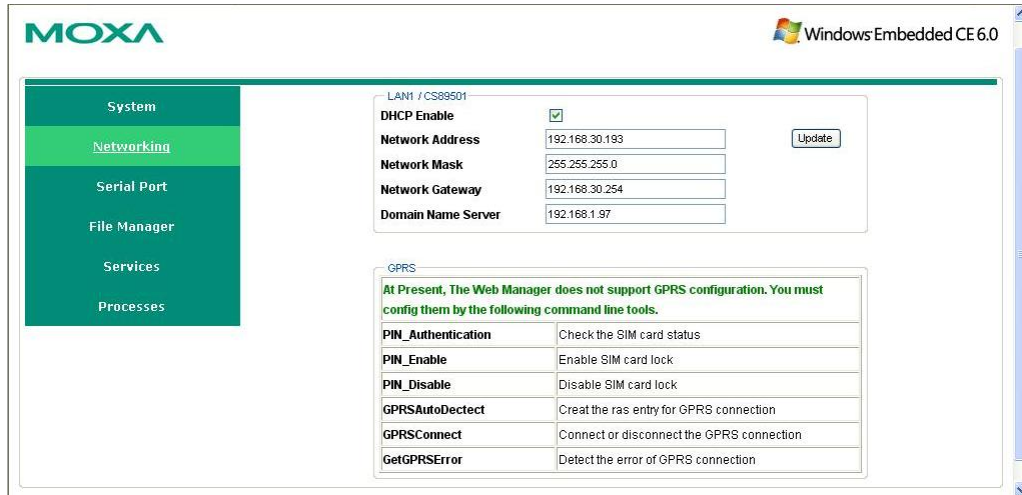
System Information

After you log on successfully, the main page displays the system information of the target computer, including the firmware version of the computer, the CPU system time, and system resources, including main memory and file system usage (RAM and Flash).



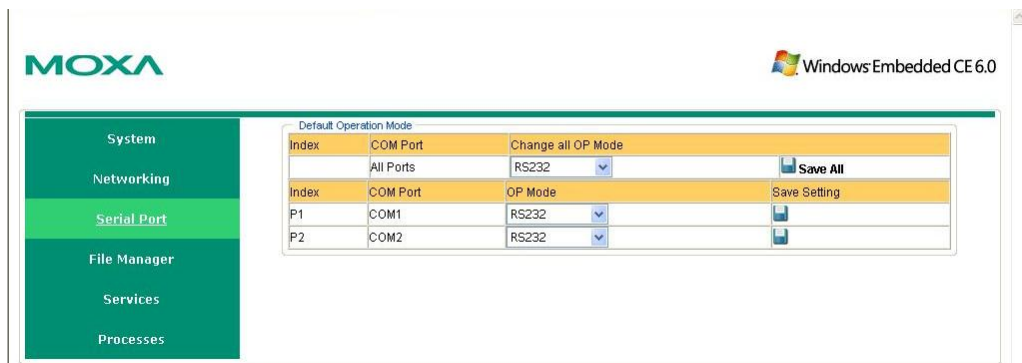
Networking/Server Configuration

The target computer has two network interfaces. To view or change the settings, click the Networking item on the menu bar. After the page loads, enter the relevant details in the corresponding text fields and then click **Update** to activate the changes.



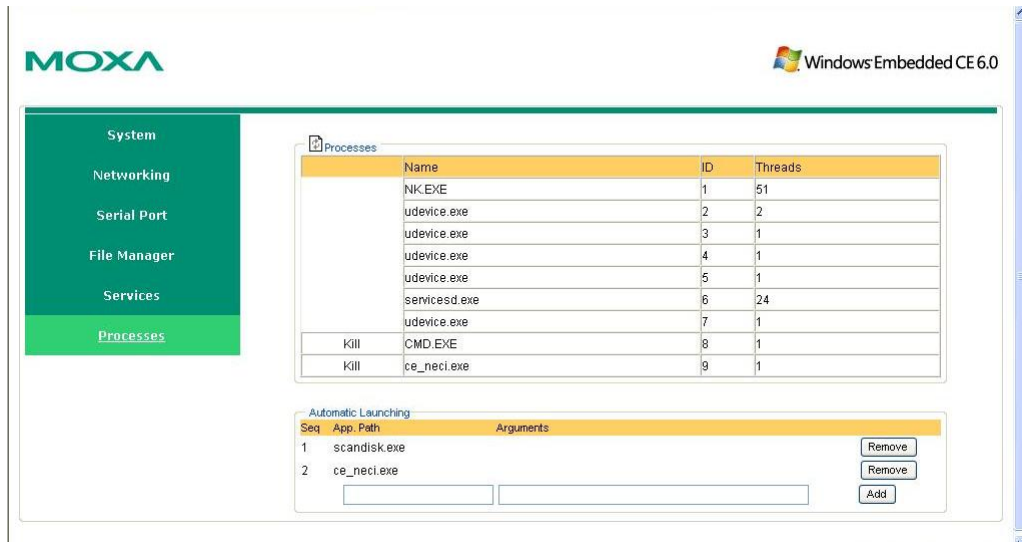
Serial Port Configuration

The target computer has several high-performance serial ports, each of which supports RS-232, RS-422, and RS-485. By default, each port is set for RS-232 data transmission. Each port can be assigned a different serial interface. The updated settings take effect after the system has rebooted, and remain in force until another update is made.



Monitoring and Controlling Processes (Threads)

At runtime, you can monitor and control with the management system. To view current processes, click the **Processes** item on the main menu bar. You can kill a process by clicking the **kill** button next to the process name.



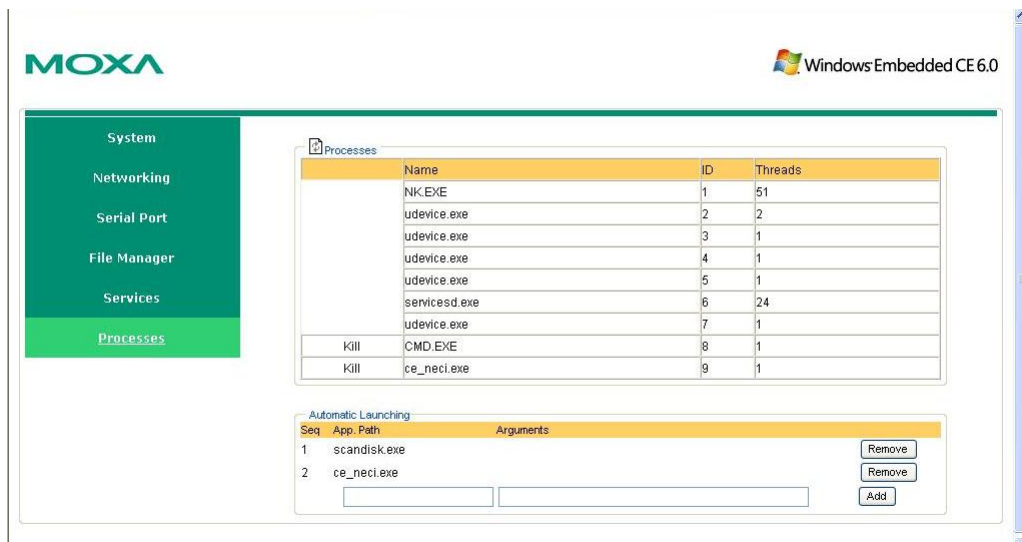
Launching Processes Automatically

To set your application to start on boot up, do the following:

Step 1: Click the **Processes** item on the main menu bar. In the lower part of the page, there is an area marked **Automatic Launching**.

Step 2: Enter the full path of the application in the first text field, and enter its arguments (if there are any) in a separate text field.

Step 3: Click **Add**.

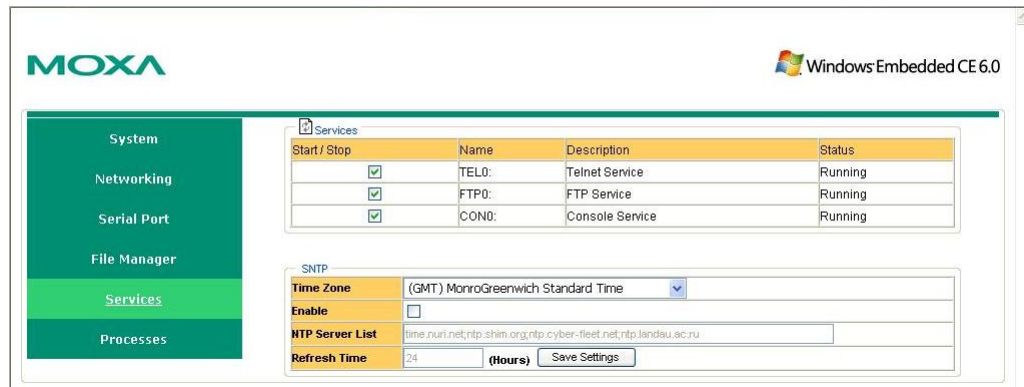


Monitoring and Controlling Services

Some services, such as ftp and telnet daemons, run in the background to provide services for user requests. To monitor and control these services, do the following:

Step 1: Click the **Services** item on the main menu bar. The running services are displayed.

