



usbLan.dll(for .net platform) instruction

CONTENTS

1.	COMMONMODULE	2
2.	SWITCHMODULE	3
3.	ATTENUATORMODULE	3
4.	PHASESHIFTERMEDIATE	4
5.	MULTICHANNELATTENUATORMODULE	4
6.	MULTICHANNELPHASESHIFTERMEDIATE	6

1. commonModule

Common module for all USB+LAN products.

1.1 **string[]** getCOMnames()

Get all COM ports for this PC.

Return the COM ports.

1.2 **SerialPort** openConnectUSB (**string** comPort,**int** baudrate)

Connect the serial port with baudrate. StopBits = One, DataBits = 8, Parity = None.

Return the serial port of COMx.

Notes: The baudrate is 115200 by default or 9600, please refer to the product manual.

1.3 **void** closeConnectUSB(**SerialPort** usbPort)

Close the connection for serial port.

1.4 **string** getProductInformationUSB(**SerialPort** usbPort)

Get the product information by serial port.

Return: PN, SN, manufacturer, version.

1.5 **void** resetDeviceUSB(**SerialPort** usbPort)

Restart the product. No characters returned.

1.6 **string** getMAC_USB(**SerialPort** usbPort)

Query the MAC of device, return the MAC.

1.7 **string** setBaudrateUSB(**SerialPort** usbPort,**int** baudrate)

Set the baudrate for the serial port, return 'Baud:115200' or 'Baud:9600'.

This function will take effect after restart.

Parameters: baudrate, only be 0 or 1. 0-9600; 1-115200.

1.8 **string** queryErrorDescription(**string** errorCode)

Find the error description by error code.

Notes:The error code can be found on the product return string, and it will only contain: E1,E2,E3,E4,E5.

1.9 **string[]** getLocalIPs()

Get the IP address for all IPv4 in this PC.

Return IP address.

1.10 **Socket** openConnectTCP(**string** localIP, **string** productIP,**int** productPort)

Open the TCP connection.Return the handle of a TCP connection socket.

The default product IP address is: 192.168.1.225, and the default port is: 5100.

You can use 'USB Control Panel' to query or modify it. The range of port is: 2000-65535.

1.11 **void** closeConnectTCP(**Socket** TCPConnect)

Close the TCP connection.

1.12 **string** getProductInformationTCP(**Socket** TCPConnect)

Get the product information by TCP connect.

Return: PN, SN, manufacturer, version.

1.13 **void** resetDeviceTCP(**Socket** tcpConnect)

Restart the product. No characters returned.

1.14 **string** getMAC_TCP(**Socket** tcpConnect)

Query the MAC of this product, return the MAC.

2. switchModule

Modules for Programmable Switch.

- 2.1 **string** openChannelUSB(**SerialPort** usbPort, **int** channelNumber)
Open the channel by channel number. Return strings 'SOx'.
- 2.2 **string** openChannelTCP(**Socket** TCPConnect, **int** channelNumber)
Open the channel by channel number. Return strings 'SOx'.
- 2.3 **string** cycleChannelUSB(**SerialPort** usbPort, **int** channelNumber, **double** period, **double** width)
Automatically open the channel with period and pulse width. Return 'SLx'.
Range of period: 0.1-2000ms, Range of pulse width: 0.1-2000ms.
- 2.4 **string** cycleChannelTCP(**Socket** TCPConnect, **int** channelNumber, **double** period, **double** width)
Automatically open the channel with period and pulse width. Return 'SLx'.
Range of period: 0.1-2000ms, Range of pulse width: 0.1-2000ms.
- 2.5 **string** cycleMultichannelUSB(**SerialPort** usbPort, **int[]** channelNumbers, **double** width)
Looping for channels x1, x2,... with pulse width, return 'MLx1'.
Range of pulse width: 0.1-2000ms.
- 2.6 **string** cycleMultichannelTCP(**Socket** TCPConnect, **int[]** channelNumbers, **double** width)
Looping for channels x1, x2,... with pulse width, return 'MLx1'.
Range of pulse width: 0.1-2000ms.
- 2.7 **string** closeChannelUSB(**SerialPort** usbPort)
Close all channels. Stop loop, close all channels. Return 'SC0'.
- 2.8 **string** closeChannelTCP(**Socket** TCPConnect)
Close all channels. Stop loop, close all channels. Return 'SC0'.

