PROGRAMMING MANUAL

Mechanical Switch Systems

RC Series | ZTRC Series

testsolutions@minicircuits.com | www.minicircuits.com

Contents

1. Overview

This programming manual is intended for customers wishing to create their own interface for Mini-Circuits' USB & Ethernet controlled, mechanical switch systems. The contents apply to:

- RC series mechanical switch boxes
- ZTRC series mechanical switch racks

The full software and documentation package including a GUI program, DLL files, user guide and programming examples is available for download from the Mini-Circuits website at:

https://www.minicircuits.com/softwaredownload/rfswitchcontroller.html

For details and specifications on the individual models, please see:

https://www.minicircuits.com/WebStore/RF-Mechanical-Compact-Switch.html

Files made available for download from the Mini-Circuits website are subject to Mini-Circuits' terms of use which are available on the website.

1.1. Control Methods

Communication with the system can use any of the following approaches:

1. Using SSH, HTTP or Telnet communication via an Ethernet TCP / IP connection (see Ethernet Control API), which is largely independent of the operating system

   Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

2. Using the provided API DLL files (.Net or ActiveX objects) for USB control on Microsoft Windows operating systems (see USB Control API for Microsoft Windows)
3. Using USB interrupt codes for direct programming on Linux operating systems (see USB Control via Direct Programming (Linux))

1.2. Programming Examples

Mini-Circuits provides examples for a range of programming environments and connection methods, these can be downloaded from our website at:

https://www.minicircuits.com/WebStore/pte_example_download.html

Mini-Circuits' Ethernet & USB controlled devices are designed to implement similar control interfaces, so it is usually the case that an example written for one product family can be adapted easily for use with another.

Please contact Mini-Circuit's application support team if an example is not available for the environment of interest.

1.3. Support Contacts

We are here to support you every step of the way. For technical support and assistance, please contact us at the email address below or refer to our website for your local support:

testsolutions@minicircuits.com

www.minicircuits.com/contact/worldwide_tech_support.html

2. Ethernet Control API

Control of the device via Ethernet TCP / IP networks involves sending the SCPI commands / queries detailed below via HTTP or Telnet. SSH is also available as an option on some models for secure communication (please contact testsolutions@minicircuits.com for details).

In addition, UDP is supported for discovering available systems on the network.

These protocols are widely supported and straightforward to implement in most programming environments. Any Internet browser can be used as a console / tester for HTTP control by typing the full URL directly into the address bar. Telnet and SSH are supported by a number of console applications, including PuTTY.

2.1. Configuring Ethernet Settings

The device's Ethernet IP settings can be configured using either the USB or Ethernet connections. Refer to the SCPI - Ethernet Configuration Commands section for details.

Configure all required parameters and then use the Update Ethernet Settings command to reset the controller and restart with the updated configuration. If connected via Ethernet, all subsequent commands / queries to the device will need to be attempted using the new Ethernet configuration. If a connection cannot be established, it may indicate an invalid configuration was created (for example a static IP address which clashes with another device on the network). The Ethernet settings can always be overwritten by connecting to the system using the USB connection.

2.2. Default Ethernet Configuration

Mini-Circuits' products ship with DHCP enabled by default so in most cases the device should be assigned a dynamic IP address when connected to the network. The assigned IP can be identified from the network administrator, by using UDP to broadcast a query, or by using the USB connection. The latter 2 options can be accomplished using the Mini-Circuits GUI, or via a custom program. Please contact Mini-Circuits for support.

Once a valid IP address has been assigned and identified it can be re-configured in multiple ways:

1. Using the Windows GUI when connected by USB
2. Using the API when connected by USB or Ethernet
3. Using Mini-Circuits' HTML configuration tool when connected by Ethernet

2.3. Link-Local / Auto IP Address

A default "link-local" auto IP address will be assumed for models with firmware F0 or later when DHCP is enabled but the device does not receive a valid response from a DHCP server. This could be the case on networks with no DHCP server or when the device is connected directly via an Ethernet cable to a PC instead of via a network.

The default static / link-local IP address for all Mini-Circuits devices with the relevant firmware is 169.254.10.10.

The default auto-IP features provide a method to implement a static IP configuration, without first relying on DHCP, or resorting to the USB connection. The process would be:

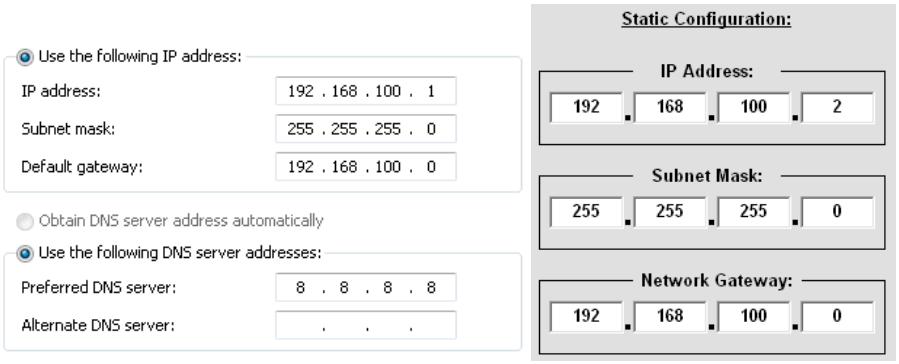
1. Connect the device directly to a PC using the Ethernet cable
2. No DHCP response will be received from the PC so the device will assume the default auto-IP
3. Connect to the device on 169.254.10.10
4. Disable DHCP, set the required static IP configuration and reset the device
5. Reconnect using the updated IP configuration

2.4. Direct Ethernet Cable Connection to PC

It may be necessary to set a compatible TCP / IP configuration on a PC in order to establish a direct connection by Ethernet cable between the device and PC rather than via a network. The values can be chosen arbitrarily as long as a valid and compatible range is applied to both PC and Mini-Circuits device. The key points are:

1. Set different IP addresses on the same subnet for the PC and device
2. Set another different IP address on the same subnet as the network gateway IP, using the same value on both PC and Mini-Circuits device
3. Set the same subnet mask on PC and device (ensuring the mask allows the above IP range)

An example of a working configuration is shown below, with the PC settings on the left and the static IP settings for the Mini-Circuits device on the right.

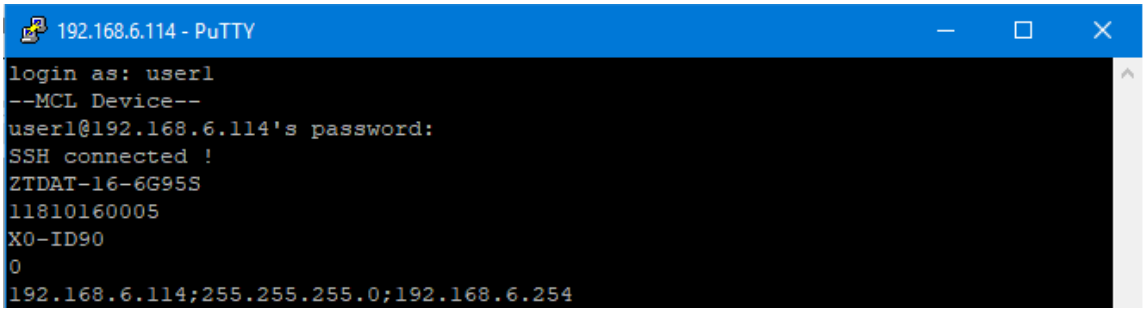


2.5. SSH Communication

SSH allows secure communication with the device, using the configured SSH port (default is port 22) and password. The default username is `ssh_user`.

Note: SSH communication is not supported as standard on all models, please contact testsolutions@minicircuits.com for details.

SSH is widely supported and can be implemented in most programming environments. Alternatively, a client such as PuTTY can be used as a console to quickly establish an SSH connection and control the system.



2.6. HTTP Communication

HTTP Get / Post are supported. The basic format of the HTTP command is:

```
http://ADDRESS:PORT/PWD;COMMAND
```

Where:

- http:// is required
- ADDRESS = IP address (required)
- PORT = TCP/IP port (can be omitted if port 80 is used)
- PWD = Password (can be omitted if password security is not enabled)
- COMMAND = Command / query to send to the device

Example 1:

```
http://192.168.100.100:800/PWD=123;SETA=1
```

- The switch has IP address 192.168.100.100 and uses port 800
- Password security is enabled and set to "123"
- The command is to set switch A to state 1

Example 2:

```
http://10.10.10.10/SETB=0
```

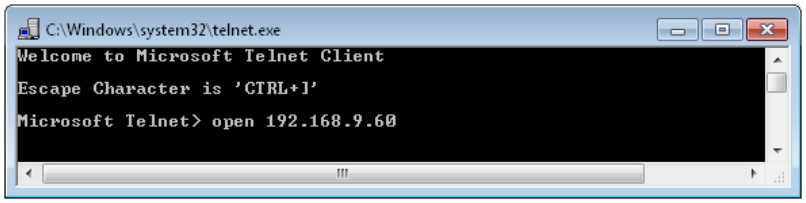
- The switch has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled
- The command is to set switch B to state 0

2.7. Telnet Communication

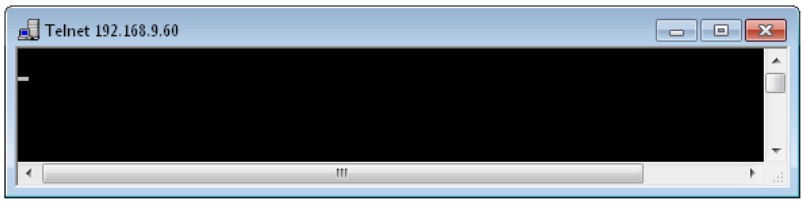
Communication is started by creating a Telnet connection to the device's IP address. On successful connection the "line feed" character will be returned. If the system has a password enabled, this must be sent as the first command after connection.

Each command must be terminated with the carriage return and line-feed characters (\r\n). Responses will be similarly terminated. A basic example of the Telnet communication structure using the Windows Telnet Client is summarized below:

1) Set up Telnet connection to a switch matrix with IP address 192.168.9.60:

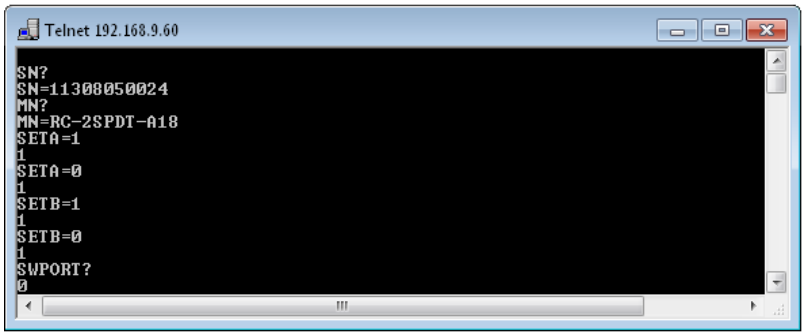


2) The "line feed" character is returned indicating the connection was successful:



3) The password (if enabled) must be sent as the first command in the format "PWD=x;". A return value of "1" indicates success:

4) Any number of commands and queries can be sent as needed:



5) Use the control and "]" keys to end the session.

2.8. Device Discovery Using UDP

Limited support of UDP is provided for the purpose of "device discovery." This allows a user to request the IP address and configuration of all Mini-Circuits' devices within the same family, connected on the network. Full control of those units is then accomplished using SSH, HTTP or Telnet, as detailed previously.

Note: UDP is a simple transmission protocol that provides no method for error correction or guarantee of receipt.

UDP Ports

Mini-Circuits' Ethernet enabled devices are configured to listen on UDP port 4950 and answer on UDP port 4951. Communication on these ports must be allowed through the computer's firewall in order to use UDP for device discovery. If the switch's IP address is already known, it is not necessary to use UDP.

Transmission

The command `MCLRFSWITCH?` should be broadcast to the local network using UDP protocol on port 4950.

Receipt

All Mini-Circuits RC switch matrices that receive the request will respond with the following information (each field separated by CrLf) on port 4951:

- Model Name
- Serial Number
- IP Address/Port
- Subnet Mask
- Network Gateway
- Mac Address

Example

Sent Data: `MCLRFSWITCH?`

Received Data:

```
Model Name: RC-2SPDT-A18
Serial Number: 11302120001
IP Address=192.168.9.101 Port: 80
Subnet Mask=255.255.0.0
Network Gateway=192.168.9.0
Mac Address=D0-73-7F-82-D8-01

Model Name: RC-2SPDT-A18
Serial Number: 11302120002
IP Address=192.168.9.102 Port: 80
Subnet Mask=255.255.0.0
Network Gateway=192.168.9.0
Mac Address=D0-73-7F-82-D8-02
```

3. SCPI Commands for Switch Control

The recommended method for controlling the device is a series of ASCII text commands based on SCPI (Standard Commands for Programmable Instruments). These commands can be sent using any of the APIs detailed in this manual.

The SCPI commands / queries are case insensitive and sent as an ASCII text string (up to 63 characters). The response from the system is also in the form of an ASCII text string.

3.1. SCPI – Identification

3.1.1. GET MODEL NAME

MN?

Return the Mini-Circuits model name.

Return Value

`MN=[model]`

Value	Description
`[model]`	Model name of the connected device

Examples

String to Send	String Returned
`MN?`	`MN=RC-2SPDT-A18`

HTTP Implementation: `http://10.10.10.10/MN?`

3.1.2. GET SERIAL NUMBER

SN?

Returns the serial number.

Return Value

`SN=[serial]`

Value	Description
`[serial]`	Serial number of the connected device

Examples

String to Send	String Returned
`SN?`	`SN=12208010025`

HTTP Implementation: `http://10.10.10.10/SN?`

3.1.3. GET FIRMWARE

FIRMWARE?

Returns the internal firmware version.

Return Value

Value	Description
[firmware]	The current firmware version, for example “B3”.

Examples

String to Send	String Returned
FIRMWARE?	B3

HTTP Implementation: http://10.10.10.10/FIRMWARE?

3.1.4. GET INTERNAL TEMPERATURE

TEMP[sensor]?

Returns the internal temperature measured at 1 of the internal sensors (model dependent):

- RC-1SPDT-x (1 sensor)
- RC-2SPDT-x (2 sensors)
- RC-3SPDT-x (2 sensors)
- RC-4SPDT-x (2 sensors)
- RC-8SPDT-x (3 sensors)
- RC-1SP4T-x (1 sensor)
- RC-2SP4T-x (2 sensors)
- RC-1SP6T-x (1 sensor)
- RC-2SP6T-x (2 sensors)
- RC-2MTS-x (2 sensors)
- RC-3MTS-x (2 sensors)
- ZTRC-4SPDT-x (2 sensors)
- ZTRC-8SPDT-x (3 sensors)

Note: "+25.00" will be returned when polling a sensor which is not fitted

Parameters

Parameter	Description
[sensor]	The internal temperature sensor (1 to 3) to poll

Return Value

Value	Description
[temp]	The temperature reading of the specified internal sensor in degrees Celsius.

Examples

String to Send	String Returned
TEMP1?	+37.25

HTTP Implementation: http://10.10.10.10/TEMP1?

See Also

Get Heat Alarm
Get Fan Status

3.1.5. GET HEAT ALARM

HEATALARM?

Returns an alarm notification if any internal temperature sensor exceeds the factory programmed limits (45°C on the PCB or 48°C on the internal switch case).

Return Value

Value	Description
[status]	Integer value to indicate the heat alarm status: 0 – Device is within normal operating temperature limits 1 – Device temperature has exceeded the recommended limits

Examples

String to Send	String Returned
HEATALARM?	0

HTTP Implementation: http://10.10.10.10/HEATALARM?

See Also

Get Internal Temperature

3.1.6. GET FAN STATUS

FAN?

This function indicates whether the internal fan is currently operating (model dependent).

Applies To

All models except ZTRC-4SPDT-A18

Return Value

Value	Description
[status]	Integer value to indicate the fan status: 0 – Fan is not operating 1 – Fan is currently operating

Examples

String to Send	String Returned
FAN?	1

HTTP Implementation: http://10.10.10.10/FAN?

See Also

Get Internal Temperature

3.2. SCPI – Switch States

3.2.1. SET SINGLE SPDT / TRANSFER SWITCH

SET[switch]=[state]

Sets a single SPDT or transfer switch.

Parameters

Parameter	Description
[switch]	The individual switch (A-H) to be controlled, eg. SETA for switch A or SETB for switch B
[state]	The state (0 or 1) into which the switch should be set: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch)

Return Value

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
SETA=1	1

HTTP Implementation: http://10.10.10.10/SETA=1

3.2.2. SET ALL SPDT / TRANSFER SWITCHES

SETP=[states]

Sets all SPDT or transfer switches.

Parameters

Parameter	Description
[states]	Integer value of a byte that represents the switch states. Each bit in the byte represents the state of an individual switch with value: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch) The least significant bit (LSB) represents switch A and the most significant bit (MSB) represents switch H (if applicable).

Return Value

Value	Description
0	Command failed
1	Command completed successfully

Example

RC-8SPDT-A18 has 8 SPDT switches available (named A to H). To set switches A, B and H to state 1 (Com connected to port 2) and all other switches to state 0 (Com port connected to port 1), the byte can be represented as:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**
Description	State	State	State	State	State	State	State	State
Value	1	0	0	0	0	0	1	1

[states] = 10000011 (binary)
= 131 (decimal)

String to Send	String Returned
SETP=131	1

HTTP Implementation: http://10.10.10.10/SETP=131

See Also

Set Single SPDT / Transfer Switch

3.2.3. GET ALL SPDT / TRANSFER SWITCH STATES

SWPORT?

Return the state of all switches within an SPDT or transfer switch box.

Parameters

Value	Description
[states]	Integer value representing the switch states. The value should be interpreted as a byte string, with each bit representing the state of an individual SPDT or transfer switch as below: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch) The least significant bit (LSB) represents switch A and the most significant bit (MSB) represents switch H (if applicable).

Examples

String to Send	String Returned
SWPORT?	131

HTTP Implementation: http://10.10.10.10/SWPORT?

The decimal value 131 corresponds to 10000011 in binary so the switch states are:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	H	G	F	E	D	C	B	A
Value	1	0	0	0	0	0	1	1

Switches A, B and H are in state 1 (Com port connected to port 2); all others are in state 0 (Com port connected to port 1).

3.2.4. SET SINGLE SP4T SWITCH

SP4T[switch]:STATE:[state]

Sets a single SP4T switch.

Parameters

Parameter	Description
[switch]	The individual switch (A or B) to be controlled
[state]	The switch state to set: 0 = All ports disconnected 1 = Connect Com port to port 1 2 = Connect Com port to port 2 3 = Connect Com port to port 3 4 = Connect Com port to port 4

Return Value

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
SP4TA:STATE:3	1

HTTP Implementation: http://10.10.10.10/SP4TA:STATE:3

3.2.5. GET SINGLE SP4T SWITCH STATE

SP4T[switch]:STATE?

Returns the state of a single SP4T switch.

Parameters

Parameter	Description
[switch]	The individual switch (A or B) to be controlled.