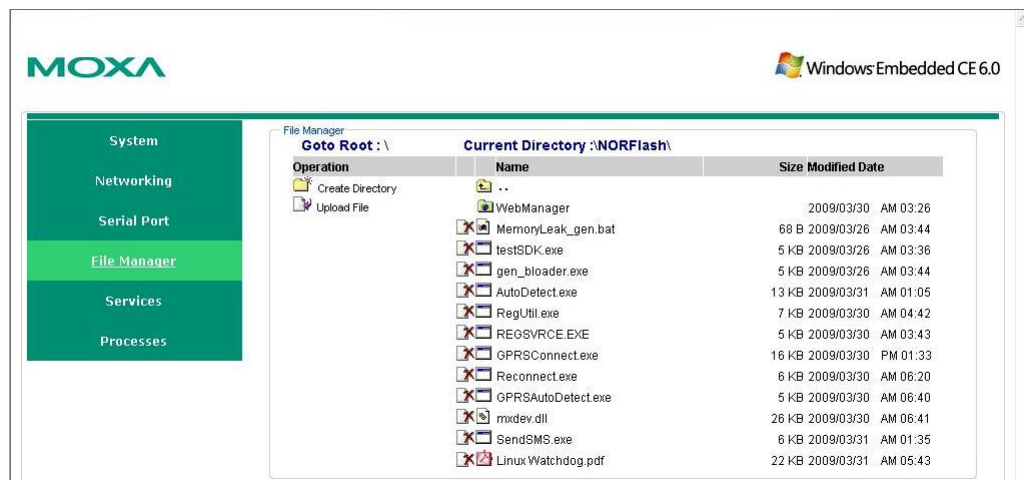
Step 2: Click the relevant check box to toggle a start/stop operation for the desired service.



Binary/Text File Management

PC users enjoy the convenience of using Windows' friendly windows-based file manager to browse, delete, and organize files and directories. Moxa's web-based management system provides the same kind of convenience for managing files on the target computer. Click **File Manager** to view the directory tree of your target computer. The file manager can be used to perform the following operations:

- To browse a child directory, click the name of the directory.
- To delete a file, click the **X** in front of the file icon.
- To create a child directory, click **Create Directory** and then follow the on-screen instructions.
- To refresh the current directory, click **Current Directory** at the top of the page.



In addition, the management system offers a mechanism for uploading files. The mechanism gives you an easy way to transfer files from your workstation to the target computer. For example, after you build an application on the development workstation, you can use this mechanism to upload the application to the current directory of the target computer.

- Step 1:** Click **Upload File**. A browser window pops up.
- Step 2:** From the pop-up browser window, click **Browse** to bring up a local file manager.
- Step 3:** Browse to and select the file that you want to upload and click **Open**.
- Step 4:** Navigate back to the browser window, and click **OK**. The system starts to upload the file.
- Step 5:** After the file is uploaded completely, refresh the page.

A

Firmware Upgrade Procedure

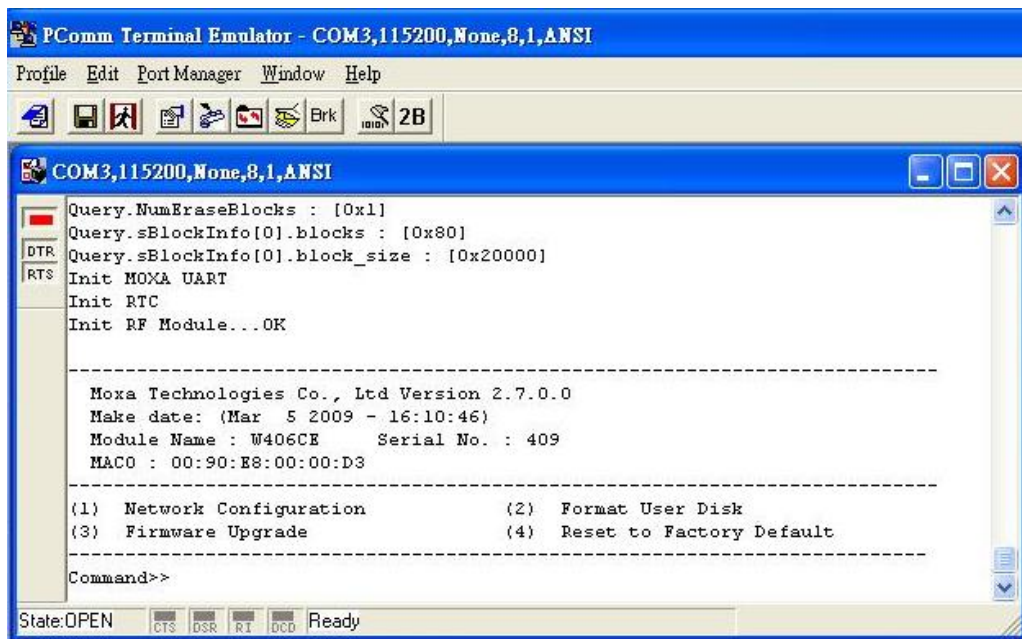
The latest firmware for Moxa's embedded computers can be downloaded from the download center on Moxa's website. For the W406, we provide a solution to upload the firmware file by TFTP, and upgrade the firmware using boot loader utilities. The following steps show how to upgrade the firmware:

Setting up the TFTP server

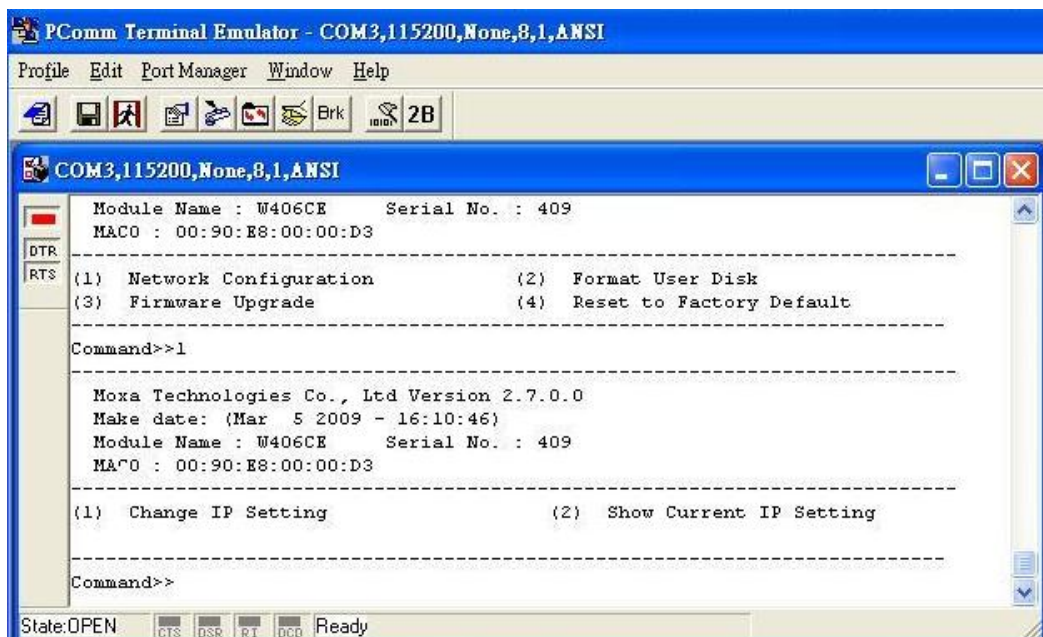
1. Connect LAN of the W406 to your PC using a cross-over Ethernet cable.
2. Download a free TFTP server package from the following site:
<ftp://papa.indstate.edu/winsoc-1/Windows95/Daemons/TFTPD/>
3. Refer the "Help" file in the package for instructions on how to set up the TFTP server.
4. Put the latest firmware image (e.g., W406CE_V1.0_YYMMDDHH.hfm) in the same directory that the TFTP resides.