3. AttenuatorModule

Modules for Programmable Attenuator.

- 3.1 **string** setAttenuationStepUSB(**SerialPort** usbPort, **int** step)
Set attenuation value by attenuation step.
Return the current attenuation value(step*stepsize).
Range of attenuation step: 0-(2^controlBits-1).
The actual attenuation value is **step*stepsize + insertion loss**.
- 3.2 **string** setAttenuationStepTCP(**Socket** TCPConnect, **int** step)
Set attenuation value by attenuation step.
Return the current attenuation value(step*stepsize).
Range of attenuation step: 0-(2^controlBits-1).
The actual attenuation value is **step*stepsize + insertion loss**.
- 3.3 **string** getCurrentSettingsUSB(**SerialPort** usbPort)
Get current attenuation value.
Return: Attenuation step and attenuation value.
- 3.4 **string** getCurrentSettingsTCP(**Socket** TCPConnect)
Get current attenuation value.
Return: Attenuation step and attenuation value.
- 3.5 **string** getStepsizeUSB(**SerialPort** usbPort)
Get stepsize of attenuator. Return stepsize.
- 3.6 **string** getStepsizeTCP(**Socket** TCPConnect)

Get stepsize of attenuator.Return stepsize.

4. PhaseShifterModule

Modules for Programmable Phase Shifter.

- 4.1 **string** **setPhaseValueStepUSB**(SerialPort usbPort, int step)
Set phase value by step. Return the current phase value(step*stepsize).
- 4.2 **string** **setPhaseValueStepTCP**(Socket TCPConnect, int step)
Set phase value by step. Return the current phase value(step*stepsize).
- 4.3 **string** **getCurrentSettingsUSB**(SerialPort usbPort)
Get current phase value. Return: Phase value step and phase value.
- 4.4 **string** **getCurrentSettingsTCP**(Socket TCPConnect)
Get current phase value. Return: Phase value step and phase value.
- 4.5 **string** **getStepsizeUSB**(SerialPort usbPort)
Get stepsize of phase shifter.Return stepsize.
- 4.6 **string** **getStepsizeTCP**(Socket TCPConnect)
Get stepsize of phase shifter.Return stepsize.

5. multichannelAttenuatorModule

Modules for Multichannel Programmable Attenuator.

- 5.1 **string** **getCycleSettingsUSB**(SerialPort usbPort,int channelNumber)
Query the current loop settings for channel.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 5.2 **string** **getCycleSettingsTCP**(Socket TCPConnect, int channelNumber)
Query the current loop settings for channel.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 5.3 **string** **getProtectionStatusUSB**(SerialPort usbPort)
Query the external protection status.
Return: Protect:1--Protected. Protect:0--Unprotected.
- 5.4 **string** **getProtectionStatusTCP**(Socket TCPConnect)
Query the external protection status.
Return: Protect:1--Protected. Protect:0--Unprotected.
- 5.5 **string** **getStatusUSB**(SerialPort usbPort, int channelNumber)
Query the channel status.
Return 0 or 1. And 0 means fixed attenuation, 1 means loop mode.
- 5.6 **string** **getStatusTCP**(Socket TCPConnect, int channelNumber)
Query the channel status.
Return 1 or 0. And 0 means fixed attenuation, 1 means loop mode.
- 5.7 **string** **getStepsizeUSB**(SerialPort usbPort, int channelNumber)
Query the stepsize of channel.
- 5.8 **string** **getStepsizeTCP**(Socket TCPConnect, int channelNumber)
Query the stepsize of channel.
- 5.9 **string** **getFixedStepUSB**(SerialPort usbPort, int channelNumber)
Query the current fixed step of channel.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 5.10 **string** **getFixedStepTCP**(Socket TCPConnect, int channelNumber)

Query the current fixed step of channel.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

5.11 **string** getControlBitsUSB(**SerialPort** usbPort, **int** channelNumber)

Query the control bits of channel.