Return Value

Value	Description
[state]	The switch state: 0 = All ports disconnected 1 = Connect Com port to port 1 2 = Connect Com port to port 2 3 = Connect Com port to port 3 4 = Connect Com port to port 4

Examples

String to Send	String Returned
SP4TA:STATE?	3

HTTP Implementation: http://10.10.10.10/SP4TA:STATE?

3.2.6. SET ALL SP4T SWITCHES

SETP=[state]

Sets all SP4T switches.

Parameters

Parameter	Description
[state]	Integer value representing all SP4T switch states. The value is derived from a byte string, with each bit corresponding to the input / output port of a single switch: Switch A (all models): If Bits 0 to 3 = 0, switch A has all ports disconnected If Bit 0 = 1, switch A has Com connected to port 1 If Bit 1 = 1, switch A has Com connected to port 2 If Bit 2 = 1, switch A has Com connected to port 3 If Bit 3 = 1, switch A has Com connected to port 4 Switch B (model dependent): If Bits 4 to 7 = 0, switch B has all ports disconnected If Bit 4 = 1, switch B has Com connected to port 1 If Bit 5 = 1, switch B has Com connected to port 2 If Bit 6 = 1, switch B has Com connected to port 3 If Bit 7 = 1, switch B has Com connected to port 4

Return Value

Value	Description
0	Command failed
1	Command completed successfully
4	Switch not set (invalid switch state requested)

Examples

Set switch A to position 2 (COM<>2) and switch B to position 4 COM<>4):

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	**Switch B**				**Switch A**			
Description	Port 4	Port 3	Port 2	Port 1	Port 4	Port 3	Port 2	Port 1
Value	1	0	0	0	0	0	1	0

[state] = 10000010 (binary)
= 130 (decimal)

String to Send	String Returned
SETP=130	1

HTTP Implementation: http://10.10.10.10/SETP=130

See Also
Set Single SP4T Switch

3.2.7. GET ALL SP4T SWITCH STATES

SWPORT?

Returns the state of all SP4T switches.

Return Value

Value	Description
[state]	Integer value representing all SP4T switch states. The value is derived from a byte string, with each bit corresponding to the input / output port of a single switch: Switch A (all models): If Bits 0 to 3 = 0, switch A has all ports disconnected If Bit 0 = 1, switch A has Com connected to port 1 If Bit 1 = 1, switch A has Com connected to port 2 If Bit 2 = 1, switch A has Com connected to port 3 If Bit 3 = 1, switch A has Com connected to port 4 Switch B (model dependent): If Bits 4 to 7 = 0, switch B has all ports disconnected If Bit 4 = 1, switch B has Com connected to port 1 If Bit 5 = 1, switch B has Com connected to port 2 If Bit 6 = 1, switch B has Com connected to port 3 If Bit 7 = 1, switch B has Com connected to port 4

Examples

String to Send	String Returned
SWPORT?	129

HTTP Implementation: http://10.10.10.10/SWPORT?

The decimal value 129 corresponds to 10000001 in binary so the switch states are:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	**Switch B**				**Switch A**			
Description	Port 4	Port 3	Port 2	Port 1	Port 4	Port 3	Port 2	Port 1
Value	1	0	0	0	0	0	0	1

Switch B is in position 4; Com port connected to port 2
Switch A is in position 1; Com port connected to port 1

3.2.8. SET SP6T SWITCH

SP6T[switch]:STATE:[state]

Sets a single SP6T switch state.

Parameters

Parameter	Description
[switch]	The individual switch (A or B) to be controlled
[state]	The switch state to set: 0 = All ports disconnected 1 = Connect Com port to port 1 2 = Connect Com port to port 2 3 = Connect Com port to port 3 4 = Connect Com port to port 4 5 = Connect Com port to port 5 6 = Connect Com port to port 6

Return Value

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
SP6TA:STATE:3	1

HTTP Implementation: http://10.10.10.10/SP6TA:STATE:3

3.2.9. GET SP6T SWITCH STATE

SP6T[switch]:STATE?

Returns the state of a single SP6T switch.

Parameters

Parameter	Description
[switch]	The individual switch (A or B) to be controlled.

Return Value

Value	Description
[state]	The switch state: 0 = All ports disconnected 1 = Connect Com port to port 1 2 = Connect Com port to port 2 3 = Connect Com port to port 3 4 = Connect Com port to port 4 5 = Connect Com port to port 5 6 = Connect Com port to port 6

Examples

String to Send	String Returned
SP6TA:STATE?	3

HTTP Implementation: http://10.10.10.10/SP6TA:STATE?

3.3. SCPI – Counters & Power-Up

3.3.1. GET SPDT / TRANSFER SWITCH COUNTERS

SC[switch_name]?

Returns the number of switching cycles undertaken by a specified SPDT or transfer switch. The command to save all switch counters and states must be issued before powering off the switch box to ensure an accurate count is maintained.

Parameters

Parameter	Description
[switch_name]	The letter indicating the switch name, A to H (model dependent)

Return Value

Value	Description
[count]	The number of switching cycles undertaken by the specified switch

Examples

String to Send	String Returned
SCB?	10500

HTTP Implementation: http://10.10.10.10/SCB?

3.3.2. GET SP4T SWITCH COUNTERS

SP4T[switch_name]:COUNTERS?

Returns the number of switching cycles undertaken by a specified SP4T switch. The command to save all switch counters and states must be issued before powering off the switch box to ensure an accurate count is maintained.

Parameters

Parameter	Description
[switch_name]	The letter indicating the switch name, A to B (model dependent)

Return Value

1=[count1] 2=[count2] 3=[count3] 4=[count4]

Value	Description
[count1]	Number of cycles to port 1
[count2]	Number of cycles to port 2
[count3]	Number of cycles to port 3
[count4]	Number of cycles to port 4

Examples

String to Send	String Returned
SP4TA:COUNTERS?	1=1553 2=34650 3=10952 4=520

HTTP Implementation: http://10.10.10.10/SP4TA:COUNTERS?

3.3.3. GET SP6T SWITCH COUNTERS

SP6T[switch_name]:COUNTERS?

Returns the number of switching cycles undertaken by a specified SP6T switch. The command to save all switch counters and states must be issued before powering off the switch box to ensure an accurate count is maintained.

Parameters

Parameter	Description
[switch_name]	The letter indicating the switch name, A to B (model dependent)

Return Value

1=[count1] 2=[count2] 3=[count3] 4=[count4] 5=[count5] 6=[count6]

Value	Description
[count1]	Number of cycles to port 1
[count2]	Number of cycles to port 2
[count3]	Number of cycles to port 3
[count4]	Number of cycles to port 4
[count5]	Number of cycles to port 5
[count6]	Number of cycles to port 6

Examples

String to Send	String Returned
SP6TA:COUNTERS?	1=1553 2=34650 3=10952 4=520 5=6514 6=109

HTTP Implementation: http://10.10.10.10/SP6TA:COUNTERS?

3.3.4. SET POWER-UP MODE

ONPOWERUP:LASTSTATE[mode]

Sets whether or not the switch box should power-up with all switches in the last saved state. If this mode is not selected then all switches will power-up in the default state.

Note: Switch states & counters must be saved prior to powering off the switch box in order to remember the last state.

Applies To

Firmware version D2 or later

Parameters

Parameter	Description
[mode]	ON = Switches will power-up in the last saved state OFF = Switches will power-up in the default state: • SP4T & SP6T: All ports disconnected • SPDT: Com connected to port 1 • Transfer: J1 <> J3 and J2 <> J4

Return Value

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
ONPOWERUP:LASTSTATE:ON	1

HTTP Implementation: http://10.10.10.10/ONPOWERUP:LASTSTATE:ON

3.3.5. GET POWER-UP MODE

ONPOWERUP:LASTSTATE?

Indicates whether or not the switch box will power-up with all switches in the last saved state.

Applies To

Firmware version D2 or later

Return Value

Value	Description
0	Switch box will power-up with all switches in the last remembered state
1	Switch box will power-up with all switches in the default state

Examples

String to Send	String Returned
ONPOWERUP:LASTSTATE?	1

HTTP Implementation: http://10.10.10.10/ONPOWERUP:LASTSTATE?

3.3.6. SAVE SWITCH COUNTERS & STATES

SCOUNTERS:STORE:INITIATE

Transfers the latest switch counters and switch states from temporary to permanent memory. This command should be sent following completion of all switching routines and prior to powering off the switch in order to ensure that the data is permanently saved. During normal operation, this data is internally stored in volatile memory but automatically updated into permanent memory every 3 minutes. Therefore, if the device is idle for more than 3 minutes there is no need to re-issue the command.

Applies To

Firmware version D1 or later

Return Value

Value	Description
0	Command failed
1	Command completed successfully
2	Save not completed; command already issued within last 3 minutes

Examples

String to Send	String Returned
SCOUNTERS:STORE:INITIATE	1

HTTP Implementation: http://10.10.10.10/SCOUNTERS:STORE:INITIATE

3.4. SCPI - Ethernet Configuration

The following functions provide a method to review and edit the Ethernet TCP / IP connection parameters. The Update Ethernet Settings command must be used to save settings and restart the controller. Refer to Ethernet Control API for further details on Ethernet control of the device.

These commands require firmware E9 or later to be installed.

3.4.1. GET CURRENT ETHERNET CONFIGURATION

:ETHERNET:CONFIG:LISTEN?

Returns the IP configuration that is currently use, either the static IP entered by the user, or the server assigned dynamic IP configuration when DHCP is enabled.

Applies To

Firmware E9 or later

Return String
`[ip];[mask];[gateway]`

Variable	Description
`[ip]`	Active IP address of the device
`[mask]`	Subnet mask for the network
`[gateway]`	IP address of the network gateway

Examples

String to Send	String Returned
`:ETHERNET:CONFIG:LISTEN?`	`192.100.1.1;255.255.255.0;192.100.1.0`

HTTP Implementation: `http://10.10.10.10/:ETHERNET:CONFIG:LISTEN?`

3.4.2. GET MAC ADDRESS

:ETHERNET:CONFIG:MAC?

Returns the physical MAC (media access control) address of the device.

Applies To

Firmware E9 or later

Return String

Variable	Description
`[mac]`	MAC address

Examples

String to Send	String Returned
`:ETHERNET:CONFIG:MAC?`	`D0-73-7F-82-D8-01`

HTTP Implementation: `http://10.10.10.10/:ETHERNET:CONFIG:MAC?`

3.4.3. GET DHCP STATUS

:ETHERNET:CONFIG:DHCPENABLED?

Indicates whether DHCP (dynamic host control protocol) is currently enabled. When disabled, the device will attempt to connect using the user-entered static IP parameters.

Applies To

Firmware E9 or later

Return String

Value	Description
0	DHCP disabled (static IP settings will be used)
1	DHCP enabled (IP address will be requested from DHCP server on the network)

Examples

String to Send	String Returned
:ETHERNET:CONFIG:DHCPENABLED?	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:DHCPENABLED?

3.4.4. USE DHCP

:ETHERNET:CONFIG:DHCPENABLED:[enabled]

Enables or disables DHCP (dynamic host control protocol). When disabled, the device will attempt to connect using the user-entered static IP parameters. By default, DHCP is enabled.

Applies To

Firmware E9 or later

Parameters

Value	Description
0	DHCP disabled (static IP settings will be used)
1	DHCP enabled (IP address will be requested from DHCP server on the network)

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:DHCPENABLED:1	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:DHCPENABLED:1

3.4.5. GET STATIC IP ADDRESS

:ETHERNET:CONFIG:IP?

Returns the user-entered static IP address.

Applies To

Firmware E9 or later

Return String

Variable	Description
[ip]	String containing the static IP address

Examples

String to Send	String Returned
:ETHERNET:CONFIG:IP?	192.100.1.1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:IP?

3.4.6. SET STATIC IP ADDRESS

:ETHERNET:CONFIG:IP:[ip]

Sets the static IP address to be used when DHCP is disabled. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[ip]	String containing the static IP address

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:IP:192.100.1.1	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:IP:192.100.1.1

3.4.7. GET STATIC NETWORK GATEWAY

:ETHERNET:CONFIG:NG?

Returns the user-entered network gateway IP address.

Applies To

Firmware E9 or later

Return String

Variable	Description
[gateway]	String containing the IP address of the network gateway

Examples

String to Send	String Returned
:ETHERNET:CONFIG:NG?	192.168.1.0

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:NG?

3.4.8. SET STATIC NETWORK GATEWAY

:ETHERNET:CONFIG:NG:[gateway]

Sets the IP address of the network gateway to be used when DHCP is disabled.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[gateway]	String containing IP address of the network gateway

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:NG:192.100.1.0	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:NG:192.168.100.1.0

3.4.9. GET STATIC SUBNET MASK

:ETHERNET:CONFIG:SM?

Returns the subnet mask to be used for Ethernet communication when static IP settings are in use. DHCP must be disabled for this setting to apply, otherwise a dynamic IP address will be in use.

Applies To

Firmware E9 or later

Return String

Variable	Description
[mask]	String containing the subnet mask

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SM?	255.255.255.0

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SM?

3.4.10. SET STATIC SUBNET MASK

:ETHERNET:CONFIG:SM:[mask]

Sets the subnet mask to be used when DHCP is disabled. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[mask]	String containing the subnet mask

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SM:255.255.255.0	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SM:255.255.255.0

3.4.11. GET HTTP PORT

:ETHERNET:CONFIG:HTPORT?

Returns the TCP/IP port in use for HTTP communication. The default is port 80.

Applies To

Firmware E9 or later

Return String

Variable	Description
[port]	TCP / IP port to be used for HTTP communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:HTPORT?	8080

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:HTPORT?

3.4.12. SET HTTP PORT & ENABLE / DISABLE HTTP

:ETHERNET:CONFIG:HTPORT:[port]

Sets the TCP / IP port to be used for HTTP communication. The default is port 80. Set port 0 or 65535 to disable HTTP (requires firmware F1 or later) or any valid port to enable.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[port]	TCP / IP port to be used for HTTP communication

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:HTPORT:8080	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:HTPORT:8080

3.4.13. GET TELNET PORT

:ETHERNET:CONFIG:TELNETPORT?

Returns the TCP/IP port in use for Telnet communication. The default is port 23.

Applies To

Firmware E9 or later

Return String

Variable	Description
[port]	TCP / IP port to be used for Telnet communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:TELNETPORT?	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:TELNETPORT?

3.4.14. SET TELNET PORT & ENABLE / DISABLE TELNET

:ETHERNET:CONFIG:TELNETPORT:[port]

Sets the TCP / IP port to be used for Telnet communication. The default is port 23. Set port 0 or 65535 to disable Telnet (requires firmware F1 or later) or any valid port to enable.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[port]	TCP / IP port to be used for Telnet communication

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:TELNETPORT:21	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:TELNETPORT:21

3.4.15. GET SSH PORT

:ETHERNET:CONFIG:SSHPORT?

Returns the TCP/IP port in use for SSH communication. The default is port 22.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Applies To

Firmware E9 or later

Return String

Variable	Description
[port]	TCP / IP port to be used for SSH communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SSHPORT?	21

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SSHPORT?

3.4.16. SET SSH PORT

:ETHERNET:CONFIG:SSHPORT:[port]

Sets the TCP / IP port to be used for SSH communication. The default is port 22.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[port]	TCP / IP port to be used for SSH communication

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SSHPORT:21	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SSHPORT:21

3.4.17. GET SSH LOGIN NAME

:ETHERNET:CONFIG:SSHLOGINNAME?

Returns the login name to be used for SSH communication.

Applies To

Firmware E9 or later

Return String

Variable	Description
[name]	Login name to be used for SSH communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SSHLOGINNAME?	Ssh_user

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SSHLOGINNAME?

3.4.18. SAVE SSH LOGIN NAME

:ETHERNET:CONFIG:SSHLOGINNAME:[name]

Sets the login name to be used for SSH communication.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[name]	The login name for SSH communication

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:SSHLOGINNAME:ssh_user	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:SSHLOGINNAME:ssh_user

3.4.19. GET PASSWORD REQUIREMENT

:ETHERNET:CONFIG:PWDENABLED?

Indicates whether the password is currently enabled for HTTP / Telnet (the password is always required for SSH).

Applies To

Firmware E9 or later

Return String

Value	Description
0	Password not required for Ethernet communication
1	Password required for Ethernet communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:PWDENABLED?	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:PWDENABLED?

3.4.20. SET PASSWORD REQUIREMENT

:ETHERNET:CONFIG:PWDENABLED:[enabled]

Enables or disables the password requirement for HTTP / Telnet (the password is always required for SSH).

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Applies To

Firmware E9 or later

Parameters

Value	Description
0	Password not required for Ethernet communication
1	Password required for Ethernet communication

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:PWDENABLED:1	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:PWDENABLED:1

3.4.21. GET PASSWORD

:ETHERNET:CONFIG:PWD?

Returns the current password for SSH / HTTP / Telnet communication.

Applies To

Firmware E9 or later

Return String

Variable	Description
[pwd]	Password for Ethernet communication

Examples

String to Send	String Returned
:ETHERNET:CONFIG:PWD?	PASS-123

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:PWD?

3.4.22. SET PASSWORD

:ETHERNET:CONFIG:PWD:[pwd]

Sets the password used for SSH / HTTP / Telnet communication. The password will only apply for HTTP / Telnet after enabling using Set Password Requirement.

Applies To

Firmware E9 or later

Parameters

Variable	Description
[pwd]	Password for Ethernet communication (not case sensitive, 20 characters maximum)

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:PWD:PASS-123	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:PWD:PASS-123

3.4.23. UPDATE ETHERNET SETTINGS

:ETHERNET:CONFIG:INIT

Resets the Ethernet controller and restarts with the latest saved settings. Any subsequent commands / queries to the device will need to be attempted using the new Ethernet configuration. If a connection cannot be established, it may indicate an invalid configuration was created (for example a static IP address which clashes with another device on the network). The Ethernet settings can always be overwritten by connecting to the system using the USB connection.

Applies To

Firmware E9 or later

Return String

Value	Description
0	Command failed
1	Command completed successfully

Examples

String to Send	String Returned
:ETHERNET:CONFIG:INIT	1

HTTP Implementation: http://10.10.10.10/:ETHERNET:CONFIG:INIT

4. USB Control API for Microsoft Windows

Mini-Circuits' API for USB control from a computer running Microsoft Windows is provided in the form of a DLL file. 3 DLL options are provided to offer the widest possible support, with the same functionality in each case.

1) .Net Framework 4.5 DLL - This is the recommended API for most modern operating systems
2) .Net Framework 2.0 DLL - Provided for legacy support of older computers / operating systems, with an installed version of the .Net framework prior to 4.5
3) ActiveX com object - Provided for legacy support of older environments which do not support .Net

The latest version of each DLL file can be downloaded from the Mini-Circuits website at:

https://www.minicircuits.com/WebStore/RF-Mechanical-Compact-Switch.html

4.1. DLL API Options

4.1.1. .NET FRAMEWORK 4.5 DLL (RECOMMENDED)

The recommended API option for USB control from most modern programming environments running on Windows.