Configuring the TFTP client on the W406

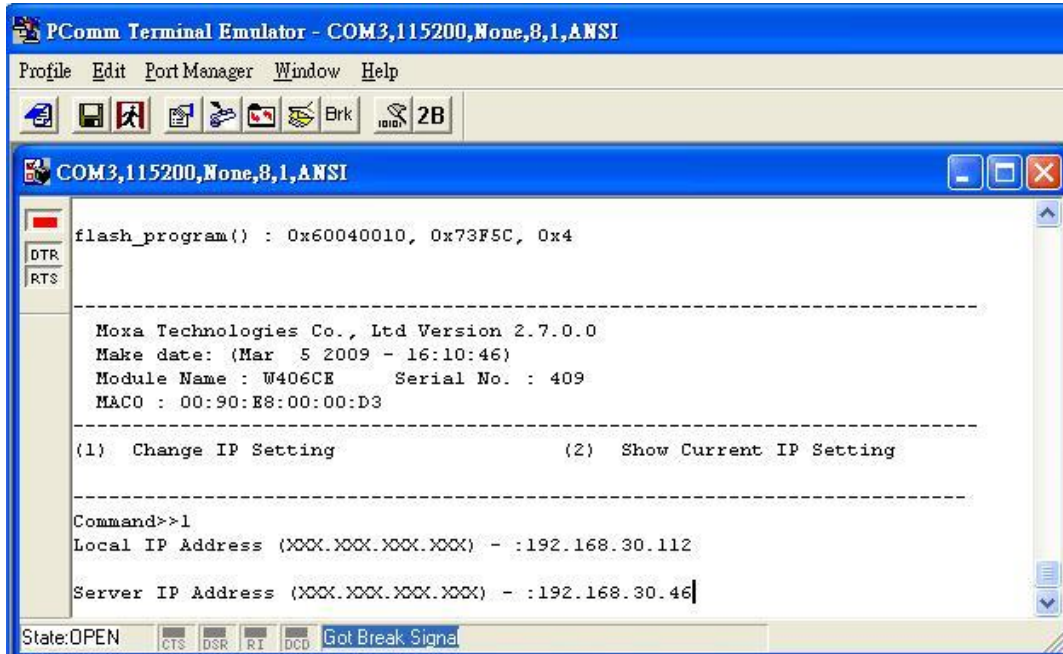
1. Power off the W406.
2. Connect the W406 to your PC with a console port cable.
3. Start a terminal program (such as HyperTerminal or PCComm) with the settings: Baudrate 115200, no hardware flow control, 8 N 1, character set VT100.
4. Hold down the "DEL" key on your PC.
5. Power on the W406 embedded computer. You will be guided to the boot loader utility menu.



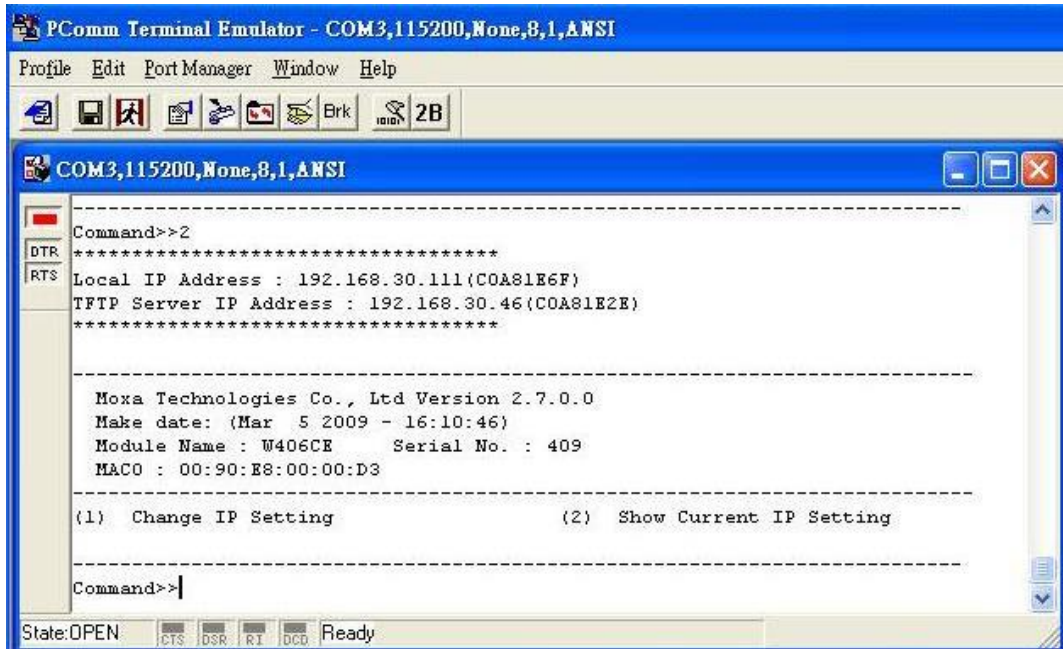
Select (1) Network Configuration.



Select **(1) Change IP Setting**, input Local IP Address of the W406 and Server IP Address (TFTP host IP Address).

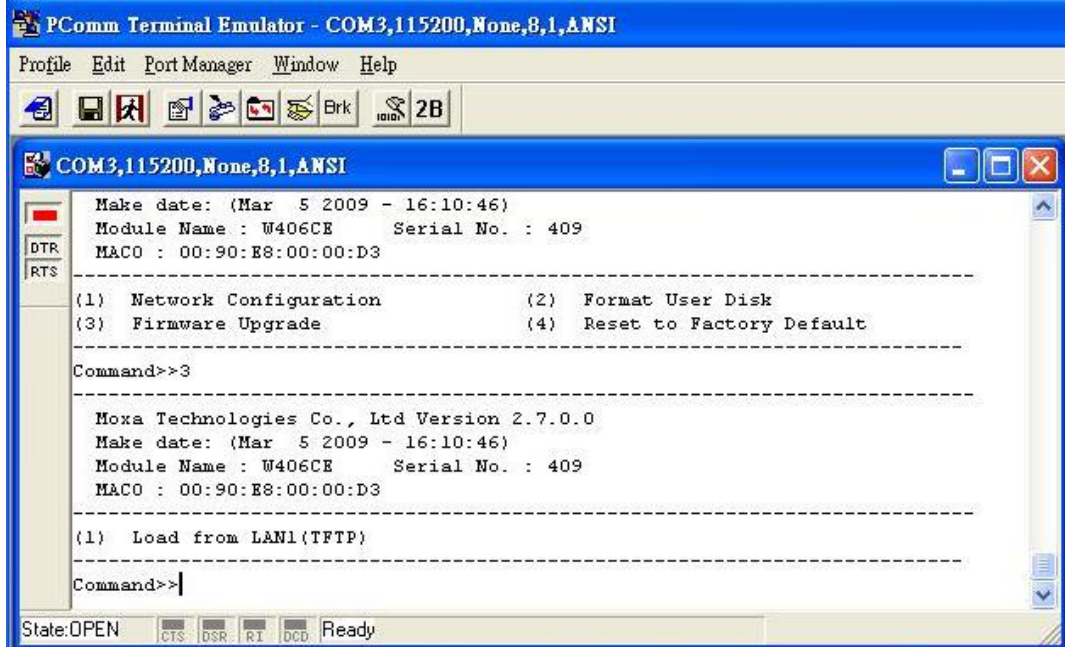


Select **(2) Show Current IP Setting**. You can view the IP setting on the following screen.

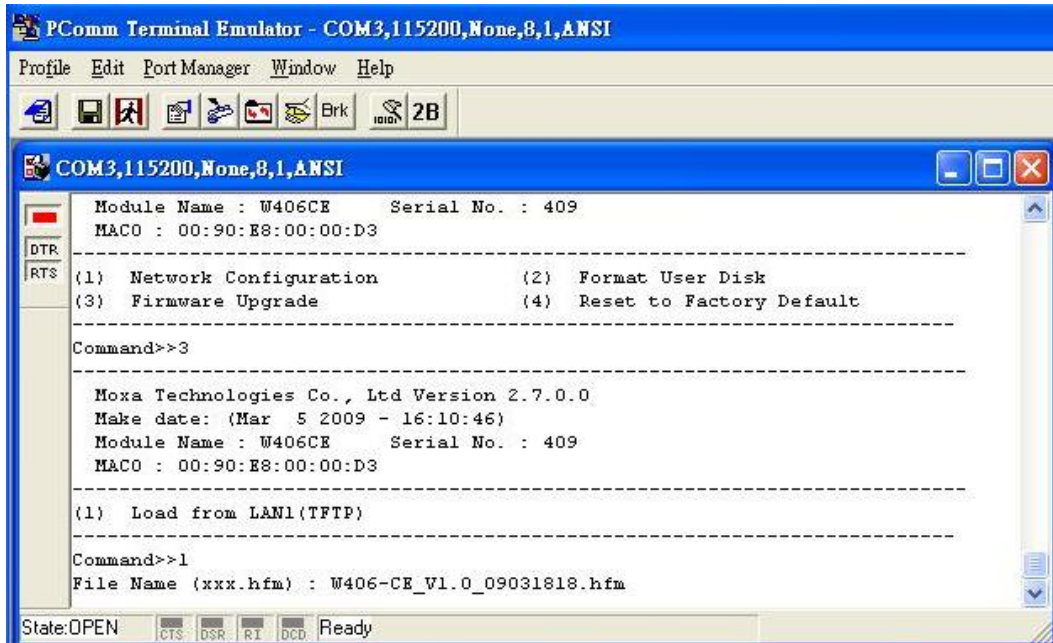


Press **ESC** to go back to the main menu.

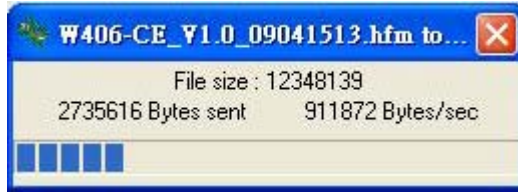
Select **(3) Firmware Upgrade**.