5.12 **string** getControlBitsTCP(**Socket** TCPConnect, **int** channelNumber)

Query the control bits of channel.

5.13 **string** setCycleUSB(**SerialPort** usbPort, **int** channelNumber, **int** startStep, **int** stopStep, **int** multiStepsize, **double** wellTime, **int** loopCount)

Set the loop parameters for channel. Return 'SetL:p1-p2-p3-p4-p5-p6'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

Attention: This command has to follow up with the command 'AT:StartLoop' in order to be execute.

5.14 **string** setCycleTCP(**Socket** TCPConnect, **int** channelNumber, **int** startStep, **int** stopStep, **int** multiStepsize, **double** dwellTime, **int** loopCount)

Set the loop parameters for each channel. Return 'SetL:p1-p2-p3-p4-p5-p6'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

Attention: This command has to follow up with the command 'AT:StartLoop' in order to be execute.

5.15 **string** startCycleUSB(**SerialPort** usbPort, **bool** channel1, **bool** channel2, **bool** channel3, **bool** channel4, **bool** channel5, **bool** channel6, **bool** channel7, **bool** channel8)

Execute the loop settings of all channels. Return 'StartL:...'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

5.16 **string** startCycleTCP(**Socket** TCPConnect, **bool** channel1, **bool** channel2, **bool** channel3, **bool** channel4, **bool** channel5, **bool** channel6, **bool** channel7, **bool** channel8)

Execute the loop settings of all channels. Return 'StartL:...'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

5.17 **string** setFixedStepUSB(**SerialPort** usbPort, **int** channelNumber, **int** fixedStep)

Set the fixed attenuation step as y for each channel. Return 'SetS:x-y'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

Attention: This command has to follow up with the command 'AT:StartStep' in order to be execute.

The actual attenuation value is step*stepsize + insertion loss.

5.18 **string** setFixedStepTCP(**Socket** TCPConnect, **int** channelNumber, **int** fixedStep)

Set the fixed attenuation step y for channel. Return 'SetS:x-y'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

Attention: This command has to follow up with the command 'AT:StartStep' in order to be execute.

The actual attenuation value is step*stepsize + insertion loss.

5.19 **string** startFixedUSB(**SerialPort** usbPort, **bool** channel1, **bool** channel2, **bool** channel3, **bool** channel4, **bool** channel5, **bool** channel6, **bool** channel7, **bool** channel8)

Execute the fixed settings of all channels. Return 'StartS:...'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

5.20 **string** startFixedTCP(**Socket** TCPConnect, **bool** channel1, **bool** channel2, **bool** channel3, **bool**

channel4, bool channel5, bool channel6, bool channel7, bool channel8)

Execute the fixed settings of all channels.Return 'StartS:...'.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

5.21 **string** stopCycleUSB(**SerialPort** usbPort, **bool** channel1, **bool** channel2, **bool** channel3, **bool** channel4, **bool** channel5, **bool** channel6, **bool** channel7, **bool** channel8)

Serial Port: Stop the loop for all channels.

If the external trigger is ON, this command is invalid and return 'Protect:1', else return 'StopL:...'.

5.22 `string stopCycleTCP(SocketTCPConnect, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)`

Stop the loop of all channels.

If the external trigger is ON, this command is invalid and return 'Protect:1', else return 'StopL:...'.

6. multichannelPhaseShifterModule

Modules for Multichannel Programmable Phase Shifter.