Filename: mcl_RF_Switch_Controller_NET45.dll

Requirements

1) Microsoft Windows with .Net framework 4.5 or later
2) Programming environment with ability to support .Net components

Installation

1) Download the latest DLL file from the Mini-Circuits website
2) Copy the .dll file to the preferred directory (the recommendation is to use the same folder as the programming project, or `C:\WINDOWS\SysWOW64`
3) Right-click on the DLL file in the save location and select Properties to check that Windows has not blocked access to the file (check "Unblock" if the option is shown)
4) No registration or further installation action is required

4.1.2. .NET FRAMEWORK 2.0 DLL (LEGACY SUPPORT)

Provided for support of systems with an older version of the .Net framework installed (prior to 4.5).

Filename: mcl_RF_Switch_Controller64

Requirements

1) Microsoft Windows with .Net framework 2.0 or later
2) Programming environment with ability to support .Net components

Installation

1) Download the latest DLL file from the Mini-Circuits website:
2) Copy the .dll file to the preferred directory (the recommendation is to use the same folder as the programming project, or `C:\WINDOWS\SysWOW64`
3) Right-click on the DLL file in the save location and select Properties to check that Windows has not blocked access to the file (check "Unblock" if the option is shown)
4) No registration or further installation action is required

4.1.3. ACTIVEX COM OBJECT DLL (LEGACY SUPPORT)

Provided for support of programming environments which do not support .Net components.

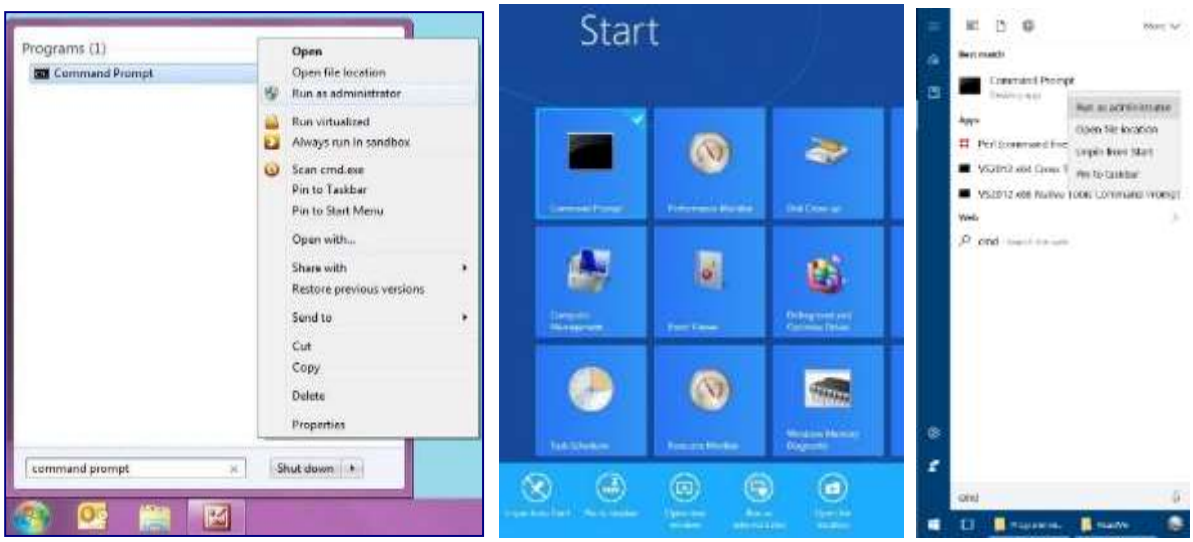
Filename: MCL_RF_Switch_Controller.dll

Requirements

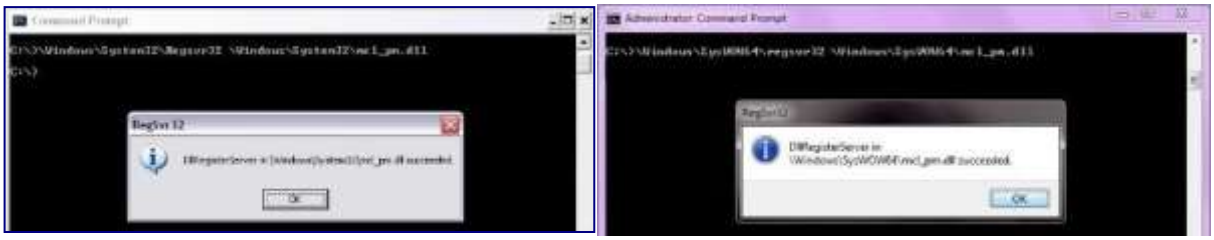
1) Microsoft Windows
2) Programming environment with support for ActiveX components

Installation

1. Copy the DLL file to the correct directory:
   For 32-bit Windows operating systems: `C:\WINDOWS\System32`
   For 64-bit Windows operating systems: `C:\WINDOWS\SysWOW64`
2. Open the Command Prompt in "Elevated" mode:
   a. Open the Start Menu/Start Screen and type "Command Prompt"
   b. Right-click on the shortcut for the Command Prompt
   c. Select "Run as Administrator"
   d. You may be prompted to enter the log in details for an Administrator account if the current user does not have Administrator privileges on the local PC
3. Use regsvr32 to register the DLL:
   a. 32-bit PC:
      `\WINDOWS\System32\Regsvr32 \WINDOWS\System32\ MCL_RF_Switch_Controller.dll`
   b. 64-bit PC:
      `\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\ MCL_RF_Switch_Controller.dll`
4. Hit enter to confirm and a message box will appear to advise of successful registration



Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)



Registering the DLL

4.2. Referencing the DLL

Most programming environments require a reference to be set to the relevant DLL file, either in the IDE menu or within the program. Multiple instances of the DLL control class can then be created (referred to as MyPTE1 and MyPTE2 below) in order to connect to as many devices as needed

Example Declarations Using the .NET 4.5 DLL (mcl_RF_Switch_Controller_NET45.dll)

(For operation with the .Net 2.0 DLL, replace "mcl_RF_Switch_Controller_NET45" with "mcl_RF_Switch_Controller64")

Python	`import clr  # Import the pythonnet CLR library`
	`clr.AddReference('C:\Windows\SysWOW64\mcl_RF_Switch_Controller_NET45.dll')`
	`from mcl_RF_Switch_Controller_NET45 import USB_RF_SwitchBox`
	`MyPTE1 = USB_RF_SwitchBox()`
	`MyPTE2 = USB_RF_SwitchBox()`

Visual Basic	`Public MyPTE1 As New mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox` `Public MyPTE2 As New mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox`

Visual C++	`mcl_RF_Switch_Controller_NET45::USB_RF_SwitchBox ^MyPTE1 = gcnew`
	`mcl_RF_Switch_Controller_NET45::USB_RF_SwitchBox();`
	`mcl_RF_Switch_Controller_NET45::USB_RF_SwitchBox ^MyPTE2 = gcnew`
	`mcl_RF_Switch_Controller_NET45::USB_RF_SwitchBox();`

Visual C#	`mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox MyPTE1 = new`
	`mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox();`
	`mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox MyPTE2 = new`
	`mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox();`

MatLab	`MCL_SW=NET.addAssembly('C:\Windows\SysWOW64\mcl_RF_Switch_Controller_NET45.dll')`
	`MyPTE1 = mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox`
	`MyPTE2 = mcl_RF_Switch_Controller_NET45.USB_RF_SwitchBox`

Example Declarations using the ActiveX DLL (MCL_RF_Switch_Controller.dll)

Visual Basic	`Public MyPTE1 As New MCL_RF_Switch_Controller.USB_RF_Switch` `Public MyPTE2 As New MCL_RF_Switch_Controller.USB_RF_Switch`

Visual C++	`MCL_RF_Switch_Controller::USB_RF_Switch ^MyPTE1 = gcnew`
	`MCL_RF_Switch_Controller::USB_RF_Switch();`
	`MCL_RF_Switch_Controller::USB_RF_Switch ^MyPTE2 = gcnew`
	`MCL_RF_Switch_Controller::USB_RF_Switch();`

Visual C#	`public MCL_RF_Switch_Controller.USB_RF_Switch MyPTE1 = new`
	`MCL_RF_Switch_Controller.USB_RF_Switch();`
	`public MCL_RF_Switch_Controller.USB_RF_Switch MyPTE2 = new`
	`MCL_RF_Switch_Controller.USB_RF_Switch();`

MatLab	`MyPTE1 = actxserver(MCL_RF_Switch_Controller.USB_RF_Switch')` `MyPTE2 = actxserver(MCL_RF_Switch_Controller.USB_RF_Switch')`

4.3. Additional DLL Considerations

4.3.1. MINI-CIRCUITS' DLL USE IN PYTHON / MATLAB

Some functions are defined within Mini-Circuits' DLLs with arguments to be passed by reference. This allows the variables (with their updated values) to be used later within the program, after the DLL function has executed. This methodology fits with many programming environments (including C#, C++ and VB) but is interpreted slightly differently by Python and MatLab:

- Typical operation (C#, C++, VB):
  - The function has an integer return value to indicate success / failure (1 or 0)
  - One or more variables are passed to the function by reference so that the updated values are available to the program once function execution has completed
- Python implementation:
  - Any parameters passed by reference to a function can be ignored (an empty string can be provided in place of the variable)
  - The return value from the function will change from the single integer value as defined in this manual, to a tuple
  - The tuple format will be [function_return_value, function_parameter]
- MatLab implementation:
  - Any parameters passed by reference to a function can be ignored (an empty string can be provided in place of the variable)
  - The return value from the function will change from the single integer value as defined in this manual to an array of values
  - The function must be assigned to an array variable of the correct size, in the format [function_return_value, function_parameter]

The examples below illustrate how a function of this type is defined in the DLL and how that same function is implemented in C#, Python and MatLab.

Definition	**int Read_SN(ByRef string SN)**

Visual C#	`status = MyPTE1.Read_SN(ref(SN));`
	`if(status > 0)`
	`{`
	`  MessageBox.Show("The connected device is " + SN);`
	`}`

Python	`status = MyPTE1.Read_SN("")`
	`if status[0] > 0:`
	`  SN = str(status[1])`
	`  print('The connected device is ', SN)`

MatLab	`[status, SN] = MyPTE1.Read_SN('')`
	`if status > 0`
	`  h = msgbox('The connected device is ', SN)`
	`end`

4.3.2. MINI-CIRCUITS' DLL USE IN LABWINDOWS / CVI

It is recommended to use one of the .Net DLL files for USB control of Mini-Circuits devices from CVI. The initial steps to set up the instrument driver in the CVI project are as below (note: there may be slight variations between CVI versions):

1. It is recommended to place the DLL into the CVI project folder (refer to the instructions above, to download the .Net DLL and ensure that Windows has not blocked it)
2. Open CVI:
    a. From the menu select Tools > Create .NET controller
    b. Check the option to specify the assembly by path and filename
    c. Browse to the working directory and select the DLL file
    d. Under the target instrument enter the working directory path
    e. CVI should now compile and create the instrument driver (.fp) file
    f. Under Instrument files on the left of the CVI screen there should now be an object for the DLL file
    g. Clicking on the object provides access to all methods and properties within the DLL.

The process above creates an instrument driver in CVI which has the effect of a "wrapper" around the Mini-Circuits DLL. This "wrapper" provides a set of definitions for each of the DLL functions which allow them to be used within CVI. The new definitions contain some additional CVI specific content which make them appear slightly different to the DLL definitions summarized in this manual, but the arguments and return values remain the same.

The example below demonstrates how the DLL definitions in this manual convert to the form used within the CVI "wrapper":

Mini-Circuits' DLL Definition	`short Connect(string [SN])`

Implementation in CVI instrument driver	`int CVIFUNC ModularZT_NET45_USB_ZT_Open_Sensor(`
	`  ModularZT_NET45_USB_ZT __instance,`
	`  char ** SN,`
	`  short * __returnValue,`
	`  CDotNetHandle * __exception);`

Explanation	The CVI function definition contains the following arguments: 1. An instance of the Mini-Circuits DLL class 2. The argument(s) defined in the Mini-Circuits DLL function 3. The return value from the Mini-Circuits DLL function 4. An error indicator object (part of the CVI / .Net instrument driver)

4.4. DLL – Connect & Identify

4.4.1. CONNECT

Short Connect(Optional String SN)

Initializes the USB connection. The serial number should be included if multiple devices are connected. The device should be disconnected on completion of the program using the Disconnect function.

Parameters

Variable	Description
SN	Optional. The serial number for the specific device to connect (if omitted the first device found on the bus will be connected).

Return Values

Value	Description
0	No connection was possible
>0	Connection successfully established

Examples

Python	`response = MyPTE1.Connect()` `status = response[0]`
Visual Basic	`status = MyPTE1.Connect(SN)`
Visual C++	`status = MyPTE1->Connect(SN);`
Visual C#	`status = MyPTE1.Connect(ref(SN));`
MatLab	`status = MyPTE1.Connect(SN);`

4.4.2. CONNECT BY ADDRESS

Short ConnectByAddress(Optional Short Address)

Initialize the USB connection by referring to a user-defined USB address. The address is an integer number from 1 to 255 which can be assigned using the Set_Address function (the factory default is 255). The device should be disconnected on completion of the program using the Disconnect function.

Parameters

Variable	Description
Address	The USB address of the device to connect (if omitted the first device found on the bus will be connected).

Return Values

Value	Description
0	No connection was possible
>0	Connection successfully established

Examples

Python	`response = MyPTE1.ConnectByAddress(Address)` `status = response[0]`
Visual Basic	`status = MyPTE1.ConnectByAddress(Address)`
Visual C++	`status = MyPTE1->ConnectByAddress(Address);`
Visual C#	`status = MyPTE1.ConnectByAddress(ref(Address));`
MatLab	`[status, Address] = MyPTE1.ConnectByAddress(Address);`

4.4.3. DISCONNECT

Void Disconnect()

Terminates the connection. It is strongly recommended that this function is used prior to ending the program. Failure to do so may result in a connection failure on the next attempt, requiring a power cycle to rectify.

Examples

Python	`status = MyPTE1.Disconnect()`
Visual Basic	`status = MyPTE1.Disconnect()`
Visual C++	`status = MyPTE1->Disconnect();`
Visual C#	`status = MyPTE1.Disconnect();`
MatLab	`status = MyPTE1.Disconnect();`

4.4.4. READ MODEL NAME

Short Read_ModelName(String ModelName)

Returns the Mini-Circuits part number.

Parameters

Variable	Description
ModelName	Required. A string variable that will be updated with the Mini-Circuits part number.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.Read_ModelName("")`
	`if status[0] > 0:`
	`  ModelName = str(status[1])`
	`  print('The connected device is ', ModelName)`

Visual Basic	`If MyPTE1.Read_ModelName(ModelName) > 0 Then`
	`  MsgBox ("The connected device is " & ModelName)`
	`End If`

Visual C++	`if (MyPTE1->Read_ModelName(ModelName) > 0 )`
	`{`
	`  MessageBox::Show("The connected device is " + ModelName);`
	`}`

Visual C#	`if (MyPTE1.Read_ModelName(ref(ModelName)) > 0 )`
	`{`
	`  MessageBox.Show("The connected device is " + ModelName);`
	`}`

MatLab	`[status, ModelName] = MyPTE1.Read_ModelName('')`
	`if status > 0`
	`  h = msgbox('The connected device is ', ModelName)`
	`end`

4.4.5. READ SERIAL NUMBER

Short Read_SN(String SN)

Returns the serial number.

Parameters

Variable	Description
ModelName	Required. String variable that will be updated with the serial number.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.Read_SN("")`
	`if status[0] > 0:`
	`  SN = str(status[1])`
	`  print('The connected device is ', SN)`

Visual Basic	`If MyPTE1.Read_SN(SN) > 0 Then`
	`  MsgBox ("The connected device is " & SN)`
	`End If`

Visual C++	`if (MyPTE1->Read_SN(SN) > 0 )`
	`{`
	`  MessageBox::Show("The connected device is " + SN);`
	`}`

Visual C#	`if (MyPTE1.Read_SN(ref(SN)) > 0 )`
	`{`
	`  MessageBox.Show("The connected device is " + SN);`
	`}`

MatLab	`[status, SN] = MyPTE1.Read_SN('')`
	`if status > 0`
	`  h = msgbox('The connected device is ', SN)`
	`end`

4.4.6. GET FIRMWARE

Short GetExtFirmware(ByRef Short A0, ByRef Short A1, By Ref Short A2, By Ref Short A3, By Ref String Firmware)

Returns the internal firmware version of the switch matrix along with three reserved variables (for factory use).

Parameters

Variable	Description
A0	Required. User defined variable for factory use only.
A1	Required. User defined variable for factory use only.
A2	Required. User defined variable for factory use only.
A3	Required. User defined variable for factory use only.
Firmware	Required. User defined string variable which will be updated with the current firmware version, for example "B3".

Return Values

Value	Description
0	Command failed
>0	Command completed successfully

Examples

Python	`status = MyPTE1.GetExtFirmware(A0, A1, A2, A3, Firmware)`
	`if status[0] > 0:`
	`  Firmware = str(status[5])`
	`  print("Firmware Version:", Firmware)`

Visual Basic	`If MyPTE1.GetExtFirmware(A0, A1, A2, A3, Firmware) > 0 Then`
	`  MsgBox ("Firmware Version: " & Firmware)`
	`End If`

Visual C++	`if (MyPTE1->GetExtFirmware(A0, A1, A2, A3, Firmware) > 0)`
	`{`
	`  MessageBox::Show("Firmware Version: " + Firmware);`
	`}`

Visual C#	`if(MyPTE1.GetExtFirmware(ref(A0),ref(A1),ref(A2),ref(A3),ref(Firmware)) > 0)`
	`{`
	`  MessageBox.Show("Firmware Version: " + Firmware);`
	`}`

MatLab	`[status,A0, A1, A2, A3, Firmware] = MyPTE1.GetExtFirmware(0, 0, 0, 0, '')`
	`if status > 0`
	`  h = msgbox('Firmware Version: ', Firmware)`
	`end`

4.4.7. GET FIRMWARE VERSION (ANTIQUATED)

Short GetFirmware()

4.4.8. SET ADDRESS

Short Set_Address(Short Address)

Sets the internal USB address which can be used in place of the serial number for future connections. The factory default for all units is 255.

Parameters

Variable	Description
Address	Required. An integer value from 1 to 255

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Example

Python	`status = MyPTE1.Set_Address(1)`
Visual Basic	`status = MyPTE1.Set_Address(1)`
Visual C++	`status = MyPTE1->Set_Address(1);`
Visual C#	`status = MyPTE1.Set_Address(1);`
MatLab	`status = MyPTE1.Set_Address(1);`

4.4.9. GET ADDRESS

Short Get_Address()

Returns the internal USB address which can be used in place of the serial number for future connections. The factory default for all units is 255.