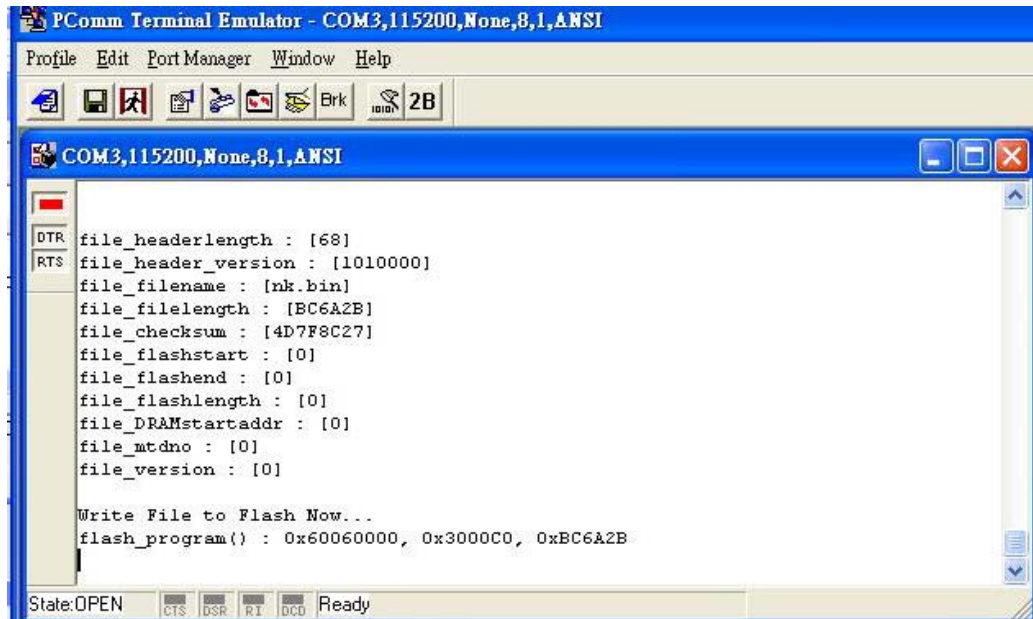
Select **(1) Load from LAN1 (TFTP)**. Input the filename of the firmware for the W406.



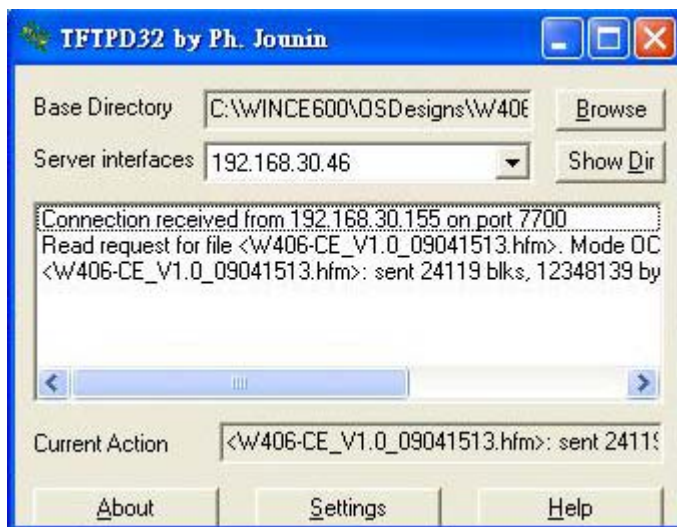
The firmware upgrade will start running. You can view the status from the TFTP screen.



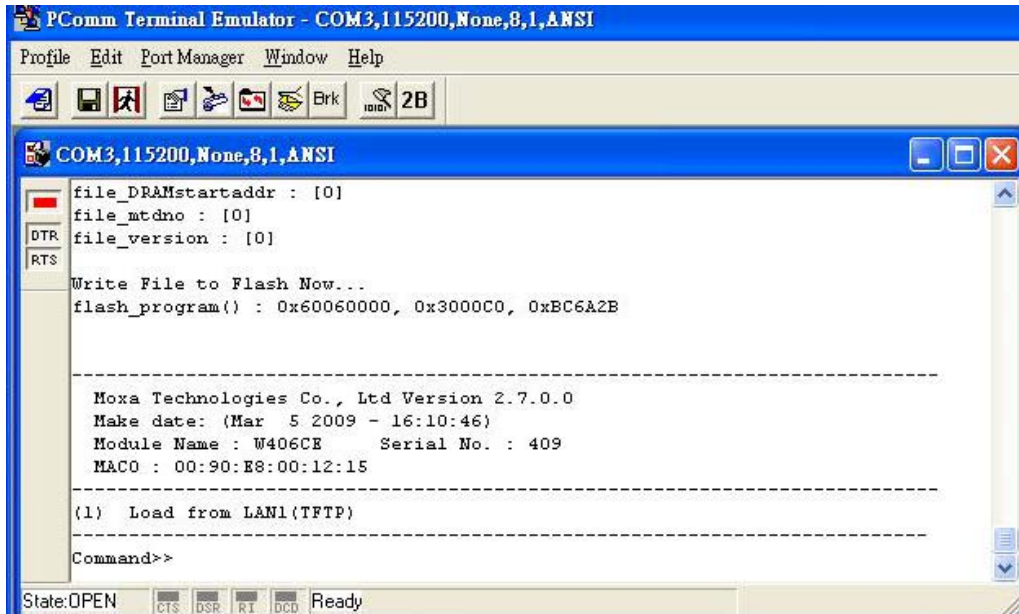
It will take several minutes for the firmware files to be written to your embedded computer. Do not power off your computer!



When finished, you can view it from the TFTP screen.



You can also view it from the terminal screen.



The screenshot shows a PCOMM Terminal Emulator window titled "COM3,115200,None,8,1,ANSI". The window contains the following text:

```
file_DRAMstartaddr : [0]
file_mtdno : [0]
file_version : [0]
Write File to Flash Now...
flash_program() : 0x60060000, 0x3000C0, 0xBC6A2B

-----
Moxa Technologies Co., Ltd Version 2.7.0.0
Make date: (Mar 5 2009 - 16:10:46)
Module Name : W406CE      Serial No. : 409
MACO : 00:90:E8:00:12:15
-----
(1) Load from LAN1(TFTP)
-----
Command>>
```

At the bottom of the window, the status bar shows "State:OPEN" and "Ready".

You may reboot the embedded computer to complete the firmware upgrade.

B

Application Development

The **mxdev** library for C++ and the **mxdevice** library for C Sharp are provided to help users develop their application on the W406 quickly and easily. The complete source code can be found in the sample directory on the software CD.

Developing an application with VS2005

- Open Microsoft® Visual Studio .Net 2005.
- From the **File** menu, choose **New Project**.
- Choose the **Project Type** and then select the **Smart Device Application** as the type of project.
- Fill in the project name and click **OK**.
- Choose **Windows CE** as the target platform.
- Select the desired project type and click **OK**.
- Write your application code.
- From the **Device** toolbar, choose **Windows CE.Net Device**.
- From the **Build** menu, choose **Build Project or Rebuild Project**.
- When you complete your application, upload it to the embedded computer.
- Log on to the embedded computer. At the console prompt, execute it directly if it is a C++ or C# file.

Visual C++ Examples

A library (mxdev.lib) is provided to simplify application development. This library covers APIs for the buzzer, serial, and digital I/O devices.

To link it to your VS2005 developing environment, perform the following steps:

1. From the VS2005 solution explorer, right-click the project and choose **Property**.
2. From the left window, choose **Configuration Properties → Linker → Input**.
3. Append **mxdev.lib** to the text field **Additional Dependencies**.

Before you compile your application, please make sure that the header file and the library file in the SDK directories are as shown below. If any of them is not in the specified directory, find it on the package CD and copy it to the specified directory.

**C:\Program Files\Windows CE
Tools\wce600\W406-CE_V1.0\Include\Armv4i\moxa\devices.h**



ATTENTION

1. The programming examples for Moxa embedded computers are frequently updated. The latest examples can be downloaded at ftp://esource.moxa.com/moxasys/WinCE_Examples/C++.
2. Make sure you do not configure Mobile Equipment Error Message Format (CMEE), or the return value of GPRS connection functions may be incorrect.