6.1 string getCycleSettingsUSB(SerialPort usbPort,int channelNumber)

Query the current loop settings for each channel.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

6.2 string getCycleSettingsTCP(Socket TCPConnect, int channelNumber)

Query the current loop settings for each channel.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

6.3 string getProtectionStatusUSB(SerialPort usbPort)

Query the external protection status.

Return: Protect:1--Protected. Protect:0--Unprotected.

6.4 string getProtectionStatusTCP(Socket TCPConnect)

Query the external protection status.

Return: Protect:1--Protected. Protect:0--Unprotected.

6.5 string getStatusUSB(SerialPort usbPort, int channelNumber)

Query the channel status.

Return 0 or 1. And 0 means fixed phase value, 1 means cycle mode.

6.6 string getStatusTCP(Socket TCPConnect, int channelNumber)

Query the channel status.

Return 1 or 0. And 0 means fixed phae value, 1 means cycle mode.

6.7 string getStepsizeUSB(SerialPort usbPort, int channelNumber)

Query the stepsize for channel.

6.8 string getStepsizeTCP(Socket TCPConnect, int channelNumber)

Query the stepsize channel.

6.9 string getFixedStepUSB(SerialPort usbPort, int channelNumber)

Query the current fixed step for channel.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

6.10 string getFixedStepTCP(Socket TCPConnect, int channelNumber)

Query the current fixed step for channel.

If the external trigger is ON, this command is invalid and return 'Protect:1'.

- 6.11 string getControlBitsUSB(SerialPort usbPort, int channelNumber)**
Query the control bits for channel.
- 6.12 string getControlBitsTCP(Socket TCPConnect, int channelNumber)**
Query the control bits for channel.
- 6.13 string setCycleUSB(SerialPort usbPort, int channelNumber, int startStep, int stopStep, int multiStepsize, double dwellTime, int loopCount)**
Set the cycle parameters for channel. Return 'SetL:p1-p2-p3-p4-p5-p6'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
Attention: This command has to follow up with the command 'PS:StartLoop' in order to be execute.
- 6.14 string setCycleTCP(Socket TCPConnect, int channelNumber, int startStep, int stopStep, int multiStepsize, double dwellTime, int loopCount)**
Set the cycle parameters for channel. Return 'SetL:p1-p2-p3-p4-p5-p6'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
Attention: This command has to follow up with the command 'PS:StartLoop' in order to be execute.
- 6.15 string startCycleUSB(SerialPort usbPort, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)**
Execute the loop settings of all channels. Return 'StartL:...'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 6.16 string startCycleTCP(Socket TCPConnect, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)**
Execute the cycle settings of all channels. Return 'StartL:...'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 6.17 string setFixedStepUSB(SerialPort usbPort, int channelNumber, int fixedStep)**
Set the fixed phase step for channel. Return 'SetS:x-y'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
Attention: This command has to follow up with the command 'PS:StartStep' in order to be execute.
- 6.18 string setFixedStepTCP(Socket TCPConnect, int channelNumber, int fixedStep)**
Set the fixed phase step as y of channel. Return 'SetS:x-y'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
Attention: This command has to follow up with the command 'PS:StartStep' in order to be execute.
- 6.19 string startFixedUSB(SerialPort usbPort, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)**
Execute the fixed settings of all channels. Return 'StartS:...'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 6.20 string startFixedTCP(Socket TCPConnect, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)**
Execute the fixed settings of all channels. Return 'StartS:...'.
If the external trigger is ON, this command is invalid and return 'Protect:1'.
- 6.21 string stopCycleUSB(SerialPort usbPort, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)**

channel4, bool channel5, bool channel6, bool channel7, bool channel8)

Stop the loop for all channels.

If the external trigger is ON, this command is invalid and return 'Protect:1', else return 'StopL:...'.

6.22 `string stopCycleTCP(Socket TCPConnect, bool channel1, bool channel2, bool channel3, bool channel4, bool channel5, bool channel6, bool channel7, bool channel8)`

Stop the loop for all channels.

If the external trigger is ON, this command is invalid and return 'Protect:1', else return 'StopL:...'.