Return Values

Value	Description
0	Command failed
1-255	Address of the switch matrix

Examples

Python	`status = MyPTE1.Get_Address()`
Visual Basic	`status = MyPTE1.Get_Address()`
Visual C++	`status = MyPTE1->Get_Address();`
Visual C#	`status = MyPTE1.Get_Address();`
MatLab	`status = MyPTE1.Get_Address();`

4.4.10. GET LIST OF CONNECTED SERIAL NUMBERS

Short Get_Available_SN_List(ByRef String SN_List)

Provides a list of serial numbers for all available devices.

Parameters

Variable	Description
SN_List	Required. String variable which will be updated with a list of all available serial numbers, separated by a single space character; for example, "11301020001 11301020002 11301020003".

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Example

Python	`status = MyPTE1.Get_Available_SN_List("")`
	`if status[0] > 0:`
	`  SN_List = str(status[1])`
	`  print("Connected devices:", SN_List)`

Visual Basic	`If MyPTE1.Get_Available_SN_List(SN_List) > 0 Then`
	`  MsgBox ("Connected devices: " & SN_List)`
	`End If`

Visual C++	`if (MyPTE1->Get_Available_SN_List(SN_List) > 0 )`
	`{`
	`  MessageBox::Show("Connected devices: " + SN_List);`
	`}`

Visual C#	`if (MyPTE1.Get_Available_SN_List(ref(SN_List)) > 0 )`
	`{`
	`  MessageBox.Show("Connected devices: " + SN_List);`
	`}`

MatLab	`[status, SN_List] = MyPTE1.Get_Available_SN_List('')`
	`if status > 0`
	`  h = msgbox('Connected devices: ', SN_List)`
	`end`

4.4.11. GET LIST OF AVAILABLE ADDRESSES

Short Get_Available_Address_List(ByRef String Add_List)

Provides a list of addresses for all available devices.

Parameters

Variable	Description
Add_List	Required. String variable which will be updated with a list of addresses separated by a single space character, for example, "5 101 254 255"

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Example

Python	`status = MyPTE1.Get_Available_Add_List("")`
	`if status[0] > 0:`
	`    Address_List = str(status[1])`
	`    print("Connected devices:", Address_List)`

Visual Basic	`If MyPTE1.Get_Available_Add_List(SN_List) > 0 Then`
	`    MsgBox ("Connected devices: " & Address_List)`
	`End If`

Visual C++	`if (MyPTE1->Get_Available_Add_List(Address_List) > 0 )`
	`{`
	`    MessageBox::Show("Connected devices: " + Address_List);`
	`}`

Visual C#	`if (MyPTE1.Get_Available_Add_List(ref(Address_List)) > 0 )`
	`{`
	`    MessageBox.Show("Connected devices: " + Address_List);`
	`}`

MatLab	`[status, Address_List] = MyPTE1.Get_Available_Add_List('')`
	`if status > 0`
	`    h = msgbox('Connected devices: ', Address_List)`
	`end`

4.4.12. GET TEMPERATURE

Float GetDeviceTemperature(Short TSensor)

Returns the internal temperature measured at 1 of the internal sensors (model dependent):

- RC-1SPDT-x (1 sensor)
- RC-2SPDT-x (2 sensors)
- RC-3SPDT-x (2 sensors)
- RC-4SPDT-x (2 sensors)
- RC-8SPDT-x (3 sensors)
- RC-1SP4T-x (1 sensor)
- RC-2SP4T-x (2 sensors)
- RC-1SP6T-x (1 sensor)
- RC-2SP6T-x (2 sensors)
- RC-2MTS-x (2 sensors)
- RC-3MTS-x (2 sensors)
- ZTRC-4SPDT-x (2 sensors)
- ZTRC-8SPDT-x (3 sensors)

Note: "+25.00" will be returned when polling a sensor which is not fitted

Parameters

Variable	Description
TSensor	Required. Short integer variable (1 to 3) to define which temperature sensor to read.

Return Values

Value	Description
Temperature	The device internal temperature in degrees Celsius

Examples

Python	`temp = MyPTE1.GetDeviceTemperature(1)`
Visual Basic	`temp = MyPTE1.GetDeviceTemperature(1)`
Visual C++	`temp = MyPTE1->GetDeviceTemperature(1);`
Visual C#	`temp = MyPTE1.GetDeviceTemperature(1);`
MatLab	`temp = MyPTE1.GetDeviceTemperature(1);`

4.4.13. GET HEAT ALARM

Short GetHeatAlarm()

Returns an alarm notification if any of the internal temperature sensors exceeds the factory programmed limits (45°C on the PCB or 48°C on the internal case).

Return Values

Value	Description
0	Temperature within normal range
1	Temperature exceeds specified limit

Examples

Python	`alarm = MyPTE1.GetHeatAlarm()`
	`if alarm > 0:`
	`  print("Temperature has exceeded specified limit")`

Visual Basic	`If MyPTE1.GetHeatAlarm() > 0 Then`
	`  MsgBox ("Temperature has exceeded specified limit")`
	`End If`

Visual C++	`if (MyPTE1->GetHeatAlarm() > 0)`
	`{`
	`  MessageBox::Show("Temperature has exceeded specified limit");`
	`}`

Visual C#	`if(MyPTE1.GetHeatAlarm() > 0)`
	`{`
	`  MessageBox.Show("Temperature has exceeded specified limit");`
	`}`

MatLab	`alarm = MyPTE1.GetHeatAlarm()`
	`if status > 0`
	`  h = msgbox('Temperature has exceeded specified limit')`
	`end`

4.4.14. GET FAN STATUS

Short Get_FAN_Indicator()

Checks whether the internal fan is currently operating.

Note: Does not apply to ZTRC-4SPDT-A18 which has no fans fitted

Return Values

Value	Description
0	Fan not currently operating
1	Fan operating

Examples

Python	`status = MyPTE1.Get_FAN_Indicator()`
Visual Basic	`status = MyPTE1.Get_FAN_Indicator()`
Visual C++	`status = MyPTE1->Get_FAN_Indicator();`
Visual C#	`status = MyPTE1.Get_FAN_Indicator();`
MatLab	`status = MyPTE1.Get_FAN_Indicator();`

4.4.15. GET SOFTWARE CONNECTION STATUS

Short GetConnectionStatus()

Confirms whether the USB connection to the device is active. This will be true if the Connect function (or similar) has previously been called.

Return Values

Value	Description
0	No connection
1	Switch matrix is connected

Examples

Python	`status = MyPTE1.GetConnectionStatus()`
Visual Basic	`status = MyPTE1.GetConnectionStatus()`
Visual C++	`status = MyPTE1->GetConnectionStatus();`
Visual C#	`status = MyPTE1.GetConnectionStatus();`
MatLab	`status = MyPTE1.GetConnectionStatus();`

4.4.16. GET USB CONNECTION STATUS

Short GetUSBConnectionStatus()

Checks whether the USB connection to the device is still active.

Return Values

Value	Description
0	No connection
1	USB connection to switch matrix is active

Examples

Python	`status = MyPTE1.GetUSBConnectionStatus()`
Visual Basic	`status = MyPTE1.GetUSBConnectionStatus()`
Visual C++	`status = MyPTE1->GetUSBConnectionStatus();`
Visual C#	`status = MyPTE1.GetUSBConnectionStatus();`
MatLab	`status = MyPTE1.GetUSBConnectionStatus();`

4.4.17. CHECK CONNECTION (ANTIQUATED)

Short Check_Connection()

This function is antiquated, GetUSBConnectionStatus should be used instead. Checks whether the USB connection to the device is active.

4.4.18. GET USB DEVICE NAME

String GetUSBDeviceName()

Identify the USB device name for direct communication.

Return Values

Value	Description
Name	Device name of the switch matrix

Examples

Python	`usbname = MyPTE1.GetUSBDeviceName()`
Visual Basic	`usbname = MyPTE1.GetUSBDeviceName()`
Visual C++	`usbname = MyPTE1->GetUSBDeviceName();`
Visual C#	`usbname = MyPTE1.GetUSBDeviceName();`
MatLab	`usbname = MyPTE1.GetUSBDeviceName();`

4.5. DLL – SCPI Communication

4.5.1. SEND SCPI COMMAND

Short Send_SCPI(String SndSTR, ByRef String RetSTR)

Sends an ASCII / SCPI command and returns the response. SCPI (Standard Commands for Programmable Instruments) is a common method for communicating with and controlling instrumentation products.

Requirements

Firmware version E3 or later

Parameters

Variable	Description
SndSTR	The SCPI command / query to send
RetSTR	String variable which will be updated with the attenuator's response to the command / query

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.Send_SCPI(":MN?", "")` `response = str(status[1])`
Visual Basic	`status = MyPTE1.Send_SCPI(":MN?", response)`
Visual C++	`status = MyPTE1->Send_SCPI(":MN?", response);`
Visual C#	`status = MyPTE1.Send_SCPI(":MN?", ref(response));`
MatLab	`[status, response] = MyPTE1.Send_SCPI(":MN?", response)`

4.6. DLL - Switch Control

4.6.1. SET INDIVIDUAL SPDT / TRANSFER SWITCH

Short Set_Switch(String SwitchName, Short Val)

Sets an individual SPDT or transfer switch. The switches are designated A to H, as labeled on the front panel (model dependent).

Applies To

Model	Serial Numbers
RC-xSPDT-x & ZTRC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers

Parameters

Variable	Description
SwitchName	String consisting of a single letter from "A" to "H", designating the specific SPDT switch is to operate.
Val	0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 (SPDT) Connect J1 <> J2 and J3 <> J4 (transfer switch)

Return Values

Data Type	Description
Short	Command failed
	Command completed successfully

Examples

Python	`status = MyPTE1.Set_Switch("A", 1)`
Visual Basic	`status = MyPTE1.Set_Switch("A", 1)`
Visual C++	`status = MyPTE1->Set_Switch("A", 1);`

Visual C#	`swname = "A";`
	`swstate = 1;`
	`status = MyPTE1.Set_Switch(ref(swname), ref(swstate));`

MatLab	`status = MyPTE1.Set_Switch('A', 1);`

4.6.2. SET ALL SPDT / TRANSFER SWITCHES

Short Set_SwitchesPort(Byte Val)

Sets all SPDT or transfer switches at once.

Applies To

Model	Serial Numbers
RC-xSPDT-x & ZTRC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers

Parameters

Variable	Description
Val	Each bit corresponds to a single switch, from MSB (switch "H") to LSB (switch "A"), as applicable. Each bit can take 0 or 1: 0 = Com <> 1 (SPDT) J1 <> J3 and J2 <> J4 (transfer switch) 1 = Com <> 2 (SPDT) J1 <> J2 and J3 <> J4 (transfer switch) For example: Val=5 (binary 00000101) sets switches "A" and "C" for Com to port 2 and all other switches (if applicable) for Com to port 1.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Example (Set switches B & C to state 2)

Python	`status = MyPTE1.Set_SwitchesPort(5)`
Visual Basic	`status = MyPTE1.Set_SwitchesPort(5)`
Visual C++	`status = MyPTE1->Set_SwitchesPort(5);`
Visual C#	`status = MyPTE1.Set_SwitchesPort(5);`
MatLab	`status = MyPTE1.Set_SwitchesPort(5);`

4.6.3. SET SINGLE SP4T SWITCH

Short Set_SP4T_COM_To(Byte Port)

Sets the state of the SP4T switch.

Applies To

Model	Serial Numbers
RC-xSP4T-x	All serial numbers (sets switch A for dual switch boxes)

Parameters

Variable	Description
Port	Integer value for the switch state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_SP4T_COM_To(3)`
Visual Basic	`status = MyPTE1.Set_SP4T_COM_To(3)`
Visual C++	`status = MyPTE1->Set_SP4T_COM_To(3);`
Visual C#	`status = MyPTE1.Set_SP4T_COM_To(3);`
MatLab	`status = MyPTE1.Set_SP4T_COM_To(3);`

4.6.4. SET DUAL SP4T – BOTH SWITCHES

Short Set_2SP4T_COM_To(Short P1, Short P2)

Sets the state of both SP4T switches at once.

Applies To

Model	Serial Numbers
RC-2SP4T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch A state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4
P2	Integer value for the switch B state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP4T_COM_To(3, 1)`
Visual Basic	`status = MyPTE1.Set_2SP4T_COM_To(3, 1)`
Visual C++	`status = MyPTE1->Set_2SP4T_COM_To(3, 1);`
Visual C#	`status = MyPTE1.Set_2SP4T_COM_To(3, 1);`
MatLab	`status = MyPTE1.Set_2SP4T_COM_To(3, 1);`

4.6.5. SET DUAL SP4T – SWITCH A

Short Set_2SP4T_COMA_To(Short P1)

Sets the state of switch A within a dual SP4T switch box.

Applies To

Model	Serial Numbers
RC-2SP4T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch A state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP4T_COMA_To(3)`
Visual Basic	`status = MyPTE1.Set_2SP4T_COMA_To(3)`
Visual C++	`status = MyPTE1->Set_2SP4T_COMA_To(3);`
Visual C#	`status = MyPTE1.Set_2SP4T_COMA_To(3);`
MatLab	`status = MyPTE1.Set_2SP4T_COMA_To(3);`

4.6.6. SET DUAL SP4T – SWITCH B

Short Set_2SP4T_COMB_To(Short P1)

Sets the state of switch B within a dual SP4T switch box.

Applies To

Model	Serial Numbers
RC-2SP4T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch B state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP4T_COMB_To(3)`
Visual Basic	`status = MyPTE1.Set_2SP4T_COMB_To(3)`
Visual C++	`status = MyPTE1->Set_2SP4T_COMB_To(3);`
Visual C#	`status = MyPTE1.Set_2SP4T_COMB_To(3);`
MatLab	`status = MyPTE1.Set_2SP4T_COMB_To(3);`

4.6.7. SET DUAL SP6T – BOTH SWITCHES

Short Set_2SP6T_COM_To(Short P1, Short P2)

Sets the state of both SP6T switches within a dual SP6T switch box.

Applies To

Model	Serial Numbers
RC-2SP6T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch A state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4 5 = Com connected to port 5 6 = Com connected to port 6
P2	Integer value for the switch B state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4 5 = Com connected to port 5 6 = Com connected to port 6

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP6T_COM_To(6, 1)`
Visual Basic	`status = MyPTE1.Set_2SP6T_COM_To(6, 1)`
Visual C++	`status = MyPTE1->Set_2SP6T_COM_To(6, 1);`
Visual C#	`status = MyPTE1.Set_2SP6T_COM_To(6, 1);`
MatLab	`status = MyPTE1.Set_2SP6T_COM_To(6, 1);`

4.6.8. SET DUAL SP6T – SWITCH A

Short Set_2SP6T_COMA_To(Short P1)

Sets the state of switch A within a dual SP6T switch box.

Applies To

Model	Serial Numbers
RC-2SP6T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch A state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4 5 = Com connected to port 5 6 = Com connected to port 6

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP6T_COMA_To(3)`
Visual Basic	`status = MyPTE1.Set_2SP6T_COMA_To(3)`
Visual C++	`status = MyPTE1->Set_2SP6T_COMA_To(3);`
Visual C#	`status = MyPTE1.Set_2SP6T_COMA_To(3);`
MatLab	`status = MyPTE1.Set_2SP6T_COMA_To(3);`

4.6.9. SET DUAL SP6T – SWITCH B

Short Set_2SP6T_COMB_To(Short P1)

Sets the state of switch B within a dual SP6T switch box.

Applies To

Model	Serial Numbers
RC-2SP6T-x	All serial numbers

Parameters

Variable	Description
P1	Integer value for the switch B state: 0 = All ports disconnected 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4 5 = Com connected to port 5 6 = Com connected to port 6

Return Values

Value	Description
0	Command failed or invalid switch state requested
1	Command completed successfully

Examples

Python	`status = MyPTE1.Set_2SP6T_COMB_To(3)`
Visual Basic	`status = MyPTE1.Set_2SP6T_COMB_To(3)`
Visual C++	`status = MyPTE1->Set_2SP6T_COMB_To(3);`
Visual C#	`status = MyPTE1.Set_2SP6T_COMB_To(3);`
MatLab	`status = MyPTE1.Set_2SP6T_COMB_To(3);`

4.6.10. GET ALL SPDT / TRANSFER SWITCH STATES

Short GetSwitchesStatus(ByRef Short StatusRet)

Returns the states of all SPDT or transfer switches in a switch box. The indicated status differs between switch type, see explanations below.

Applies To

Model	Serial Numbers
RC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers
ZTRC-xSPDT-x	All serial numbers

Parameters

Variable	Description
StatusRet	Integer variable that will be updated with the state of all switches. The value should be interpreted as a binary word, where each bit corresponds to a single switch, from MSB (switch "H") to LSB (switch "A"), as applicable. Each bit can take 0 or 1: 0 = Com <> 1 (SPDT) / J1 <> J3 and J2 <> J4 (transfer switch) 1 = Com <> 2 (SPDT) / J1 <> J2 and J3 <> J4 (transfer switch) For example: StatusRet = 12 (00001100 binary) indicates Bits 3 & 4 (switch C D) = 1, others = 0

Return Values (All Models)

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python

```
status = MyPTE1.GetSwitchesStatus(portvalue)
if status[0] > 0:
        portvalue = str(status[1])
        print(portvalue)
```

Visual Basic

```
If MyPTE1.GetSwitchesStatus(portvalue) > 0 Then
        MsgBox (portvalue)
End If
```

Visual C++

```
if (MyPTE1->GetSwitchesStatus(portvalue) > 0)
{
        MessageBox::Show(portvalue);
}
```

Visual C#

```
if(MyPTE1.GetSwitchesStatus(ref(portvalue)) > 0)
{
        MessageBox.Show(portvalue);
}
```

MatLab

```
[status, portvalue] = MyPTE1.GetSwitchesStatus('')
if status > 0
        h = msgbox(portvalue)
end
```

4.6.11. GET SP4T SWITCH STATE

Short Get_2SP4T_State(String sw)

Returns the state of an SP4T switch box.

Applies To

Model	Serial Numbers
RC-xSP4T-x	All serial numbers

Parameters

Variable	Description
sw	String to indicate which SP4T switch state to return, either “A” or “B”.

Return Values

Value	Description
-1	Command failed
0	Switch has all ports disconnected
1	Switch has Com port connected to port 1
2	Switch has Com port connected to port 2
3	Switch has Com port connected to port 3
4	Switch has Com port connected to port 4

Examples

Python	`state = MyPTE1.Get_2SP4T_State('A')`
Visual Basic	`state = MyPTE1.Get_2SP4T_State('A')`
Visual C++	`State = MyPTE1->Get_2SP4T_State('A');`
Visual C#	`State = MyPTE1.Get_2SP4T_State('A');`
MatLab	`State = MyPTE1.Get_2SP4T_State('A')`

4.6.12. GET SP6T SWITCH STATE

Short Get_2SP6T_State(String sw)

Returns the state of an SP6T switch box.

Applies To

Model	Serial Numbers
RC-2SP6T-x	All serial numbers

Parameters

Variable	Description
sw	String to indicate which SP6T switch state to return, either “A” or “B”.