C++ Example—Moxa UART Send(RS-232/422/485)

```

BYTE pattern[20] = "1234567890";    // data
HANDLE hSerial = NULL;              // serial port handle
DCB dcb;                            // setting structure of Serial port
WCHAR szPort[10] = L"COM1:";       // define COM1: to be the port of data sent
// Open serial port
hSerial = CreateFile( szPort, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING,
0, NULL);
if ( hSerial != INVALID_HANDLE_VALUE )
{
// Get Port setting structure
GetCommState(hSerial, &dcb);

dcb.BaudRate = 115200;              // set baud rate to 115200 bps
dcb.fRtsControl = RTS_CONTROL_HANDSHAKE; // set Hardware flow control
dcb.ByteSize = 8;                  // data size = 8
dcb.StopBits = ONESTOPBIT;         // stop bit = 1
dcb.Parity = NOPARITY;             // no parity
// Update Port setting structure
SetCommState(hSerial, &dcb);

```

```

// set Mode to RS232
BYTE bMode = RS232_MODE;

DeviceIoControl(hSerial, MOXA_SET_OP_MODE, &bMode, sizeof(BYTE), NULL, 0, &dwReturned,
NULL);

// send pattern
WriteFile( hSerial, pattern, 10 /* send 10 bytes only */, &dwReturned, NULL);

// Close Serial Port Handle
CloseHandle(hSerial);
}

```

C++ Example—Moxa UART Receive(RS-232/422/485)

The following C/C++ code shows a sample application that transmits text data from port COM1 to port COM2 in the RS-232 operation mode. The application opens these two ports, generates a thread to receive data from port COM2, and then executes as a main thread to transmit data to port COM1.

```

BYTE buffer[100]; // data receiving buffer
HANDLE hSerial = NULL; // serial port handle
DCB dcb; // setting structure of Serial port
WCHAR szPort[10] = L"COM2: "; // define COM2: to be the port
of data received
// Open serial port
hSerial = CreateFile( szPort, GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
if ( hSerial != INVALID_HANDLE_VALUE )
{
// Get Port setting structure
GetCommState(hSerial, &dcb);
dcb.BaudRate = 115200; // set baud rate to 115200
bps
dcb.fRtsControl = RTS_CONTROL_HANDSHAKE; // set Hardware flow
control
dcb.ByteSize = 8; // data size = 8
dcb.StopBits = ONESTOPBIT; // stop bit = 1
dcb.Parity = NOPARITY; // no parity

// Update Port setting structure
SetCommState(hSerial, &dcb);

DWORD dwReturned = 0;
// set Mode to RS232
BYTE bMode = RS232_MODE;

```

```
DeviceIoControl(hSerial, MOXA_SET_OP_MODE, &bMode, sizeof(BYTE), NULL, 0, &dwReturned,
NULL);
// send pattern
memset(buffer, 0, sizeof(buffer));
dwReturned = 0;
while ( TRUE )
{
    ReadFile( hSerial, buffer, sizeof(buffer), &dwReturned, NULL);
    // dwReturned is the actually data received byte number
    printf("Received size = %d\n", dwReturned );
    if ( dwReturned > 0 ) {
        printf("data = %s\n", buffer );
        break;
    }
}
// Close Serial Port Handle
CloseHandle(hSerial);
}
```

C++ Example—Buzzer

The embedded computer supports hardware buzzers that applications can use as an alarm for critical errors. You may set the frequency and the duration of the buzzer at the application level by using the APIs. The following example code triggers the buzzer for 50 milliseconds at frequency 2200 Hz.

```
// turn on buzzer
mxbeep_on();

// sleep 200 ms
Sleep(200);

// turn off buzzer
mxbeep_off();
```


C++ Example—Digital I/O

The embedded computer with the support of programmable digital input/output channels provides you with I/O control on other devices. These channels can be accessed at run-time via the following example program.

```
//initial the DIO driver
hDIO = DIO_init();

//Set one bit data of DOUT
bolDout=set_dout(hDIO, index, data);
//get data from DOUT
intDout=get_dout(hDIO,index);
printf("intDout=%d\n",intDout);

//get data from DIN
byteDin=get_din(hDIO);
printf("Din = 0x%x\n",byteDin);

/*Close DIO driver*/
bolClose=DIO_close(hDIO);
if(!bolClose)
{
    printf("DIO_Close fail\n");
}
```

C++ Example—DeleteSMS

```
DWORD dwIndex=1;
int nStatus=-1;

/*Check if the index is entered.*/
if(argc==1)
{
    printf("Usage:\n");
    printf(" DeleteSMS [Message ID]\n");
    printf(" e.g.DeleteSMS 1\n");
    return 0;
}
dwIndex=_wtol(argv[1]);

/*Delete the specified SMS message*/
nStatus=DeleteSMS(dwIndex);

/*Check the status,zero indicates success,nonzero indicates error*/
if(!nStatus)
{
```

```
        printf("DeleteSMS success.\n");
    }
    else
    {
        printf("DeleteSMS fail.\n");
    }
}
```

C++ Example—DetectConnection

The following is an example of getting the connection status.

```
int nStatus=-1;

/*Detect the connection status*/
nStatus=DetectConnection();

if(nStatus==1)
{
    printf("Connection has been established.\n");
}
else
{
    printf("Connection hasn't been established.\n");
}
}
```

C++ Example—GetPINLockStatus

The following is an example of getting the lock status of the SIM card.

```
int nStatus=-1;

nStatus=GetPINLockStatus();

/*Check SIM card status, 0 for ready, 1 for PIN code is needed, 2 for PUK code is
needed.*/
switch(nStatus)
{
    case 0:
        printf("SIM card is ready.\n");
        break;
    case 1:
        printf("PIN code is needed.\n");
        break;
    case 2:
        printf("PUK code is needed.\n");
        break;
    default:

```

```
        printf("SIM is not inserted.\n");
        break;
    }
}
```

C++ Example—GetGPRSError

The following is an example of GetGPRSError. This program executes a series of checks and returns the error code for users to check which procedure is not correct.

```
DWORD dwErrorCode=0x0;
    DWORD dwError=0x0;
    int i=0;
dwError=GetGPRSError();
    if(dwError==ERROR_SUCCESS)
    {
        printf("No Error!!\r\n");
        return 0;
    }
    for(i=0;i<7;i++)
    {
        switch(dwError & (0x1 << i))
        {
            case BAUDRATE_ERROR_COM3:
                printf("BAUDRATE_ERROR_COM3\r\n");
                break;
            case BAUDRATE_ERROR_COM4:
                printf("BAUDRATE_ERROR_COM4\r\n");
                break;
            case FLOWCONTROL_ERROR:
                printf("FLOWCONTROL_ERROR\r\n");
                break;
            case PINCODE_ERROR:
                printf("PINCODE_ERROR\r\n");
                break;
            case TEMPERATURE_ERROR:
                printf("TEMPERATURE_ERROR\r\n");
                break;
            case SIGNALSTRENGTH_ERROR:
                printf("SIGNALSTRENGTH_ERROR\r\n");
                break;
            case RADIOBAND_ERROR:
                printf("RADIOBAND_ERROR\r\n");
                break;
            default :
                break;
        }
    }
}
```

C++ Example—GPRSAutoDetect

The following is an example of GPRSAutoDetect. This program executes a series of detection and creates the entry for connection automatically. Users don't need to create the entry manually. Instead, just execute the function.

```
int nStatus=-1;
    ATOM atom;

    /*Check if autodetect function has been executed*/
    if(atom=GlobalFindAtom(L"AutoDetect"))
    {
        printf("GPRSAutodetect has been started.\n");
        return -1;
    }
    else
    {
        GlobalAddAtom(L"AutoDetect");
    }

    //Detect and create entry for GPRS connection
    nStatus=GPRSAutoDetect();
    if(nStatus==0)
    {
        printf("GPRSAutoDetect success.\n");
    }
    else
    {
        printf("GPRSAutoDetect fail.\n");
    }

    /*Clear the flag*/
    if(atom=GlobalFindAtom(L"AutoDetect"))
    {
        GlobalDeleteAtom(atom);
    }
```

C++ Example—GPRSCONNECT

The following is an example of GPRSCONNECT. This program makes or terminates a connection according to the information stored in the entry which is created by the GPRSAutoDetect function.

```

if(bConnect)
{
    /*If reconnect function has been activated, we don't need to execute rasdial.*/
    if(atom=GlobalFindAtom(L"Reconnect"))
    {
        //printf("Find ATOM\r\n");
        printf("Reconnecting, disable reconnect function if you want to connect manually\r\n");
    }
    else /*Else, we execute rasdial to connect to internet.*/
    {
        RASDial();
        ret=DetectConnection();
        /*
        if(fConnected==TRUE)
        {
            printf("fConnected==TRUE\r\n");
            ret =1;
        }
        */
    }
}
}

```

C++ Example—Memory

The following is an example of retrieving the memory information.

```

MEMORYSTATUS ms = { sizeof(ms) };
GlobalMemoryStatus(&ms);
printf("Free Memory = %d\n", ms.dwAvailPhys);
printf("Toal Memory = %d\n", ms.dwTotalPhys);

```

C++ Example—Message Queue

The following is an example of Message Queue programming. This program creates the main thread first for waiting data and then creates the client thread to receive the collected data.

```

DWORD dwData[CLIENTS];
// Initializing the critical section variable
InitializeCriticalSection(&cs);