Return Values

Value	Description
-1	Command failed
0	Switch has all ports disconnected
1	Switch has Com port connected to port 1
2	Switch has Com port connected to port 2
3	Switch has Com port connected to port 3
4	Switch has Com port connected to port 4
5	Switch has Com port connected to port 5
6	Switch has Com port connected to port 6

Examples

Python	`state = MyPTE1.Get_2SP6T_State('A')`
Visual Basic	`state = MyPTE1.Get_2SP6T_State('A')`
Visual C++	`State = MyPTE1->Get_2SP6T_State('A');`
Visual C#	`State = MyPTE1.Get_2SP6T_State('A');`
MatLab	`State = MyPTE1.Get_2SP6T_State('A')`

4.6.13. GET SPDT / TRANSFER SWITCH COUNT

Long GetSwitchCounter(String Sw)

Returns the number of switching cycles undertaken by an individual SPDT or transfer switch.

Applies To

Model	Serial Numbers
USB-1SPDT-A18	From 11309160001 (firmware version C3 required)
USB-2SPDT-A18	From 11311270010 (firmware version C3 required)
USB-3SPDT-A18	From 11310100001 (firmware version C3 required)
USB-4SPDT-A18	From 11310100009 (firmware version C3 required)
USB-8SPDT-A18	From 11309290001 (firmware version C3 required)
RC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers
ZTRC-xSPDT-x	All serial numbers

Parameters

Variable	Description
Sw	The switch name, from “A” to “H” (model dependent)

Return Values

Value	Description
-1	Command failed
Count	The number of switch cycles for the specified switch

Examples

Python	`state = MyPTE1.GetSwitchCounter('A')`
Visual Basic	`state = MyPTE1.GetSwitchCounter('A')`
Visual C++	`State = MyPTE1->GetSwitchCounter('A');`
Visual C#	`State = MyPTE1.GetSwitchCounter('A');`
MatLab	`State = MyPTE1.GetSwitchCounter('A')`

4.6.14. GET ALL SWITCH COUNTS

Short GetAllSwitchCounters(Long Swc[])

Returns the number of switching cycles undertaken by each individual switch within an SPDT or transfer switch box. For SP4T switches, the return indicates the number of times that the Com port has connected to each port.

Applies To

Model	Serial Numbers
USB-1SPDT-A18	From 11309160001 (firmware version C3 required)
USB-2SPDT-A18	From 11311270010 (firmware version C3 required)
USB-3SPDT-A18	From 11310100001 (firmware version C3 required)
USB-4SPDT-A18	From 11310100009 (firmware version C3 required)
USB-8SPDT-A18	From 11309290001 (firmware version C3 required)
USB-1SP4T-A18	From 11310100001 (firmware version C3 required)
RC-xSPDT-x	All serial numbers
RC-xSP4T-x	All serial numbers

Parameters

Variable	Description
Swc[]	8 element array to be updated with the switch counts. For SPDT / transfer switch boxes the array contains 1 integer value for each switch. For SP4T switch boxes the array contains 4 integer values per switch (each value containing the count for a single switch port). For example, RC-4SPDT-A18 may return the below, indicating 500 cycles for each switch: Swc[] = {500, 500, 500, 500, 0, 0, 0, 0} RC-2SP4T-A18 may return the below, indicating 500 cycles for switch A, port 1, 400 for switch B port 2, and so on: Swc[] = {500, 200, 200, 100, 0, 0, 0, 0}

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	response = MyPTE1.GetAllSwitchCounters("") if response[0] > 0: SwitchCounts = response[1] print('SPDT A Count:', SwitchCounts[0]) print('SPDT B Count:', SwitchCounts[1])

Visual Basic	response = MyPTE1.GetAllSwitchCounters(SwitchCounts)
	If response > 0 Then
	MsgBox ("SPDT A Count: " & SwitchCounts(0))
	MsgBox ("SPDT B Count: " & SwitchCounts(1))
	End If

Visual C++	response = MyPTE1->GetAllSwitchCounters(SwitchCounts);
	if (response > 0)
	{
	MessageBox::Show("SPDT A Count: " + SwitchCounts[0]);
	MessageBox::Show("SPDT B Count: " + SwitchCounts[1]);
	}

Visual C#	response = MyPTE1.GetAllSwitchCounters(ref(SwitchCounts));
	if (response > 0)
	{
	MessageBox.Show("SPDT A Count: " + SwitchCounts[0]);
	MessageBox.Show("SPDT B Count: " + SwitchCounts[1]);
	}

MatLab	[status, SwitchCounts] = MyPTE1.GetAllSwitchCounters(SwitchCounts) if status > 0 h = msgbox('SPDT A Count: ', SwitchCounts(0)) h = msgbox('SPDT B Count: ', SwitchCounts(1)) end

4.6.15. SET POWER-UP MODE - LAST SWITCH STATES

Int OnPowerUp_LastState_ON()

Sets the switch box to power-up with all switches in the last saved switch state.

Requirements

Firmware version D2 or later

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.OnPowerUp_LastState_ON()`
Visual Basic	`status = MyPTE1.OnPowerUp_LastState_ON()`
Visual C++	`status = MyPTE1->OnPowerUp_LastState_ON();`
Visual C#	`status = MyPTE1.OnPowerUp_LastState_ON();`
MatLab	`status = MyPTE1.OnPowerUp_LastState_ON()`

4.6.16. SET POWER-UP MODE - DEFAULT SWITCH STATES

Int OnPowerUp_LastState_OFF()

Sets the switch box to power up with all switches in the default state (SP4T and SP6T switches with all ports disconnected and SPDT switches with Com connected to port 1).

Requirements

Firmware version D2 or later

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.OnPowerUp_LastState_OFF()`
Visual Basic	`status = MyPTE1.OnPowerUp_LastState_OFF()`
Visual C++	`status = MyPTE1->OnPowerUp_LastState_OFF();`
Visual C#	`status = MyPTE1.OnPowerUp_LastState_OFF();`
MatLab	`status = MyPTE1.OnPowerUp_LastState_OFF()`

4.6.17. GET POWER-UP MODE

Int Get_OnPowerUp_LastState_Indicator()

Indicates whether or not the switch box will power-up with all switches in the last saved state or in the default state (SP4T and SP6T switches with all ports disconnected and SPDT switches with Com connected to port 1).

Requirements

Firmware version D2 or later

Return Values

Value	Description
0	Switch box will power-up with all switches in the last saved state
1	Switch box will power-up with all switches in the default state

Examples

Python	`state = MyPTE1.Get_OnPowerUp_LastState_Indicator()`
Visual Basic	`state = MyPTE1.Get_OnPowerUp_LastState_Indicator()`
Visual C++	`state = MyPTE1->Get_OnPowerUp_LastState_Indicator();`
Visual C#	`state = MyPTE1.Get_OnPowerUp_LastState_Indicator();`
MatLab	`state = MyPTE1.Get_OnPowerUp_LastState_Indicator()`

4.6.18. SAVE SWITCH COUNTERS & STATES

Int InitiateStoreSCounters()

Transfers the latest switch counters and switch states from temporary to permanent memory. During normal operation, this data is internally stored in volatile memory but automatically updated into permanent memory every 3 minutes. This command should be sent following completion of all switching routines and prior to powering off the switch in order to ensure that the data is permanently saved.

Requirements

Firmware version D1 or later

Return Values

Value	Description
0	Command failed (or has already been sent by the user within the last 3 minutes)
1	Command completed successfully

Examples

Python	`status = MyPTE1.InitiateStoreSCounters()`
Visual Basic	`status = MyPTE1.InitiateStoreSCounters()`
Visual C++	`status = MyPTE1->InitiateStoreSCounters();`
Visual C#	`status = MyPTE1.InitiateStoreSCounters();`
MatLab	`status = MyPTE1.InitiateStoreSCounters()`

4.7. DLL - Ethernet Configuration

These functions provide a method of configuring the device's Ethernet IP settings. The controller must be reset after updating Ethernet parameters in order to load the new configuration, this can be achieved with a power cycle or by using the ResetDevice function.

Refer to Ethernet Control API for additional details on the Ethernet configuration and default behavior.

4.7.1. GET ETHERNET CONFIGURATION

int GetEthernet_CurrentConfig
(ByRef int IP1, ByRef int IP2, ByRef int IP3, ByRef int IP4,
ByRef int Mask1, ByRef int Mask2, ByRef int Mask3, ByRef int Mask4,
ByRef int Gateway1, ByRef int Gateway2, ByRef int Gateway3, ByRef int Gateway4)

Returns the IP configuration that is currently use, either the static IP entered by the user, or the server assigned dynamic IP configuration when DHCP is enabled.

Parameters

Variable	Description
IP1	Required. Integer variable which will be updated with the first (highest order) octet of the IP address.
IP2	Required. Integer variable which will be updated with the second octet of the IP address.
IP2	Required. Integer variable which will be updated with the third octet of the IP address.
IP4	Required. Integer variable which will be updated with the last (lowest order) octet of the IP address.
Mask1	Required. Integer variable which will be updated with the first (highest order) octet of the subnet mask.
Mask2	Required. Integer variable which will be updated with the second octet of the subnet mask.
Mask3	Required. Integer variable which will be updated with the third octet of the subnet mask.
Mask4	Required. Integer variable which will be updated with the last (lowest order) octet of the subnet mask.
Gateway1	Required. Integer variable which will be updated with the first (highest order) octet of the subnet mask.
Gateway2	Required. Integer variable which will be updated with the second octet of the network gateway.
Gateway3	Required. Integer variable which will be updated with the third octet of the network gateway.
Gateway4	Required. Integer variable which will be updated with the last (lowest order) octet of the network gateway.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_CurrentConfig("", "", "", "", "", "", "", "", "", "", "", "")`
	`if status[0] > 0:`
	`  print("IP:", str(status[1]), str(status[2]), str(status[3]), str(status[4]))`
	`  print("Mask:", str(status[1]), str(status[2]), str(status[3]), str(status[4]))`
	`  print("Gateway:", str(status[1]), str(status[2]), str(status[3]), str(status[4]))`

Visual Basic	`If MyPTE1.GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4, M1, M2, M3, M4,`
	`  G1, G2, G3, G4) > 0 Then`
	`  MsgBox ("IP: " & IP1 & "." & IP2 & "." & IP3 & "." & IP4)`
	`  MsgBox ("Mask: " & M1 & "." & M2 & "." & M3 & "." & M4)`
	`  MsgBox ("Gateway: " & G1 & "." & G2 & "." & G3 & "." & G4)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4, M1, M2, M3, M4,`
	`  GW1, GW2, GW3, GW4) > 0)`
	`{`
	`  MessageBox::Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`  MessageBox::Show("Mask: " + M1 + "." + M2 + "." + M3 + "." + M4);`
	`  MessageBox::Show("Gateway: " + GW1 + "." + GW2 + "." + GW3 + "." + GW4);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_CurrentConfig(ref(IP1), ref(IP2), ref(IP3), ref(IP4),`
	`  ref(M1), ref(M2), ref(M3), ref(M4), ref(GW1), ref(GW2), ref(GW3), ref(GW4)) > 0)`
	`{`
	`  MessageBox.Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`  MessageBox.Show("Mask: " + M1 + "." + M2 + "." + M3+ "." + M4);`
	`  MessageBox.Show("Gateway: " + GW1 + "." + GW2 + "." + GW3 + "." + GW4);`
	`}`

MatLab	`[status, IP1, IP2, IP3, IP4, M1, M2, M3, M4, GW1, GW2, GW3, GW4]`
	`  = MyPTE1.GetEthernet_CurrentConfig('', '', '', '', '', '', '', '', '', '', '', '')`
	`if status > 0`
	`  h = msgbox("IP: ", IP1, ".", IP2, ".", IP3, ".", IP4)`
	`  h = msgbox("Mask: ", M1, ".", M2, ".", M3, ".", M4)`
	`  h = msgbox("Gateway: ", M1, ".", M2, ".", M3, ".", M4)`
	`end`

4.7.2. GET DHCP STATUS

int GetEthernet_UseDHCP()

Indicates whether DHCP (dynamic host control protocol) is currently enabled. When disabled, the device will attempt to connect using the user-entered static IP parameters.

Return Values

Value	Description
0	DHCP not in use (IP settings are static and manually configured)
1	DHCP in use (IP settings are assigned automatically by the network)

Examples

Python	`response = MyPTE1.GetEthernet_UseDHCP()`
Visual Basic	`response = MyPTE1.GetEthernet_UseDHCP()`
Visual C++	`response = MyPTE1->GetEthernet_UseDHCP();`
Visual C#	`response = MyPTE1.GetEthernet_UseDHCP();`
MatLab	`response = MyPTE1.GetEthernet_UseDHCP()`

4.7.3. USE DHCP

int SaveEthernet_UseDHCP(int UseDHCP)

Enables or disables DHCP (dynamic host control protocol). When disabled, the device will attempt to connect using the user-entered static IP parameters. By default, DHCP is enabled.

Parameters

Variable	Description
0	DHCP disabled (static IP settings used)
1	DHCP enabled (IP setting assigned by network)

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_UseDHCP(1)`
Visual Basic	`status = MyPTE1.SaveEthernet_UseDHCP(1)`
Visual C++	`status = MyPTE1->SaveEthernet_UseDHCP(1);`
Visual C#	`status = MyPTE1.SaveEthernet_UseDHCP(1);`
MatLab	`status = MyPTE1.SaveEthernet_UseDHCP(1);`

4.7.4. GET IP ADDRESS

int GetEthernet_IPAddress(ByRef int b1, ByRef int b2, ByRef int b3, ByRef int b4)

Returns the user-entered static IP address.

Parameters

Variable	Description
IP1	Required. Integer variable which will be updated with the first (highest order) octet of the IP address (for example "192" for the IP address "192.168.1.0").
IP2	Required. Integer variable which will be updated with the second octet of the IP address (for example "168" for the IP address "192.168.1.0").
IP2	Required. Integer variable which will be updated with the third octet of the IP address (for example "1" for the IP address "192.168.1.0").
IP4	Required. Integer variable which will be updated with the last (lowest order) octet of the IP address (for example "0" for the IP address "192.168.1.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_IPAddress("", "", "", "")`
	`if status[0] > 0:`
	`    print("IP:", str(status[1]), str(status[2]), str(status[3]), str(status[4]))`

Visual Basic	`If MyPTE1.GetEthernet_IPAddress(IP1, IP2, IP3, IP4) > 0 Then`
	`        MsgBox ("IP: " & IP1 & "." & IP2 & "." & IP3 & "." & IP4)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_IPAddress(IP1, IP2, IP3, IP4) > 0)`
	`{`
	`        MessageBox::Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_IPAddress(ref(IP1), ref(IP2), ref(IP3), ref(IP4)) > 0)`
	`{`
	`        MessageBox.Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

MatLab	`[status, IP1, IP2, IP3, IP4] = MyPTE1.GetEthernet_IPAddress('', '', '', '')`
	`if status > 0`
	`        h = msgbox("IP: ", IP1, ".", IP2, ".", IP3, ".", IP4)`
	`end`

4.7.5. SAVE IP *ADDRESS*

int SaveEthernet_IPAddress(int b1, int b2, int b3, int b4)

Sets the static IP address to be used when DHCP is disabled.

Parameters

Variable	Description
IP1	Required. First (highest order) octet of the IP address to set (for example "192" for the IP address "192.168.1.0").
IP2	Required. Second octet of the IP address to set (for example "168" for the IP address "192.168.1.0").
IP2	Required. Third octet of the IP address to set (for example "1" for the IP address "192.168.1.0").
IP4	Required. Last (lowest order) octet of the IP address to set (for example "0" for the IP address "192.168.1.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0)`
Visual Basic	`status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0)`
Visual C++	`status = MyPTE1->SaveEthernet_IPAddress(192, 168, 1, 0);`
Visual C#	`status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0);`
MatLab	`status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0);`

4.7.6. GET MAC ADDRESS

int GetEthernet_MACAddress(ByRef int MAC1, ByRef int MAC2, ByRef int MAC3,
ByRef int MAC4,ByRef int MAC5, ByRef int MAC6)

Returns the physical MAC (media access control) address of the device.

Parameters

Variable	Description
MAC1	Required. Integer variable which will be updated with the decimal value of the first numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC1=11
MAC2	Required. Integer variable which will be updated with the decimal value of the second numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC2=47
MAC3	Required. Integer variable which will be updated with the decimal value of the third numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC3=165
MAC4	Required. Integer variable which will be updated with the decimal value of the fourth numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC4=103
MAC5	Required. Integer variable which will be updated with the decimal value of the fifth numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC5=137
MAC6	Required. Integer variable which will be updated with the decimal value of the last numeric group of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC6=171

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_MACAddress("", "", "", "", "", "")`
	`if status[0] > 0:`
	`    print("MAC:", str(status[1]), str(status[2]), str(status[3]),`
	`str(status[4]), str(status[5]), str(status[6]))`

Visual Basic	`If MyPTE1.GetEthernet_MACAddress(M1, M2, M3, M4, M5, M6) > 0 Then`
	`    MsgBox ("MAC: " & M1 & "." & M2 & "." & M3 & "." & M4 & "." & M5 & "." & M6)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_MACAddress(M1, M2, M3, M4, M5, M6) > 0)`
	`{`
	`MessageBox::Show("Mask: " + M1 + "." + M2 + "." + M3`
	`+ "." + M4 + "." + M5 + "." + M6);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_MACAddress(ref(M1), ref(M2), ref(M3), ref(M4),`
	`ref(M5), ref(M6)) > 0)`
	`{`
	`MessageBox.Show("Mask: " + M1 + "." + M2 + "." + M3`
	`+ "." + M4 + "." + M5 + "." + M6);`
	`}`

MatLab	`[status, M1, M2, M3, M4, M5, M6]`
	`= MyPTE1.GetEthernet_MACAddress('', '', '', '', '', '')`
	`if status > 0`
	`h = msgbox("Mask: ", M1, ".", M2, ".", M3, ".", M4, ".", M5, ".", M6)`
	`end`

4.7.7. GET NETWORK GATEWAY

int GetEthernet_NetworkGateway(ByRef int b1, ByRef int b2, ByRef int b3, ByRef int b4)

Returns the user-entered network gateway IP address.

Parameters

Variable	Description
IP1	Required. Integer variable which will be updated with the first (highest order) octet of the IP address (for example "192" for the IP address "192.168.1.0").
IP2	Required. Integer variable which will be updated with the second octet of the IP address (for example "168" for the IP address "192.168.1.0").
IP2	Required. Integer variable which will be updated with the third octet of the IP address (for example "1" for the IP address "192.168.1.0").
IP4	Required. Integer variable which will be updated with the last (lowest order) octet of the IP address (for example "0" for the IP address "192.168.1.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_NetworkGateway("", "", "", "")`
	`if status[0] > 0:`
	`    print("IP:", str(status[1]), str(status[2]), str(status[3]), str(status[4]))`

Visual Basic	`If MyPTE1.GetEthernet_NetworkGateway(IP1, IP2, IP3, IP4) > 0 Then`
	`        MsgBox ("IP: " & IP1 & "." & IP2 & "." & IP3 & "." & IP4)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_NetworkGateway(IP1, IP2, IP3, IP4) > 0)`
	`{`
	`        MessageBox::Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_NetworkGateway(ref(IP1), ref(IP2), ref(IP3),ref(IP4)) > 0)`
	`{`
	`        MessageBox.Show("IP: " + IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

MatLab	`[status, IP1, IP2, IP3, IP4] = MyPTE1.GetEthernet_NetworkGateway('', '', '', '')`
	`if status > 0`
	`        h = msgbox("IP: ", IP1, ".", IP2, ".", IP3, ".", IP4)`
	`end`

4.7.8. SAVE NETWORK GATEWAY

int SaveEthernet_NetworkGateway(int b1, int b2, int b3, int b4)

Sets the IP address of the network gateway to be used when DHCP is disabled.

Parameters

Variable	Description
IP1	Required. First (highest order) octet of the network gateway IP address (for example "192" for the IP address "192.168.1.0").
IP2	Required. Second octet of the network gateway IP address (for example "168" for the IP address "192.168.1.0").
IP2	Required. Third octet of the network gateway IP address (for example "1" for the IP address "192.168.1.0").
IP4	Required. Last (lowest order) octet of the network gateway IP address (for example "0" for the IP address "192.168.1.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0)`
Visual Basic	`status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0)`
Visual C++	`status = MyPTE1->SaveEthernet_NetworkGateway(192, 168, 1, 0);`
Visual C#	`status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0);`
MatLab	`status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0);`

4.7.9. GET SUBNET MASK

int GetEthernet_SubNetMask(ByRef int b1, ByRef int b2, ByRef int b3, ByRef int b4)

Returns the user-entered subnet mask.

Parameters

Variable	Description
b1	Required. Integer variable which will be updated with the first (highest order) octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
b2	Required. Integer variable which will be updated with the second octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
b2	Required. Integer variable which will be updated with the third octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
b4	Required. Integer variable which will be updated with the last (lowest order) octet of the subnet mask (for example "0" for the subnet mask "255.255.255.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_SubNetMask("", "", "", "")`
	`if status[0] > 0:`
	`  print(str(status[1]), str(status[2]), str(status[3]), str(status[4]))`

Visual Basic	`If MyPTE1.GetEthernet_SubNetMask(IP1, IP2, IP3, IP4) > 0 Then`
	`  MsgBox (IP1 & "." & IP2 & "." & IP3 & "." & IP4)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_SubNetMask(IP1, IP2, IP3, IP4) > 0)`
	`{`
	`  MessageBox::Show(IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_SubNetMask(ref(IP1), ref(IP2), ref(IP3), ref(IP4)) > 0)`
	`{`
	`  MessageBox.Show(IP1 + "." + IP2 + "." + IP3 + "." + IP4);`
	`}`

MatLab	`[status, IP1, IP2, IP3, IP4] = MyPTE1.GetEthernet_SubNetMask('', '', '', '')`
	`if status > 0`
	`  h = msgbox(IP1, ".", IP2, ".", IP3, ".", IP4)`
	`end`

4.7.10. SAVE SUBNET MASK

int SaveEthernet_SubnetMask(int b1, int b2, int b3, int b4)

Sets the subnet mask to be used when DHCP is disabled.

Parameters

Variable	Description
IP1	Required. First (highest order) octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
IP2	Required. Second octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
IP2	Required. Third octet of the subnet mask (for example "255" for the subnet mask "255.255.255.0").
IP4	Required. Last (lowest order) octet of the subnet mask (for example "0" for the subnet mask "255.255.255.0").

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0)`
Visual Basic	`status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0)`
Visual C++	`status = MyPTE1->SaveEthernet_SubnetMask(255, 255, 255, 0);`
Visual C#	`status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0);`
MatLab	`status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0);`

4.7.11. GET TCP/IP PORT

int GetEthernet_TCPIPPort(ByRef int port)

Returns the TCP/IP port in use for HTTP communication. The default is port 80.

Parameters

Variable	Description
port	Required. Integer variable which will be updated with the TCP/IP port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_TCPIPPort("")`
	`if status[0] > 0:`
	`    port = str(status[1])`
	`    print(port)`

Visual Basic	`If MyPTE1.GetEthernet_TCPIPPort(port) > 0 Then`
	`    MsgBox (port)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_TCPIPPort(port) > 0)`
	`{`
	`    MessageBox::Show(port);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_TCPIPPort(ref(port)) > 0)`
	`{`
	`    MessageBox.Show(port);`
	`}`

MatLab	`[status, port] = MyPTE1.GetEthernet_TCPIPPort('')`
	`if status > 0`
	`    h = msgbox(port)`
	`end`

4.7.12. SET HTTP PORT & ENABLE / DISABLE HTTP

int SaveEthernet_TCPIPPort(int port)

Sets the TCP / IP port to be used for HTTP communication. The default is port 80. Set port 0 or 65535 to disable HTTP (requires firmware F1 or later) or any valid port to enable.

Parameters

Variable	Description
port	Required. Numeric value of the TCP/IP port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_TCPIPPort(70)`
Visual Basic	`status = MyPTE1.SaveEthernet_TCPIPPort(70)`
Visual C++	`status = MyPTE1->SaveEthernet_TCPIPPort(70);`
Visual C#	`status = MyPTE1.SaveEthernet_TCPIPPort(70);`
MatLab	`status = MyPTE1.SaveEthernet_TCPIPPort(70);`

4.7.13. GET TELNET PORT

int GetEthernet_TelnetPort(ByRef int port)

Returns the TCP/IP port in use for Telnet communication. The default is port 23.

Parameters

Variable	Description
port	Required. Integer variable which will be updated with the Telnet port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_TelnetPort("")`
	`if status[0] > 0:`
	`  port = str(status[1])`
	`  print(port)`

Visual Basic	`If MyPTE1.GetEthernet_TelnetPort(port) > 0 Then`
	`  MsgBox (port)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_TelnetPort(port) > 0)`
	`{`
	`  MessageBox::Show(port);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_TelnetPort(ref(port)) > 0)`
	`{`
	`  MessageBox.Show(port);`
	`}`

MatLab	`[status, port] = MyPTE1.GetEthernet_TelnetPort('')`
	`if status > 0`
	`  h = msgbox(port)`
	`end`

4.7.14. SET TELNET PORT & ENABLE / DISABLE TELNET

int SaveEthernet_TelnetPort(int port)

Sets the TCP / IP port to be used for Telnet communication. The default is port 23. Set port 0 or 65535 to disable Telnet (requires firmware F1 or later) or any valid port to enable.

Parameters

Variable	Description
port	Required. Numeric value of the Telnet port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_TelnetPort(21)`
Visual Basic	`status = MyPTE1.SaveEthernet_TelnetPort(21)`
Visual C++	`status = MyPTE1->SaveEthernet_TelnetPort(21);`
Visual C#	`status = MyPTE1.SaveEthernet_TelnetPort(21);`
MatLab	`status = MyPTE1.SaveEthernet_TelnetPort(21);`

4.7.15. GET SSH PORT

int GetEthernet_SSHPort(ByRef int port)

Returns the TCP/IP port in use for SSH communication. The default is port 22.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Parameters

Variable	Description
port	Required. Integer variable which will be updated with the SSH port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_SSHPort("")`
	`if status[0] > 0:`
	`  port = str(status[1])`
	`  print(port)`

Visual Basic	`If MyPTE1.GetEthernet_SSHPort(port) > 0 Then`
	`  MsgBox (port)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_SSHPort(port) > 0)`
	`{`
	`  MessageBox::Show(port);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_SSHPort(ref(port)) > 0)`
	`{`
	`  MessageBox.Show(port);`
	`}`

MatLab	`[status, port] = MyPTE1.GetEthernet_SSHPort('')`
	`if status > 0`
	`  h = msgbox(port)`
	`end`

4.7.16. SAVE SSH PORT

int SaveEthernet_SSHPort(int port)

Sets the TCP / IP port to be used for SSH communication. The default is port 22.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Parameters

Variable	Description
port	Required. Numeric value of the SSH port.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_SSHPort(22)`
Visual Basic	`status = MyPTE1.SaveEthernet_SSHPort(22)`
Visual C++	`status = MyPTE1->SaveEthernet_SSHPort(22);`
Visual C#	`status = MyPTE1.SaveEthernet_SSHPort(22);`
MatLab	`status = MyPTE1.SaveEthernet_SSHPort(22);`

4.7.17. GET SSH LOGIN NAME

int GetEthernet_SSHLoginName(ByRef string SSHLoginName)

Returns the login name for SSH communication.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Parameters

Variable	Description
SSHLoginName	String variable which will be updated with the SSH login name.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_SSHLoginName("")`
	`if status[0] > 0:`
	`  SSHLoginName = str(status[1])`
	`  print(SSHLoginName)`

Visual Basic	`If MyPTE1.GetEthernet_SSHLoginName(SSHLoginName) > 0 Then`
	`  MsgBox (SSHLoginName)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_SSHLoginName(SSHLoginName) > 0)`
	`{`
	`  MessageBox::Show(SSHLoginName);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_SSHLoginName(ref(SSHLoginName)) > 0)`
	`{`
	`  MessageBox.Show(SSHLoginName);`
	`}`

MatLab	`[status, SSHLoginName] = MyPTE1.GetEthernet_SSHLoginName('')`
	`if status > 0`
	`  h = msgbox(SSHLoginName)`
	`end`

4.7.18. SAVE SSH LOGIN NAME

int SaveEthernet_SSHLoginName(string SSHLoginName)

Sets the login name for SSH communication.

Note: SSH communication is not supported as standard on all models. Please contact testsolutions@minicircuits.com for details.

Parameters

Variable	Description
SSHLoginName	The login name to set

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_SSHLoginName('ssh_user')`
Visual Basic	`status = MyPTE1.SaveEthernet_SSHLoginName('ssh_user')`
Visual C++	`status = MyPTE1->SaveEthernet_SSHLoginName('ssh_user');`
Visual C#	`status = MyPTE1.SaveEthernet_SSHLoginName('ssh_user');`
MatLab	`status = MyPTE1.SaveEthernet_SSHLoginName('ssh_user');`

4.7.19. GET PASSWORD REQUIREMENT

int GetEthernet_UsePWD()

Indicates whether the password is currently enabled for HTTP / Telnet (the password is always required for SSH).

Return Values

Value	Description
0	Password not required
1	Password required

Examples

Python	`response = MyPTE1.GetEthernet_UsePWD()`
Visual Basic	`response = MyPTE1.GetEthernet_UsePWD()`
Visual C++	`response = MyPTE1->GetEthernet_UsePWD();`
Visual C#	`response = MyPTE1.GetEthernet_UsePWD();`
MatLab	`response = MyPTE1.GetEthernet_UsePWD()`

4.7.20. SET PASSWORD REQUIREMENT

int SaveEthernet_UsePWD(int UsePwd)

Enables or disables the password requirement for HTTP / Telnet (the password is always required for SSH).

Parameters

Variable	Description
0	Password not required
1	Password required

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_UsePWD(1)`
Visual Basic	`status = MyPTE1.SaveEthernet_UsePWD(1)`
Visual C++	`status = MyPTE1->SaveEthernet_UsePWD(1);`
Visual C#	`status = MyPTE1.SaveEthernet_UsePWD(1);`
MatLab	`status = MyPTE1.SaveEthernet_UsePWD(1);`

4.7.21. GET PASSWORD

int GetEthernet_PWD(ByRef string Pwd)

Returns the current password for SSH / HTTP / Telnet communication.

Parameters

Variable	Description
Pwd	Required. String variable which will be updated with the password.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.GetEthernet_PWD("")`
	`if status[0] > 0:`
	`  pwd = str(status[1])`
	`  print(pwd)`

Visual Basic	`If MyPTE1.GetEthernet_PWD(pwd) > 0 Then`
	`  MsgBox (pwd)`
	`End If`

Visual C++	`if (MyPTE1->GetEthernet_PWD(pwd) > 0)`
	`{`
	`  MessageBox::Show(pwd);`
	`}`

Visual C#	`if (MyPTE1.GetEthernet_PWD(ref(pwd)) > 0)`
	`{`
	`  MessageBox.Show(pwd);`
	`}`

MatLab	`[status, pwd] = MyPTE1.GetEthernet_PWD('')`
	`if status > 0`
	`  h = msgbox(pwd)`
	`end`

4.7.22. SET PASSWORD

int SaveEthernet_PWD(string Pwd)

Sets the password to be used for SSH / HTTP / Telnet communication. The password will only apply for HTTP / Telnet after enabling using Set Password Requirement.

Parameters

Variable	Description
Pwd	Password for Ethernet communication (not case sensitive, 20 characters maximum)

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`status = MyPTE1.SaveEthernet_PWD("123")`
Visual Basic	`status = MyPTE1.SaveEthernet_PWD("123")`
Visual C++	`status = MyPTE1->SaveEthernet_PWD("123");`
Visual C#	`status = MyPTE1.SaveEthernet_PWD("123");`
MatLab	`status = MyPTE1.SaveEthernet_PWD("123");`

4.7.23. RESET DEVICE

byte ResetDevice()

Called after updating Ethernet parameters, to reset the controller and reload with the updated Ethernet configuration.

Return Values

Value	Description
0	Command failed
1	Command completed successfully

Examples

Python	`MyPTE1.ResetDevice()`
Visual Basic	`MyPTE1.ResetDevice()`
Visual C++	`MyPTE1->ResetDevice();`
Visual C#	`MyPTE1.ResetDevice();`
MatLab	`MyPTE1.ResetDevice();`

5. USB Control via Direct Programming (Linux)

Mini-Circuits' API DLL files require a programming environment which supports either .NET or ActiveX. Where this is not available (for example on a Linux operating system) the alternative method is "direct" USB programming using USB interrupts

5.1. USB Interrupt Code Concept

To open a USB connection to the Mini-Circuits RF switch matrix series, the Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID: 0x20CE
- Switch Matrix Product ID: 0x22

Communication with the switch matrix is carried out by way of USB Interrupt. The transmitted and received buffer sizes are 64 bytes each:

- Transmit Array = [Byte 0][Byte1][Byte2]...[Byte 63]
- Returned Array = [Byte 0][Byte1][Byte2]...[Byte 63]

In most cases, the full 64 byte buffer size is not needed so any unused bytes become "don't care" bytes; they can take on any value without affecting the operation of the switch matrix.

Worked examples can be found in the Programming Examples & Troubleshooting Guide, available from the Mini-Circuits website. The examples make use of standard USB and HID (Human Interface Device) APIs to interface with the system.

5.2. Interrupts - Core Commands / Queries

Description	Code (Byte 0)	Comments
Get Device Model Name	40	
Get Device Serial Number	41	
Set Single SPDT / Transfer Switch	1	Switch A
	2	Switch B
	3	Switch C
	4	Switch D
	5	Switch E
	6	Switch F
	7	Switch G
	8	Switch H
Set All SPDT / Transfer Switches	9	
Set All SP4T Switches	9	
Set SP6T Switch	12	
Get All SPDT / Transfer Switch States	15	
Get All SP4T Switch States	15	
Get SP6T Switch State	13	
Set Power-Up Mode	89	
Get Power-UP Mode	90	
Get Firmware	99	
Check Internal Temperature	114	Sensor 1
	115	Sensor 2
	118	Sensor 3
Get 24V DC Power Indicator	116	
Get Heat Alarm	117	
Get Fan Status	119	
Get Switch Counter	17	
Get All Switch Counters	18	
Save Switch Counters & States	88	
Send SCPI Command	42	

5.2.1. GET DEVICE MODEL NAME

Returns the Mini-Circuits part number of the connected switch matrix.

Transmit Array

Byte	Data	Description
0	40	Interrupt code for Get Device Model Name
1 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	40	Interrupt code for Get Device Model Name
1 to (n-1)	Model Name	Series of bytes containing the ASCII code for each character in the model name
n	0	Zero value byte to indicate the end of the model name
(n+1) to 63	Not significant	"Don't care" bytes, can be any value

Example

The following array would be returned for Mini-Circuits' USB-4SPDT-A18 switch matrix (see Appendix A for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	40	Interrupt code for Get Device Model Name
1	85	ASCII character code for U
2	83	ASCII character code for S
3	42	ASCII character code for B
4	45	ASCII character code for -
5	52	ASCII character code for 4
6	83	ASCII character code for S
7	80	ASCII character code for P
8	68	ASCII character code for D
9	84	ASCII character code for T
10	45	ASCII character code for -
11	65	ASCII character code for A
12	49	ASCII character code for 1
13	56	ASCII character code for 8
14	0	Zero value byte to indicate end of string

See Also

Get Device Serial Number

5.2.2. GET DEVICE SERIAL NUMBER

Returns the serial number of the connected switch matrix.

Transmit Array

Byte	Data	Description
0	41	Interrupt code for Get Device Serial Number
1- 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	41	Interrupt code for Get Device Serial Number
1 to (n-1)	Serial Number	Series of bytes containing the ASCII code for each character in the serial number
n	0	Zero value byte to indicate the end of the serial number
(n+1) to 63	Not significant	"Don't care" bytes, can be any value

Example

The following example indicates that the connected switch box has serial number 1130922011 (see Appendix A for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	41	Interrupt code for Get Device Serial Number
1	49	ASCII character code for 1
2	49	ASCII character code for 1
3	51	ASCII character code for 3
4	48	ASCII character code for 0
5	57	ASCII character code for 9
6	50	ASCII character code for 2
7	50	ASCII character code for 2
8	48	ASCII character code for 0
9	49	ASCII character code for 1
10	49	ASCII character code for 1
11	0	Zero value byte to indicate end of string

See Also

Get Device Model Name

5.2.3. SET SINGLE SPDT / TRANSFER SWITCH

Sets an individual switch within an SPDT or transfer switch matrix whilst leaving any other switches unchanged. The switches are designated A to H, as labeled on the front of the switch matrix (not all switches are available in all models).

Applies To

Model	Serial Numbers
RC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers
ZTRC-xSPDT-x	All serial numbers

Transmit Array

Byte	Data	Description
0	1-8	Interrupt code for Set Single SPDT / Transfer Switch: 1 = Switch A 2 = Switch B (model dependent) ... 8 = Switch H (model dependent)
1	Switch State	Integer value for the switch position to set: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch)
2-63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	1-8	Interrupt code for Set Single SPDT / Transfer Switch: 1 = Switch A 2 = Switch B (model dependent) ... 8 = Switch H (model dependent)
1-63	Not significant	"Don't care" bytes, can be any value

Example

The following transmit array will set switch C to position 1 (Com connected to port 2):

Byte	Data	Description
0	3	Set Switch C
1	1	Set switch to state 1 (Com port connected to port 2)

The following transmit array will set switch C to position 0 (Com connected to port 1):

Byte	Data	Description
0	3	Set Switch C
1	0	Set switch to state 0 (Com port connected to port 1)

See Also

Get All SPDT / Transfer Switch States

5.2.4. SET ALL SPDT / TRANSFER SWITCHES

Sets the states of all switches simultaneously within an SPDT or transfer switch box.