```

```
// create the main thread first for waiting data
CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)thread_main, NULL,0, NULL);
Sleep(500);

for ( int i=0; i<CLIENTS; i++) {
    dwData[i] = i+1;
    // create the client thread
    CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)thread_collectdata, &dwData[i],0, NULL);
}

// wait a command to close all threads
WCHAR ch = getwchar();
bQuit = TRUE;
DeleteCriticalSection(&cs);

// make sure all threads were closed
Sleep(3000);
```

C++ Example—PINAuthenticate

```
char *pin="0000";
int nStatus=-1;
nStatus=PINAuthenticate(pin);
if(!nStatus)
{
    printf("PINAuthenticate success.\n");
}
else
{
    printf("PINAuthenticate fail.");
}
```

C++ Example—PUKAuthenticate

```
char *pin="0000";
int nStatus=-1;

nStatus=PUKAuthenticate(pin);
if(!nStatus)
{
    printf("PUKAuthenticate success.\n");
}
else
{
    printf("PUKAuthenticate fail.");
}
```

C++ Example—PINEnable

```
char *pin="0000";
int nStatus=-1;
nStatus=PINEnable(pin);
if(!nStatus)
{
    printf("PINEnable success.\n");
}
else
{
    printf("PINEnable fail.");
}
return 0;
```

C++ Example—PINDisable

```
char *pin="0000";
int nStatus=-1;
nStatus=PINDisable(pin);
if(nStatus==0)
{
    printf("PINDisable success.\n");
}
else
{
    printf("PINDisable fail.");
}
```

C++ Example—RebootDevice

The following is an example of rebooting the device.

```
HANDLE hDevService = CreateFile( L"UCS1:", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);

if ( hDevService != INVALID_HANDLE_VALUE )
{
    // Reboot device
    DeviceIoControl( hDevService, IOCTL_REBOOT_DEVICE, NULL, 0, NULL, 0, NULL, NULL);

    // Close the device service driver
    CloseHandle(hDevService);
}
```

C++ Example—RegistryReadWrite

The following is an example of reading and writing the registry.

```
CreatePartner( L"MyCompany", L"MyLastName", L"MyEmail@some.com", L"MyNation" );
CreatePartner( L"Company11111", L"Name2222", L"Email2222@some.com", L"Nation3333" );

ReadPartner(L"MyCompany");
wprintf(L"\n");
ReadPartner(L"Company11111");
wprintf(L"\n");

DeletePartner(L"MyCompany");
DeletePartner(L"Company11111");
```

C++ Example—ReceiveSMS

The following is an example of receiving an SMS message. This program receives the message from the wireless module from the message ID.

```
PSMS pSMS;
//receive the SMS message of index 1
pSMS=ReceiveSMS(1);
if(pSMS==NULL)
{
    printf("No message!!");
    return 0;
}
//print the message status
printf("Status=%s\r\n",pSMS->chStatus);
//print message date
printf("Date=%s\r\n",pSMS->chDate);
```



```
printf("Time=%s\r\n",pSMS->chTime);
printf("PhoneNumber=%s\r\n",pSMS->chPhoneNumber);
printf("Message=%s\r\n",pSMS->chMessage);
```

C++ Example—SendATCommand

The following is an example of sending an AT command. This program sends a simple “at” AT command to the wireless module and receives a response from the wireless module.

```
char *receive;
receive=SendATCommand("at+cpin?");
printf("receive=%s\r\n",receive);
```

C++ Example—SendSMS

The following is an example of sending an SMS message. This program sends a message and the phone number to the wireless module, and the wireless module sends the message to the specified phone number.

```
//prepare the phone number you want to send SMS message
char* PhoneNumber="0911265306";
//prepare the message you want to send
char * Message="test SMS message";
//send the SMS message by SendsMS API
SendSMS(PhoneNumber,Message);
```

C++ Example—SMSSyncTime

The following is an example of synchronizing the local time via a SMS message.

```
/*Get current system time and copy to strCurrentTime to be a identification string*/
GetLocalTime(&systemtime);
sprintf(strCurretnTime,"%d/%d/%d,%d:%d:%d+%d",systemtime.wYear
,systemtime.wMonth
,systemtime.wDay
,systemtime.wHour
,systemtime.wMinute
,systemtime.wSecond
,systemtime.wMilliseconds);

/*Send SMS with identification string*/
printf("Sending SMS message...");
if(SendSMS(StringConvert(wszPhoneNumber),strCurretnTime)==-1)
{
printf("ERROR\n");
//printf("SendsMS Error\n");
return -1;
}
```

```
else
{
    printf("OK\n");
}
/*Get all messages*/
SendAT("at+cmgf=1\r");
printf("Receiving message...");
receive=SendAT("at^smgl");
if(strlen(receive))
{
    printf("OK\n");
}
else
{
    printf("ERROR\n");
    return -1;
}
str_keyword=(char *)malloc(MAX_PATH);
/*Search the keyword*/
str_keyword=strstr(receive,strCurretnTime);

/*Get the update time*/
strncpy(str_update,str_keyword-23,20);
/*Split the parameters in string*/
sscanf(str_update,"%d/%d/%d,%d:%d:%d+%d",&nYear,&nMonth,&nDay,&nHour,&nMin,&nSec
,&nDiff);
/*Fill the structure with the update parameter*/
systemtime.wYear=2000+nYear;
systemtime.wMonth=nMonth;
systemtime.wDay=nDay;
systemtime.wHour=nHour;
systemtime.wMinute=nMin;
systemtime.wSecond=nSec;
/*Set new time*/
printf("Updating system time...");
if(!SetLocalTime(&systemtime))
{
    printf("ERROR\r\n");
    return -1;
}
else
{
    printf("OK\r\n");
}
```

C++ Example—TCP Server

The following is an example of TCP client/server programming. To build a server program, compile this sample code with library ws2_32.lib.

```
WORD sockVersion;
WSADATA wsaData;
int rVal;
sockVersion = MAKEWORD(1,1);
//start dll
WSAStartup(sockVersion, &wsaData);
//create socket
SOCKET s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if(s == INVALID_SOCKET)
{
    socketError("Failed socket()");
    WSACleanup();
    return SERVER_SOCKET_ERROR;
}
//fill in sockaddr_in struct
SOCKADDR_IN sin;
sin.sin_family = PF_INET;
sin.sin_port = htons(8888);
sin.sin_addr.s_addr = INADDR_ANY;

//bind the socket
rVal = bind(s, (LPSOCKADDR)&sin, sizeof(sin));
if(rVal == SOCKET_ERROR)
{
    socketError("Failed bind()");
    WSACleanup();
    return SERVER_SOCKET_ERROR;
}

//get socket to listen
rVal = listen(s, 2);
if(rVal == SOCKET_ERROR)
{
    socketError("Failed listen()");
    WSACleanup();
    return SERVER_SOCKET_ERROR;
}

//wait for a client
SOCKET client;
client = accept(s, NULL, NULL);
```

```
if(client == INVALID_SOCKET)
{
    socketError("Failed accept()");
    WSACleanup();
    return SERVER_SOCKET_ERROR;
}

char bufcmp[BUFF_SIZE];
for (int m=0; m<BUFF_SIZE; m++)
{
    bufcmp[m] = m %256;
}
char buffer[BUFF_SIZE];
fd_set      readfds;
struct timeval  authtime;
unsigned int nTotal = 0, nIndex=0;
bool bError=FALSE;
while (1)
{
    //char *data = readline(&client);
    memset(buffer,0,sizeof(buffer));
    authtime.tv_usec = 1000L;
    authtime.tv_sec = 0;
    FD_ZERO(&readfds);
    FD_SET(client, &readfds);
    int s = (int)client+1;
    if (select (s, &readfds, NULL, NULL, &authtime) >= 0)
    {
        if (FD_ISSET (client, &readfds))
        {
            //receive data
            rVal = recv(client, buffer, BUFF_SIZE, 0);
        }
    }
}

//close process
closesocket(client);
closesocket(s);
WSACleanup();
```

C++ Example—TCP Client

The following is an example of TCP client/server programming. To build a server or a client program, compile this sample code with library ws2.lib.

```
WORD version;
WSADATA wsaData;
int rVal=0;
version = MAKEWORD(1,1);
WSAStartup(version,(LPWSADATA)&wsaData);
if ( argc != 2 ) {
    printf("syntax: tcp_wince xxx.xxx.xxx.xxx\n",argv[1]);
    return 0;
}
char szIP[100];
memset(szIP,0,sizeof(szIP));
WideCharToMultiByte(CP_ACP, 0, argv[1], wcslen(argv[1]), szIP, sizeof(szIP), 0, 0);
unsigned long ip = dot2ip(szIP);
if (ip == IP_ERROR) {
    printf("Invalid IP address %s!\n",argv[1]);
    return 1;
}

//create the socket
SOCKET theSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if(theSocket == SOCKET_ERROR)
{
    sError("Failed socket()");
    return CS_ERROR;
}

//Fill in the sockaddr_in struct
SOCKADDR_IN serverInfo;

serverInfo.sin_family = PF_INET;
// serverInfo.sin_addr = *((LPIN_ADDR)*hostEntry->h_addr_list);
serverInfo.sin_addr.s_addr = ip;
serverInfo.sin_port = htons(8888);

rVal=connect(theSocket,(LPSOCKADDR)&serverInfo, sizeof(serverInfo));
if(rVal==SOCKET_ERROR)
{
    sError("Failed connect()");
    return CS_ERROR;
}
```

```

//char *buf = "simpleservermessage\n";
//char buf[MAX_PATH];
char *buf = NULL;
buf = new char[256 * 256];

for ( unsigned int i=0; i<256 * 256; i++)
    buf[i] = i % 256;

while (1)
{
    //memset(buf,0,65000);
    //scanf("%s", buf);
    //rVal = send(theSocket, buf, strlen(buf), 0);
    rVal = send(theSocket, buf, 256 * 256, 0);

    //if ( buf[0] == 'q' )
    //    break;
    if(rVal == SOCKET_ERROR)
    {
        sError("Failed send()");
        return CS_ERROR;
    }
    sleep(500);
}
delete buf;

closesocket(theSocket);
//cout << "closing client"<< endl;
WSACleanup();
return CS_OK;