Applies To

Model	Serial Numbers
RC-xSPDT-x	All serial numbers
RC-xMTS-x	All serial numbers
ZTRC-xSPDT-x	All serial numbers

Transmit Array

Byte	Data	Description
0	9	Interrupt code for Set All SPDT / Transfer Switches
1	Switch State	Numeric value indicating the required switch states. Each bit in BYTE1 represents the state of an individual SPDT or transfer switch with value: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch) The least significant bit (LSB) represents switch A and the most significant bit (MSB) represents switch H (if applicable).
2-63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	9	Interrupt code for Set All SPDT / Transfer Switches
1-63	Not significant	"Don't care" bytes, could be any value

Example

USB-8SPDT-A18 and RC-8SPDT-A18 each have 8 SPDT switches available (named A to H).
To set switches A, B and H to state 1 (Com connected to port 2) and all other switches to state 0 (Com port connected to port 1), BYTE1 would be represented as:

Bit	**Bit 7 (MSB)**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0 (LSB)**
Switch	**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**
Description	State	State	State	State	State	State	State	State
Value	1	0	0	0	0	0	1	1

BYTE1 = 10000011
= decimal 131

The complete transmit array (excluding "don't care" bytes) would be:

Byte	Data	Description
0	9	Interrupt code for Set All SPDT / Transfer Switches
1	131	Switch state to set

See Also

Get All SP4T Switch States

5.2.5. SET ALL SP4T SWITCHES

Sets the state of all switches simultaneously within an SP4T switch box. The common port (Com) of each switch can be connected to any one of the input/output ports (ports 1 to 4) or disconnected from all ports.

Transmit Array

Byte	Data	Description
0	9	Interrupt code for Set All SP4T Switches
1	Switch State	Numeric value indicating the required switch state. Each bit in BYTE1 corresponds to an input/output port, with the 4 least significant bits (LSB) for switch A (all models) and the 4 most significant bits (MSB) for switch B (where applicable). Setting any of these bits to 1 designates that this port should be connected to the Com port. Setting all bits to 0 disconnects all ports. Switch A (all models): If Bits 0 to 3 = 0, disconnect all ports for switch A If Bit 0 = 1, connect switch A Com to port 1 If Bit 1 = 1, connect switch A Com to port 2 If Bit 2 = 1, connect switch A Com to port 3 If Bit 3 = 1, connect switch A Com to port 4 Switch B (USB-2SP4T-A18/RC-2SP4T-A18 only): If Bits 4 to 7 = 0, disconnect all ports for switch B If Bit 4 = 1, connect switch B Com to port 1 If Bit 5 = 1, connect switch B Com to port 2 If Bit 6 = 1, connect switch B Com to port 3 If Bit 7 = 1, connect switch B Com to port 4
2-63	Not significant	

Returned Array

Byte	Data	Description
0	9	Interrupt code for Set All SP4T Switches
1	Status	2 = Command executed successfully 4 = Switch not set (invalid switch state requested)
2-63	Not significant	

Example

To disconnect all ports in switch A and set switch B to position 1 (connecting Com port to port 1), byte 1 is formed as below:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	**Switch B**				**Switch A**			
Description	Port 4	Port 3	Port 2	Port 1	Port 4	Port 3	Port 2	Port 1
Value	0	0	0	1	0	0	0	0

BYTE1 = 00010000
= decimal 16

The complete transmit array would be:

Byte	Data	Description
0	9	Interrupt code for Set All SP4T Switches
1	16	Disconnect switch A (all ports) and set switch B to position 1

See Also

Get All SP4T Switch States

5.2.6. SET SP6T SWITCH

Sets an individual switch within an SP6T switch matrix. Any other switches present in the matrix will be left unchanged.

Transmit Array

Byte	Data	Description
0	12	Interrupt code for Set SP6T Switch
1	Switch Number	The switch to set: 1 = Switch A 2 = Switch B
2	Switch State	The switch state to set, 0 to 6
3 - 63	Not significant	

Returned Array

Byte	Data	Description
0	12	Interrupt code for Set SP6T Switch
1 - 63	Not significant	

Example

The following transmit array will set switch A to position 5 (Com connected to port 5):

Byte	Data	Description
0	12	Set SP6T Switch
1	1	Set switch A
2	5	Set switch to state 5 (Com port connected to port 5)

See Also

Get SP6T Switch State

5.2.7. GET ALL SPDT / TRANSFER SWITCH STATES

Returns the state of all switches within an SPDT or transfer switch box.

Transmit Array

Byte	Data	Description
0	15	Interrupt code for Get All SPDT / Transfer Switch States
1-63	Not significant	

Returned Array

Byte	Data	Description
0	15	Interrupt code for Get All SPDT / Transfer Switch States
1	Switch State	Numeric value indicating the switch states. The value should be interpreted as a byte, with each bit representing the state of an individual SPDT switch as below: 0 = Connect Com port to port 1 (SPDT) Connect J1 <> J3 and J2 <> J4 (transfer switch) 1 = Connect Com port to port 2 Connect J1 <> J2 and J3 <> J4 (transfer switch) The least significant bit (LSB) represents switch A and the most significant bit (MSB) represents switch H (if applicable).
2-63	Not significant	

Example

Byte	Data	Description
0	15	Interrupt code for Get All SPDT Switch States
1	13	Switch states

RC-4SPDT-A18 has four SPDT switches, named A to D. The above returned array has BYTE1 = 13 which can be represented as:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	---	---	---	---	**D**	**C**	**B**	**A**
Description	---	---	---	---	State	State	State	State
Value	0	0	0	0	1	1	0	1

Bits 0, 2 and 3 have value 1, therefore switches A, C and D are set to position 1 (Com port connected to port2). Bit 1 has value 0, therefore switch B is set to position 0 (Com port connected to port 1).

See Also

Set All SPDT / Transfer Switches

5.2.8. GET ALL SP4T SWITCH STATES

Returns the state of all switches within an SP4T switch box. The common port (Com) of each switch can be in positions 1 to 4, indicating that the common port (Com) is connected to the respective input/output port, or in position 0 to indicate that the Com port is disconnected from all input/output ports.

Transmit Array

Byte	Data	Description
0	15	Interrupt code for Get All SP4T Switch States
1-63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	15	Interrupt code for Get All SP4T Switch States
1	Switch State	Numeric value indicating the switch states. Each bit in BYTE1 corresponds to an input/output port, with the 4 least significant bits (LSB) for switch A (all models) and the 4 most significant bits (MSB) for switch B (model dependent). Switch A (all models): If Bits 0 to 3 = 0, switch A has all ports disconnected If Bit 0 = 1, switch A has Com connected to port 1 If Bit 1 = 1, switch A has Com connected to port 2 If Bit 2 = 1, switch A has Com connected to port 3 If Bit 3 = 1, switch A has Com connected to port 4 Switch B (USB-2SP4T-A18/RC-2SP4T-A18 only): If Bits 4 to 7 = 0, switch B has all ports disconnected If Bit 4 = 1, switch B has Com connected to port 1 If Bit 5 = 1, switch B has Com connected to port 2 If Bit 6 = 1, switch B has Com connected to port 3 If Bit 7 = 1, switch B has Com connected to port 4
2 - 63	Not significant	

Example

Byte	Data	Description
0	15	Interrupt code for Get All SP4T Switch States
1	68	Switch state

RC-4SPDT-A18 has two SP4T switches, named A and B. The above returned array has BYTE1 = 68 which can be represented as:

Bit	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Switch	**Switch B**				**Switch A**			
Description	Port 4	Port 3	Port 2	Port 1	Port 4	Port 3	Port 2	Port 1
Value	0	1	0	0	0	1	0	0

Bit 2 has value 1, indicating that switch A is in position 3 (Com port connected to port 3). Bit 6 has value 1, indicating that switch B is also in position 3 (Com port connected to port 3).

See Also

Set All SP4T Switches

5.2.9. GET SP6T SWITCH STATE

Returns the state of a specific switch within an SP6T switch box.

Transmit Array

Byte	Data	Description
0	13	Interrupt code for Get SP6T Switch State
1	Switch Number	The switch to query, 1 (switch A) or 2 (switch B, model dependent)
2 - 63	Not significant	

Returned Array

Byte	Data	Description
0	13	Interrupt code for Get SP6T Switch State
1	Switch State	Numeric value indicating the switch state, from 0 to 6
2 - 63	Not significant	

Example

Send the following transmit array to query the state of switch B:

Byte	Data	Description
0	13	Interrupt code for Get SP6T Switch State
1	2	Switch B

The below returned array would indicate switch B has com port connected to port

Byte	Data	Description
0	13	Interrupt code for Get SP6T Switch State
1	6	Com connected to port 6

See Also

Set SP6T Switch

5.2.10. SET POWER-UP MODE

Sets the switch state to be loaded when DC power is first applied to the switch box, either the last remembered state or the default state.

To ensure proper operation of the "last remembered state" option, the Save Switch Counters & States command should be sent prior to powering off the switch box.

Applies To

Model	Firmware
All models	D2 or later

Transmit Array

Byte	Data	Description
0	89	Interrupt code for Set Power-Up Mode
1	Mode	The power-up mode: 0 = All switches will power-up in the last remembered state 1 = All switches will power-up in the default state: SP4T & SP6T: All ports disconnected SPDT: Com connected to port 1 Transfer: J1 <> J3 and J2 <> J4
2 - 63	Not significant	

Returned Array

Byte	Data	Description
0	89	Interrupt code for Set Power-Up Mode
1 - 63	Not significant	

Example

The following transmit array will set the switches to power-up in the last remembered state:

Byte	Data	Description
0	89	Set Power-Up Mode
1	1	Power-up in the last remembered state

See Also

Get Power-Up Mode
Save Switch Counters & States

5.2.11. GET POWER-UP MODE

Returns the power-up mode for the switch box, indicating whether the switches will initially assume the last remembered state or the default state.

Applies To

Model	Firmware
All models	D2 or later

Transmit Array

Byte	Data	Description
0	90	Interrupt code for Get Power-Up Mode
1 - 63	Not significant	

Returned Array

Byte	Data	Description
0	89	Interrupt code for Get Power-Up Mode
1	Mode	The power-up mode: 0 = All switches will power-up in the last remembered state 1 = All switches will power-up in the default state: SP4T & SP6T: All ports disconnected SPDT: Com connected to port 1 Transfer: J1 <> J3 and J2 <> J4
2 - 63	Not significant	

Example

The following returned array indicates that the switches will power-up in the last remembered state:

Byte	Data	Description
0	90	Get Power-Up Mode
1	1	Power-up in the last remembered state

See Also

Set Power-Up Mode
Save Switch Counters & States

5.2.12. GET FIRMWARE

Returns the internal firmware version of the switch box.

Transmit Array

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1- 63	Not significant	

Returned Array

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1	Reserved	Internal code for factory use only
2	Reserved	Internal code for factory use only
3	Reserved	Internal code for factory use only
4	Reserved	Internal code for factory use only
5	Firmware Letter	ASCII code for the first character in the firmware revision identifier
6	Firmware Number	ASCII code for the second character in the firmware revision identifier
7-63	Not significant	

Example

The following returned array indicates that the switch box has firmware version C3:

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1	55	Internal code for factory use only
2	52	Internal code for factory use only
3	83	Internal code for factory use only
4	87	Internal code for factory use only
5	67	ASCII code for the letter "C"
6	51	ASCII code for the number 3
7-63	Not significant	

5.2.13. GET INTERNAL TEMPERATURE

Returns the internal temperature measured at 1 of the internal sensors (model dependent):

- RC-1SPDT-x (1 sensor)
- RC-2SPDT-x (2 sensors)
- RC-3SPDT-x (2 sensors)
- RC-4SPDT-x (2 sensors)
- RC-8SPDT-x (3 sensors)
- RC-1SP4T-x (1 sensor)
- RC-2SP4T-x (2 sensors)
- RC-1SP6T-x (1 sensor)
- RC-2SP6T-x (2 sensors)
- RC-2MTS-x (2 sensors)
- RC-3MTS-x (2 sensors)
- ZTRC-4SPDT-x (2 sensors)
- ZTRC-8SPDT-x (3 sensors)

Note: "+25.00" will be returned when polling a sensor which is not fitted

Transmit Array

Byte	Data	Description
0	114, 115 or 118	Interrupt code for Get Internal Temperature: 114 = Check temperature sensor 1 115 = Check temperature sensor 2 (if available) 118 = Check temperature sensor 3 (if available)
1-63	Not significant	

Returned Array

Byte	Data	Description
0	114, 115 or 118	Interrupt code for Get Internal Temperature: 114 = Check temperature sensor 1 115 = Check temperature sensor 2 (if available) 118 = Check temperature sensor 3 (if available)
1	43 or 45	ASCII code for the first character of the temperature: 43 = positive (+) 45 = negative (-)
2	Temperature Digit 1	ASCII character code for the first digit of the temperature reading
3	Temperature Digit 2	ASCII character code for the second digit of the temperature reading
4	46	ASCII character code for the decimal point symbol (".")
5	Temperature Decimal Place 1	ASCII character code for the first decimal place of the temperature reading
6	Temperature Decimal Place 2	ASCII character code for the second decimal place of the temperature reading
7-63	Not significant	

Example

To check the internal temperature measured by sensor 2, send the following transmit array.

Byte	Data	Description
0	115	Interrupt code for Get Internal Temperature at sensor 2

The below returned array would indicate a temperature of +28.43°C.

Byte	Data	Description
0	115	Interrupt code for Get Internal Temperature at sensor 1
1	43	ASCII code for the character "+" (positive)
2	50	ASCII code for the character "2"
3	56	ASCII code for the character "8"
4	45	ASCII code for the character "."
5	52	ASCII code for the character "4"
6	51	ASCII code for the character "3"

See Also

Get Heat Alarm

Get Fan Status

5.2.14. GET HEAT ALARM

Returns an alarm notification if any of the internal temperature sensors exceeds the factory programmed limits (45°C on the PCB or 48°C on the internal switch case).

Transmit Array

Byte	Data	Description
0	117	Interrupt code for Get Heat Alarm
1-63	Not required	

Returned Array

Byte	Data	Description
0	117	Interrupt code for Get Heat Alarm
1	Alarm Status	The heat alarm status: 0 = All temperature sensors are within allowed limits 1 = Temperature exceeds specified limits
2-63	Not required	

Example

The following return array would indicate the unit is within normal temperature limits:

Byte	Data	Description
0	117	Interrupt code for Get Heat Alarm
1	0	All temperature sensors are within allowed limits

See Also

Get Internal Temperature
Get Fan Status

5.2.15. GET FAN STATUS

Indicates whether the internal fan is currently operating.

Applies To

All models except ZTRC-4SPDT-A18

Transmit Array

Byte	Data	Description
0	119	Interrupt code for Get Fan Status
1-63	Not required	

Returned Array

Byte	Data	Description
0	119	Interrupt code for Get Fan Status
1	Fan Status	The fan status: 0 = fan not currently operating 1 = fan operating
2-63	Not required	

Example

The following return array would indicate the fan is not currently operating:

Byte	Data	Description
0	119	Interrupt code for Get Fan Status
1	0	Fan not currently operating

See Also

Get Internal Temperature
Get Heat Alarm

5.2.16. GET SPDT / TRANSFER SWITCH COUNTER

Returns the number of switching cycles undertaken by an individual switch (specified by the user) within an SPDT or transfer switch box.

Transmit Array

Byte	Data	Description
0	17	Interrupt code for Get Switch Counter
1	Switch Name	ASCII character code for the switch name. For example, switch A = 65, switch B = 66...
2-63	Not required	

Returned Array

Byte	Data	Description
0	17	Interrupt code for Get Switch Counter
1-4	Counter Value	The switch counter value split across 4 bytes. The count is calculated as: BYTE1 + (BYTE2*256) + (BYTE3*256^2) + (BYTE4*256^3)
5-63	Not significant	

Example

To query the counter of switch C, send:

Byte	Data	Description
0	17	Interrupt code for Get Switch Counter
1	67	Query switch C counter (ASCII character code for C = 67)

Calculate the counter value from the below example return array:

Byte	Data	Description
0	17	Interrupt code for Get Switch Counter
1	97	The switch counter value split across 4 bytes, calculated as: COUNT = 97 + (132*256) + (11*256^2) + (0*256^3) = 754,785 cycles
2	132	
3	11	
4	0	

See Also

Get All Switch Counters
Save Switch Counters

5.2.17. GET ALL SWITCH COUNTERS

Returns the number of switching cycles undertaken by each individual switch within an SPDT or transfer switch box. For SP4T or SP6T switches, the return indicates the number of times that the Com port has connected to each of the input/output ports.

Transmit Array

Byte	Data	Description
0	18	Interrupt code for Get All Switch Counters
1-63	Not required	

Returned Array (SPDT Switch Boxes)

Byte	Data	Description
0	18	Interrupt code for Get All Switch Counters
1-4	Switch A Counter Value	Switch A counter value split across 4 bytes. The count is calculated as: BYTE1 + (BYTE2*256) + (BYTE3*256^2) + (BYTE4*256^3)
5-8	Switch B Counter Value	Switch B (if applicable) counter value split across 4 bytes: BYTE5 + (BYTE6*256) + (BYTE7*256^2) + (BYTE8*256^3)
9-12	Switch C Counter Value	Switch C (if applicable) counter value split across 4 bytes: BYTE9 + (BYTE10*256) + (BYTE11*256^2) + (BYTE12*256^3)
13-16	Switch D Counter Value	Switch D (if applicable) counter value split across 4 bytes: BYTE13 + (BYTE14*256) + (BYTE15*256^2) + (BYTE16*256^3)
17-20	Switch E Counter Value	Switch E (if applicable) counter value split across 4 bytes: BYTE17 + (BYTE18*256) + (BYTE19*256^2) + (BYTE20*256^3)
21-24	Switch F Counter Value	Switch F (if applicable) counter value split across 4 bytes: BYTE21 + (BYTE22*256) + (BYTE23*256^2) + (BYTE24*256^3)
25-28	Switch G Counter Value	Switch G (if applicable) counter value split across 4 bytes: BYTE25 + (BYTE26*256) + (BYTE27*256^2) + (BYTE28*256^3)
29-32	Switch H Counter Value	Switch H (if applicable) counter value split across 4 bytes: BYTE29 + (BYTE30*256) + (BYTE31*256^2) + (BYTE32*256^3)
33-63	Not significant	

Note: Bytes relating to switches that are not available in the connected hardware become "don't care" bytes and will contain arbitrary values. For example, RC-3SPDT-A18 has 3 SPDT switches, named A to C. Therefore, only bytes 0 to 12 are relevant in the returned array.