```

C++ Example—WatchDog

```

DWORD dwTimer;
// open the system service driver
HANDLE hWdg = CreateFile( L"\\.\\" + L"UCS1:", GENERIC_READ | GENERIC_WRITE, 0, NULL,
    OPEN_EXISTING, 0, NULL);
dwTimer = 5000;
// starts watchdog timer
DeviceIoControl( hWdg, IOCTL_START_WDG, &dwTimer, sizeof(dwTimer), NULL, 0, NULL,
    NULL);
while ( TRUE )
{
    printf( "Press \"ENTER\" in 5 seconds\n, 'q' to exit");
    WCHAR ch = getwchar();

```

```
if ( ch == (WCHAR)'q' )
{
    break;
}
// refresh watchdog timer
DeviceIoControl( hWdg, IOCTL_REFRESH_WDG, &dwTimer, sizeof(dwTimer), NULL,
0,NULL, NULL);
printf( " *** fed *** \n");
}
// stops watchdog timer
DeviceIoControl( hWdg, IOCTL_STOP_WDG, NULL, 0, NULL, 0, NULL, NULL);
// close the system service driver
CloseHandle(hWdg);
```

Visual C# Examples

A device .Net CF 2.0 class library (mxdevice.dll) is provided to simplify application development with Visual Studio 2005 tools. This library covers the .Net CF Class Library for the buzzer, and digital I/O devices. To link the library with your Visual Studio 2005 project environment, perform the following steps from your Visual Studio 2005 tool:

1. Copy the library file **mxdevice.dll** to any folder on your local disk.

This file can be found on the product CD in the folder **\sdk\dot Net Compact Framework Library**, or the file can be downloaded from the FTP site listed in the NOTE at the bottom of this page.

2. Open the Visual Studio 2005 IDE tool, and then add a new **C# Smart device console application**.
3. Enter the project name and location path.
4. In the **Solution Explorer View**, add **mxdevice.dll** to the reference section.
5. Click **OK**.



ATTENTION

1. The programming examples for Moxa embedded computers are frequently updated. The latest examples can be downloaded from the following FTP site:
ftp://esource.moxa.com/moxasys/WinCE_Examples/C#.
2. You may need to copy the "mxdevice.dll" within the example when you try to execute the program.

C# Example—Moxa UART Send (RS232/422/485)

The following code is a C# sample program for the transmission of data to serial port COM1: This serial port can be configured to support RS-232, RS-422 or RS-485 operation mode.

```
// You must change the Interface first
moxa.w406.SerialInterface.SetComPortInterface("COM1:",
moxa.w406.SerialInterface.SerialMode.RS232_MODE);

// create serial object and Set DCB
System.IO.Ports.SerialPort serial = new System.IO.Ports.SerialPort("COM1:", 921600,
System.IO.Ports.Parity.None, 8, System.IO.Ports.StopBits.One);

// Set Hardware Flow Control
serial.Handshake = System.IO.Ports.Handshake.RequestToSend;
// open the Serial Port
serial.Open();
// make sure the serial port was opened.
if (serial.IsOpen )
{
    string ss = "1234567890";
    for (int i = 0; i < 10; i++)
    {
        System.Console.WriteLine("Send:" + ss);
        // send data
        serial.WriteLine(ss);
        // wait 500 milliseconds
        System.Threading.Thread.Sleep(500);
    }
}
// close serial port
serial.Close();
serial = null;
```

C# Example—Moxa UART Receive (RS232/422/485)

The following code is a C# sample program for the reception of data from serial port COM2: This serial port can be configured to support RS-232, RS-422, or RS-485 operation mode.

```
// You must change the Interface first
moxa.da682.SerialInterface.SetComPortInterface("COM4:",
moxa.da682.SerialInterface.SerialMode.RS232_MODE);

// create serial object and Set DCB
System.IO.Ports.SerialPort serial = new System.IO.Ports.SerialPort("COM4:",
921600, System.IO.Ports.Parity.None, 8, System.IO.Ports.StopBits.One);

// Set Hardware Flow Control
```



```

serial.Handshake = System.IO.Ports.Handshake.RequestToSend;

// define Receive Event to function "serial_DataReceived"
serial.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(serial_DataReceived);

// open the Serial Port
serial.Open();

// wait a input event to quit the program
string s = Console.ReadLine();
serial.Close();
serial = null;

```

C# Example—Buzzer

The embedded computer supports hardware buzzers that applications can use as an alarm for critical errors. You may set the frequency and the duration of the buzzer at the application level by using the APIs. The following sample code triggers the buzzer for 50 milliseconds at a frequency of 2200 Hz.

```

// start buzzer
moxa.ia26x.Buzzer.BeepOn();

// wait 20 milliseconds
System.Threading.Thread.Sleep(20);

// stop buzzer
moxa.ia26x.Buzzer.BeepOff();

```

C# Example—Digital I/O

Digital input/output channels are featured in some models. These channels can be accessed at run-time for control or monitoring using the following example program.

```

int i = 0;

int port = 0, value = 0, value2 = 0;
string sdout = "";
System.Console.WriteLine("Digital In/Out Test Program");
System.Console.WriteLine("\t (0) Exit Program");
System.Console.WriteLine("\t (1) Display DIN");
System.Console.WriteLine("\t (2) Display DOUT");
System.Console.WriteLine("\t (3) Set DOUT value");
System.Console.WriteLine("\t (4) Display both DIN and DOUT");
// read a char.
string sin = System.Console.ReadLine();
int n = int.Parse(sin);
do

```

```
{
    switch (n)
    {
        // if char == '1', display the Din0 ~ Din7
        case 1:
            for (i = 0; i < 8; i++)
            {
                value = moxa.ia26x.DigitalInput.Get(i);
                System.Console.WriteLine("Din{0}={1}", i, value);
            }
            break;
        // if char == '2', display the Dout0 ~ Dout7
        case 2:
            for (i = 0; i < 8; i++)
            {
                value = moxa.ia26x.DigitalOutput.Get(i);
                System.Console.WriteLine("DOut{0}={1}", i, value);
            }
            break;

        // if char == '3', show all (DIN & DOUT)
        case 3:
            // read port Nnumber from console
            System.Console.Write("Input the DOut Port Number (0 ~ 7) = ");
            sdout = System.Console.ReadLine();
            port = int.Parse(sdout);

            // read port value from console
            System.Console.Write("Input the DOut value (0 or 1) = ");
            sdout = System.Console.ReadLine();
            value = int.Parse(sdout);
            // set the new value to the specified port
            if (!moxa.ia26x.DigitalOutput.Set(port, value))
            {
                System.Console.WriteLine("set dout fail!");
                break;
            }
            // display the new Dout value.
            for (i = 0; i < 8; i++)
            {
                value = moxa.ia26x.DigitalOutput.Get(i);
                System.Console.WriteLine("DOut{0}={1}", i, value);
            }
            break;
    }
}
```

```

        // if char == '4', display all of the Din and DOut value.
        case 4:
            for (i = 0; i < 8; i++)
            {
                value = moxa.ia26x.DigitalInput.Get(i);
                value2 = moxa.ia26x.DigitalOutput.Get(i);
                System.Console.WriteLine("DIn{0}={1} DOut{0}={2}", i, value,
value2);
            }
            break;
        }
        sin = System.Console.ReadLine();
        n = int.Parse(sin);
    } while (n != 0);

```

C# Example—DeleteSMS

```

int ret = -1;
UInt32 dwMessageID = 42;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.DeleteSMS(dwMessageID);
if (ret == 0)
{
    Console.WriteLine("DeleteSMS Success.\n");
}
else
{
    Console.WriteLine("DeleteSMS Fail.\n");
}

```

C# Example—DetectConnection

The following is an example of getting the connection status.

```

int ret = -1;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.DetectConnection();
if (ret == 1)
{
    Console.WriteLine("Connection has been established.\n");
}
else
{
    Console.WriteLine("Connection has not been established.\n");
}
}

```

C# Example — GetGPRSError

The following is an example of detecting the error if the connection can't be established.

```
int i = 0, errorCode = 0;
    moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
    errorCode = gprsfun.GetGPRSError();

    for (i = 0; i < 7; i++)
    {
        switch (errorCode & (0x1 << i))
        {
            case moxa.w406.GPRSFunction.BAUDRATE_ERROR_COM3:
                Console.WriteLine("BAUDRATE_ERROR_COM3\n");
                break;
            case moxa.w406.GPRSFunction.BAUDRATE_ERROR_COM4:
                Console.WriteLine("BAUDRATE_ERROR_COM4\n");
                break;
            case moxa.w406.GPRSFunction.FLOWCONTROL_ERROR:
                Console.WriteLine("FLOWCONTROL_ERROR:\n");
                break;
            case moxa.w406.GPRSFunction.PINCODE_ERROR:
                Console.WriteLine("PINCODE_ERROR\n");
                break;
            case moxa.w406.GPRSFunction.TEMPERATURE_ERROR:
                Console.WriteLine("TEMPERATURE_ERROR:\n");
                break;
            case moxa.w406.GPRSFunction.SIGNALSTRENGTH_ERROR:
                Console.WriteLine("SIGNALSTRENGTH_ERROR\n");
                break;
            case moxa.w406.GPRSFunction.RADIOBAND_ERROR:
                Console.WriteLine("RADIOBAND_ERROR\n");
                break;
            default:
                break;
        }
    }
}
```

C# Example — GPRSAutoDetect

The following is an example of detecting the error if the connection can't be established.

```
int ret = 0;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.GPRSAutoDetect();
if (ret == 1)
{
    Console.WriteLine("Configuration Success!!\n");
}
```

C# Example — GPRSConnect

The following is an example of establishing the connection.

```
int ret = 0;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.GPRSConnect(true);
if (ret == 1)
{
    Console.WriteLine("Connect Success");
}
```

C# Example — GPRSGetPINLockStatus

The following is an example of getting the lock status.

```
int ret = -1;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.GetPINLockStatus();
switch (ret)
{
    case 0:
        Console.WriteLine("SIM card is ready.\n");
        break;
    case 1:
        Console.WriteLine("PIN code is needed.\n");
        break;
    case 2:
        Console.WriteLine("PUK code is needed.\n");
        break;
    default:
        Console.WriteLine("GetPINLockStatus fail.\n");
        break;
}
```

C# Example—Message Queue

```
msmq q1 = new msmq(false);
    //System.Console.WriteLine("Queue1 created");
msmq q2 = new msmq(false);
    //System.Console.WriteLine("Queue2 created");
msmq q3 = new msmq(false);
    //System.Console.WriteLine("Queue3 created");
msmq q4 = new msmq(false);
    //System.Console.WriteLine("Queue4 created");
msmq q5 = new msmq(false);
    //System.Console.WriteLine("Queue5 created");
msmq qr = new msmq(true);
    //System.Console.WriteLine("Receive Queue created");

System.Threading.Thread threadr = new
System.Threading.Thread(new .Threading.ThreadStart(qr.listen));
    threadr.Start();
    //System.Console.WriteLine("Receive Queue listening");

System.Threading.Thread thread1 = new System.Threading.Thread(new
System.Threading.ThreadStart(q1.write));
    q1.setparams("queue1:", 30, 100);
    thread1.Start();

System.Threading.Thread thread2 = new System.Threading.Thread(new
System.Threading.ThreadStart(q2.write));
    q2.setparams("queue2:", 30, 110);
    thread2.Start();

System.Threading.Thread thread3 = new System.Threading.Thread(new
System.Threading.ThreadStart(q3.write));
    q3.setparams("queue3:", 30, 120);
    thread3.Start();

System.Threading.Thread thread4 = new System.Threading.Thread(new
System.Threading.ThreadStart(q4.write));
    q4.setparams("queue4:", 30, 90);
    thread4.Start();

System.Threading.Thread thread5 = new System.Threading.Thread(new
System.Threading.ThreadStart(q5.write));
    q5.setparams("queue5:", 30, 80);
    thread5.Start();

System.Console.ReadLine();

q1.close();
```

```
q2.close();
q3.close();
q4.close();
q5.close();

qr.bQuit = true;
System.Threading.Thread.Sleep(2000);
qr.close();
```

C# Example—PINAuthenticate

```
int ret = -1;
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.GetPINLockStatus();
switch (ret)
{
    case 0:
        Console.WriteLine("SIM card is ready.\n");
        break;
    case 1:
        Console.WriteLine("PIN code is needed.\n");
        break;
    case 2:
        Console.WriteLine("PUK code is needed.\n");
        break;
    default:
        Console.WriteLine("GetPINLockStatus fail.\n");
        break;
}
```

C# Example—PUKAuthenticate

```
int ret = -1;
String PINCode = "0000";
byte[] pin = new byte[64];
pin = System.Text.Encoding.ASCII.GetBytes(PINCode);
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.PINAuthenticate(pin);
if (ret==0)
{
    Console.WriteLine("PINAuthenticate success.\n");
}
else
{
    Console.WriteLine("PINAuthenticate fail.");
}
```

C# Example—PINEnable

```
int ret = -1;
    String PINCode = "0000";
    byte[] pin = new byte[64];
    pin = System.Text.Encoding.ASCII.GetBytes(PINCode);
    moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
    ret = gprsfun.PINEnable(pin);
    if (ret == 0)
    {
        Console.WriteLine("PINEnable success.\n");
    }
    else
    {
        Console.WriteLine("PINEnable fail.");
    }
}
```

C# Example—PINDisable

```
mxdevice.SMS sms = new mxdevice.SMS();
    int ret = -1;
    String PINCode = "0000";
    byte[] pin = new byte[64];
    pin = System.Text.Encoding.ASCII.GetBytes(PINCode);
    moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
    ret = gprsfun.PINDisable(pin);
    if (ret == 0)
    {
        Console.WriteLine("PINDisable success.\n");
    }
    else
    {
        Console.WriteLine("PINDisable fail.");
    }
}
```

C# Example — ReceiveSMS


```

mxdevice.SMS sms = new mxdevice.SMS();
//IntPtr lpSMS; // = Marshal.AllocHGlobal(Marshal.SizeOf(sms));
System.Console.WriteLine("Receiving message...");
moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
sms = gprsfun.ReceiveSMS(2);
System.Console.WriteLine("OK\n");
System.Console.WriteLine("Date:" + sms.chDate + "\n");
System.Console.WriteLine("Time:" + sms.chTime + "\n");
System.Console.WriteLine("Status:" + sms.chStatus + "\n");
System.Console.WriteLine("PhoneNumber:" + sms.chPhoneNumber + "\n");
System.Console.WriteLine("Message:" + sms.chMessage + "\n");

```

C# Example—SetInterface

```

//change the interface of COM1: to RS232
        moxa.W406.SerialInterface.SetComPortInterface("COM1:",
moxa.W406.SerialInterface.SerialMode.RS232_MODE);

        //change the interface of COM2: to RS422
        moxa.W406.SerialInterface.SetComPortInterface("COM2:",
moxa.W46.SerialInterface.SerialMode.RS422_MODE);

```

C# Example — SendATCommand

```

String ATCmd = "at+csq\r";
        byte[] strATCmd = new byte[64];
        IntPtr receive;
        strATCmd = System.Text.Encoding.ASCII.GetBytes(ATCmd);

        moxa.w406.GPRSFunction atcmd = new moxa.w406.GPRSFunction();
        receive = atcmd.SendATCommand(strATCmd);
        //String str = Encoding.ASCII.GetString(receive, 0, receive);
        String str = Marshal.PtrToStringUni(receive);
        System.Console.WriteLine("Recv:" + str);

```

C# Example — SendSMS

```
UInt32 ret = 0;
String Phone = "0911265306";
String Message = "Phone Number=0911265306";

byte[] strPhone = new byte[64];
byte[] strMessage = new byte[1024];

strPhone = System.Text.Encoding.ASCII.GetBytes(Phone);
strMessage = System.Text.Encoding.ASCII.GetBytes(Message);

moxa.w406.GPRSFunction gprsfun = new moxa.w406.GPRSFunction();
ret = gprsfun.SendsSMS(strPhone, strMessage);
if (ret == 1)
{
    Console.WriteLine("Connect Success");
}
```

C# Example—WatchDog

```
moxa.W406.Watchdog wdg = new moxa.W406.Watchdog();

// start watchdog timer with 5000 milliseconds (5 seconds).
wdg.start(5000);

System.Console.WriteLine("Enter 'q' to quit the watchdog test..");
while (true)
{
    System.Console.WriteLine("Press Enter in 5 seconds..");
    string s = System.Console.ReadLine();
    if (s.StartsWith("q") || s.StartsWith("Q"))
    {
        break;
    }

    // refresh watchdog timer
    wdg.refresh();
}

// stop watchdog timer
wdg.stop();
```