Returned Array (SP4T Switch Boxes)

Byte	Data	Description
0	18	Interrupt code for Get All Switch Counters
1-4	Switch A, Port 1 Counter Value	Counter of Switch A Com to port 1 connections, split across 4 bytes: BYTE1 + (BYTE2*256) + (BYTE3*256^2) + (BYTE4*256^3)
5-8	Switch A, Port 2 Counter Value	Counter of Switch A Com to port 2 connections, split across 4 bytes: BYTE5 + (BYTE6*256) + (BYTE7*256^2) + (BYTE8*256^3)
9-12	Switch A, Port 3 Counter Value	Counter of Switch A Com to port 3 connections, split across 4 bytes: BYTE9 + (BYTE10*256) + (BYTE11*256^2) + (BYTE12*256^3)
13-16	Switch A, Port 4 Counter Value	Counter of Switch A Com to port 4 connections, split across 4 bytes: BYTE13 + (BYTE14*256) + (BYTE15*256^2) + (BYTE16*256^3)
17-20	Switch B, Port 1 Counter Value	Counter of Switch B Com to port 1 connections, split across 4 bytes: BYTE17 + (BYTE18*256) + (BYTE19*256^2) + (BYTE20*256^3)
21-24	Switch B, Port 2 Counter Value	Counter of Switch B Com to port 2 connections, split across 4 bytes: BYTE21 + (BYTE22*256) + (BYTE23*256^2) + (BYTE24*256^3)
25-28	Switch B, Port 3 Counter Value	Counter of Switch B Com to port 3 connections, split across 4 bytes: BYTE25 + (BYTE26*256) + (BYTE27*256^2) + (BYTE28*256^3)
29-32	Switch B, Port 4 Counter Value	Counter of Switch B Com to port 4 connections, split across 4 bytes: BYTE29 + (BYTE30*256) + (BYTE31*256^2) + (BYTE32*256^3)
33-63	Not significant	

Example (SPDT Switch Boxes)

The following returned array shows the counter values for RC-3SPDT-A18 (three SPDT switch box):

Byte	Data	Description
0	18	Interrupt code for Get All Switch Counters
1	97	Switch A counter value split across 4 bytes, calculated as: 97 + (132*256) + (11*256^2) + (0 * 256^3) = 754,785 cycles
2	132	
3	11	
4	0	
5	5	Switch B counter value split across 4 bytes, calculated as: 5 + (15*256) + (5*2562) + (0 * 2563) = 331,525 cycles
6	15	
7	5	
8	0	
9	111	Switch C counter value split across 4 bytes, calculated as: 111 + (97*256) + (2*2562) + (1 * 2563) = 16,933,231 cycles
10	97	
11	2	
12	1	

See Also

Save Switch Counters

5.2.18. SAVE SWITCH COUNTERS & STATES

Transfers the latest switch counters and switch states from temporary to permanent memory. During normal operation, this data is internally stored in volatile memory but automatically updated into permanent memory every 3 minutes. This command should be sent following completion of all switching routines and prior to powering off the switch in order to ensure that the data is permanently saved.

Applies To

Firmware version D1 or later

Transmit Array

Byte	Data	Description
0	88	Interrupt code for Save Switch Counters
1-63	Not significant	

Returned Array

Byte	Data	Description
0	88	Interrupt code for Save Switch Counters
1-63	Not significant	

See Also

Get SPDT / Transfer Switch Counter
Get All Switch Counters

5.2.19. SEND SCPI COMMAND

This function sends a SCPI command to the switch box and collects the returned acknowledgement. SCPI (Standard Commands for Programmable Instruments) is a common method for communicating with and controlling instrumentation products.

Applies To

Firmware version E3 or later

Transmit Array

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1 - 63	SCPI String	The SCPI command as a series of ASCII character codes, one character code per byte

Returned Array

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1 to (n-1)	SCPI String	SCPI return string, one character per byte, represented as ASCII character codes
n	0	Zero value byte to indicate the end of the SCPI return string
(n+1) to 63	Not significant	

Example

The SCPI command to request the model name is `:MN?` (see Get Model Name). The ASCII character codes representing the 4 characters in this command should be sent in bytes 1 to 4 of the transmit array as follows:

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1	49	ASCII character code for :
2	77	ASCII character code for M
3	78	ASCII character code for N
4	63	ASCII character code for ?

The returned array for RUDAT-6000-30 would be as follows:

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1	82	ASCII character code for R
2	67	ASCII character code for C
3	45	ASCII character code for -
4	50	ASCII character code for 2
5	53	ASCII character code for S
6	50	ASCII character code for P
7	52	ASCII character code for 4
8	54	ASCII character code for T
9	45	ASCII character code for -
10	65	ASCII character code for A
11	49	ASCII character code for 1
12	56	ASCII character code for 8
13	0	Zero value byte to indicate end of string

See Also

SCPI - Summary of Commands / Queries

5.3. Interrupts - Ethernet Configuration Commands / Queries

These commands and queries apply to Mini-Circuits' RC series of mechanical switch boxes for configuring the Ethernet parameters.

	Description	Command Code	
		Byte 0	Byte 1
a	Set Static IP Address	250	201
b	Set Static Subnet Mask	250	202
c	Set Static Network Gateway	250	203
d	Set HTTP Port	250	204
e	Set Telnet Port	250	214
f	Use Password	250	205
g	Set Password	250	206
h	Use DHCP	250	207
i	Get Static IP Address	251	201
j	Get Static Subnet Mask	251	202
k	Get Static Network Gateway	251	203
l	Get HTTP Port	251	204
m	Get Telnet Port	251	214
n	Get Password Status	251	205
o	Get Password	251	206
p	Get DHCP Status	251	207
q	Get Dynamic Ethernet Configuration	253	
r	Get MAC Address	252	
s	Reset Ethernet Configuration	101	101

5.3.1. SET STATIC IP ADDRESS

Description

Sets the static IP address to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	201	Interrupt code for Set IP Address
2	IP_Byte0	First byte of IP address
3	IP_Byte1	Second byte of IP address
4	IP_Byte2	Third byte of IP address
5	IP_Byte3	Fourth byte of IP address
6 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To set the static IP address to 192.168.100.100, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	201	Interrupt code for Set IP Address
2	192	First byte of IP address
3	168	Second byte of IP address
4	100	Third byte of IP address
5	100	Fourth byte of IP address

See Also

Use DHCP
Get Static IP Address
Reset Ethernet Configuration

5.3.2. SET STATIC SUBNET MASK

Description

Sets the static subnet mask to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	202	Interrupt code for Set Subnet Mask
2	IP_Byte0	First byte of subnet mask
3	IP_Byte1	Second byte of subnet mask
4	IP_Byte2	Third byte of subnet mask
5	IP_Byte3	Fourth byte of subnet mask
6 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To set the static subnet mask to 255.255.255.0, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	202	Interrupt code for Set Subnet Mask
2	255	First byte of subnet mask
3	255	Second byte of subnet mask
4	255	Third byte of subnet mask
5	0	Fourth byte of subnet mask

See Also

Use DHCP
Get Static Subnet Mask
Reset Ethernet Configuration

5.3.3. SET STATIC NETWORK GATEWAY

Description

Sets the network gateway IP address to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	203	Interrupt code for Set Network Gateway
2	IP_Byte0	First byte of network gateway IP address
3	IP_Byte1	Second byte of network gateway IP address
4	IP_Byte2	Third byte of network gateway IP address
5	IP_Byte3	Fourth byte of network gateway IP address
6 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To set the static IP address to 192.168.100.0, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	203	Interrupt code for Set Network Gateway
2	192	First byte of IP address
3	168	Second byte of IP address
4	100	Third byte of IP address
5	0	Fourth byte of IP address

See Also

Use DHCP
Get Static Network Gateway
Reset Ethernet Configuration

5.3.4. SET HTTP PORT

Description

Sets the port to be used for HTTP communication (default is port 80).

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	204	Interrupt code for Set HTTP Port
2	Port_Byte0	First byte (MSB) of HTTP port value: Port_Byte0 = INTEGER (Port / 256)
3	Port_Byte1	Second byte (LSB) of HTTP port value: Port_byte1 = Port - (Port_Byte0 * 256)
4 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To set the HTTP port to 8080, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	204	Interrupt code for Set HTTP Port
2	31	Port_Byte0 = INTEGER (8080 / 256)
3	144	Port_byte1 = 8080 - (31 * 256)

See Also

Set Telnet Port
Get HTTP Port
Get Telnet Port
Reset Ethernet Configuration

5.3.5. SET TELNET PORT

Description

Sets the port to be used for Telnet communication (default is port 23).

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	214	Interrupt code for Set Telnet Port
2	Port_Byte0	First byte (MSB) of Telnet port value: Port_Byte0 = INTEGER (Port / 256)
3	Port_Byte1	Second byte (LSB) of Telnet port value: Port_byte1 = Port - (Port_Byte0 * 256)
4 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To set the Telnet port to 22, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	214	Interrupt code for Set Telnet Port
2	0	Port_Byte0 = INTEGER (22 / 256)
3	22	Port_byte1 = 22 - (0 * 256)

See Also

Set HTTP Port
Get HTTP Port
Get Telnet Port
Reset Ethernet Configuration

5.3.6. USE PASSWORD

Description

Enables or disables the requirement to password protect the HTTP / Telnet communication.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	205	Interrupt code for Use Password
2	PW_Mode	0 = password not required (default) 1 = password required
3 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To enable the password requirement for Ethernet communication, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	205	Interrupt code for Use Password
2	1	Enable password requirement

See Also

Set Password
Get Password Status
Get Password
Reset Ethernet Configuration

5.3.7. SET PASSWORD

Description

Sets the password to be used for Ethernet communicatoin (when password security is enabled, maximum 20 characters.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	206	Interrupt code for Set Password
2	PW_Length	Length (number of characters) of the password
3 to n	PW_Char	Series of ASCII character codes (1 per byte) for the Ethernet password
n + 1 to 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 to 63	Not significant	Any value

Example

To set the password to Pass_123, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	206	Interrupt code for Set Password
2	8	Length of password (8 characters)
3	80	ASCII character code for P
4	97	ASCII character code for a
5	115	ASCII character code for s
6	115	ASCII character code for s
7	95	ASCII character code for _
8	49	ASCII character code for 1
9	50	ASCII character code for 2
10	51	ASCII character code for 3

See Also

Use Password
Get Password Status
Get Password
Reset Ethernet Configuration

5.3.8. USE DHCP

Description

Enables or disables DHCP (dynamic host control protocol). With DHCP enabled, the attenuators Ethernet / IP configuration is assigned by the network and any user defined static IP settings are ignored. With DHCP disabled, the user defined static IP settings are used.

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	205	Interrupt code for Use DHCP
2	DHCP_Mode	0 = DCHP disabled (static IP settings in use) 1 = DHCP enabled (default - dynamic IP in use)
3 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1 - 63	Not significant	Any value

Example

To enable DHCP for Ethernet communication, the transmit array is:

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	205	Interrupt code for Use DHCP
2	1	Enable DHCP

See Also

Use DHCP
Get DHCP Status
Get Dynamic Ethernet Configuration
Reset Ethernet Configuration

5.3.9. GET STATIC IP ADDRESS

Description

Gets the static IP address (configured by the user) to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	201	Interrupt code for Get IP Address
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	IP_Byte0	First byte of IP address
2	IP_Byte1	Second byte of IP address
3	IP_Byte2	Third byte of IP address
4	IP_Byte3	Fourth byte of IP address
5 - 63	Not significant	Any value

Example

The following returned array would indicate that a static IP address of 192.168.100.100 has been configured:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	192	First byte of IP address
2	168	Second byte of IP address
3	100	Third byte of IP address
4	100	Fourth byte of IP address

See Also

Use DHCP
Set Static IP Address

5.3.10. GET STATIC SUBNET MASK

Description

Gets the subnet mask (configured by the user) to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	202	Interrupt code for Get Subnet Mask
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	IP_Byte0	First byte of subnet mask
2	IP_Byte1	Second byte of subnet mask
3	IP_Byte2	Third byte of subnet mask
4	IP_Byte3	Fourth byte of subnet mask
5 - 63	Not significant	Any value

Example

The following returned array would indicate that a subnet mask of 255.255.255.0 has been configured:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	255	First byte of subnet mask
2	255	Second byte of subnet mask
3	255	Third byte of subnet mask
4	0	Fourth byte of subnet mask

See Also

Use DHCP
Set Static Subnet Mask

5.3.11. GET STATIC NETWORK GATEWAY

Description

Gets the static IP address (configured by the user) of the network gateway to be used when DHCP (dynamic host control protocol) is disabled.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	203	Interrupt code for Get Network Gateway
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	IP_Byte0	First byte of IP address
2	IP_Byte1	Second byte of IP address
3	IP_Byte2	Third byte of IP address
4	IP_Byte3	Fourth byte of IP address
5 - 63	Not significant	Any value

Example

The following returned array would indicate that a network gateway IP address of 192.168.100.0 has been configured:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	192	First byte of IP address
2	168	Second byte of IP address
3	100	Third byte of IP address
4	0	Fourth byte of IP address

See Also

Use DHCP
Set Static Network Gateway

5.3.12. GET HTTP PORT

Description

Gets the port to be used for HTTP communication (default is port 80).

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	204	Interrupt code for Get HTTP Port
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	Port_Byte0	First byte (MSB) of HTTP port value:
2	Port_Byte1	Second byte (LSB) of HTTP port value: Port = (Port_Byte0 * 256) + Port_Byte1
3 - 63	Not significant	Any value

Example

The following returned array would indicate that the HTTP port has been configured as 8080:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	31	
2	144	Port = (31 * 256) + 144 = 8080

See Also

Set HTTP Port
Set Telnet Port
Get Telnet Port

5.3.13. GET TELNET PORT

Description

Gets the port to be used for Telnet communication (default is port 23).

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	214	Interrupt code for Get Telnet Port
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	Port_Byte0	First byte (MSB) of Telnet port value:
2	Port_Byte1	Second byte (LSB) of Telnet port value: Port = (Port_Byte0 * 256) + Port_Byte1
3 - 63	Not significant	Any value

Example

The following returned array would indicate that the Telnet port has been configured as 22:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	0	
2	22	Port = (0 * 256) + 22 = 22

See Also

Set HTTP Port
Set Telnet Port
Get HTTP Port

5.3.14. GET PASSWORD STATUS

Description

Checks whether the attenuators has been configured to require a password for HTTP / Telnet communication.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	205	Interrupt code for Get Password Status
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Set Ethernet Configuration
1	PW_Mode	0 = password not required (default) 1 = password required
2 - 63	Not significant	Any value

Example

The following returned array indicates that password protection is enabled:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	1	Password protection enabled

See Also

Use Password
Set Password
Get Password

5.3.15. GET PASSWORD

Description

Gets the password to be used for Ethernet communicatoin (when password security is enabled, maximum 20 characters.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	206	Interrupt code for Get Password
2 to 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	PW_Length	Length (number of characters) of the password
2 to n	PW_Char	Series of ASCII character codes (1 per byte) for the Ethernet password
n to 63	Not significant	Any value

Example

The following returned array indicated that the password has been set to Pass_123:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	8	Length of password (8 characters)
2	80	ASCII character code for P
3	97	ASCII character code for a
4	115	ASCII character code for s
5	115	ASCII character code for s
6	95	ASCII character code for _
7	49	ASCII character code for 1
8	50	ASCII character code for 2
9	51	ASCII character code for 3

See Also

Use Password
Set Password
Get Password Status

5.3.16. GET DHCP STATUS

Description

Checks whether DHCP (dynamic host control protocol) is enabled or disabled. With DHCP enabled, the attenuators Ethernet / IP configuration is assigned by the network and any user defined static IP settings are ignored. With DHCP disabled, the user defined static IP settings are used.

Transmit Array

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	207	Interrupt code for Get DHCP Status
2 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	251	Interrupt code for Set Ethernet Configuration
1	DCHP_Mode	0 = DCHP disabled (static IP settings in use) 1 = DHCP enabled (default - dynamic IP in use)
2 - 63	Not significant	Any value

Example

The following returned array indicates that DHCP is enabled:

Byte	Data	Description
0	251	Interrupt code for Get Ethernet Configuration
1	1	DHCP enabled

See Also

Use DHCP
Get Dynamic Ethernet Configuration

5.3.17. GET DYNAMIC ETHERNET CONFIGURATION

Description

Returns the IP address, subnet mask and default gateway currently used by the programmable attenuator. If DHCP is enabled then these values are assigned by the network DHCP server. If DHCP is disabled then these values are the static configuration defined by the user.

Transmit Array

Byte	Data	Description
0	253	Interrupt code for Get Dynamic Ethernet Configuration
1 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	253	Interrupt code for Get Dynamic Ethernet Configuration
1	IP_Byte0	First byte of IP address
2	IP_Byte1	Second byte of IP address
3	IP_Byte2	Third byte of IP address
4	IP_Byte3	Fourth byte of IP address
5	SM_Byte0	First byte of subnet mask
6	SM_Byte1	Second byte of subnet mask
7	SM_Byte2	Third byte of subnet mask
8	SM_Byte3	Fourth byte of subnet mask
9	NG_Byte0	First byte of network gateway IP address
10	NG_Byte1	Second byte of network gateway IP address
11	NG_Byte2	Third byte of network gateway IP address
12	NG_Byte3	Fourth byte of network gateway IP address
13 - 63	Not significant	Any value

Example

The following returned array would indicate the below Ethernet configuration is active:

- IP Address: 192.168.100.100
- Subnet Mask: 255.255.255.0
- Network Gateway: 192.168.100.0

Byte	Data	Description
0	253	Interrupt code for Get Dynamic Ethernet Configuration
1	192	First byte of IP address
2	168	Second byte of IP address
3	100	Third byte of IP address
4	100	Fourth byte of IP address
5	255	First byte of subnet mask
6	255	Second byte of subnet mask
7	255	Third byte of subnet mask
8	0	Fourth byte of subnet mask
9	192	First byte of network gateway IP address
10	168	Second byte of network gateway IP address
11	100	Third byte of network gateway IP address
12	0	Fourth byte of network gateway IP address

See Also

Use DHCP
Get DHCP Status

5.3.18. GET MAC ADDRESS

Description

Returns the MAC address of the programmable attenuator.

Transmit Array

Byte	Data	Description
0	252	Interrupt code for Get MAC Address
1 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	252	Interrupt code for Get MAC Address
1	MAC_Byte0	First byte of MAC address
2	MAC_Byte1	Second byte of MAC address
3	MAC_Byte2	Third byte of MAC address
4	MAC_Byte3	Fourth byte of MAC address
5	MAC_Byte4	Fifth byte of MAC address
6	MAC_Byte5	Sixth byte of MAC address
7 - 63	Not significant	Any value

Example

The following returned array would indicate a MAC address (in decimal notation) of 11:47:165:103:137:171:

Byte	Data	Description
0	252	Interrupt code for Get MAC Address
1	11	First byte of MAC address
2	47	Second byte of MAC address
3	165	Third byte of MAC address
4	103	Fourth byte of MAC address
5	137	Fifth byte of MAC address
6	171	Sixth byte of MAC address

See Also

Get Dynamic Ethernet Configuration

5.3.19. RESET ETHERNET CONFIGURATION

Description

Forces the programmable attenuator to reset and adopt the latest Ethernet configuration. Must be sent after any changes are made to the configuration.

Transmit Array

Byte	Data	Description
0	101	Reset Ethernet configuration sequence
1	101	Reset Ethernet configuration sequence
2	102	Reset Ethernet configuration sequence
3	103	Reset Ethernet configuration sequence
4 - 63	Not significant	Any value

Returned Array

Byte	Data	Description
0	101	Confirmation of reset Ethernet configuration sequence
1 - 63	Not significant	Any value

6. Control Options for MacOS

Mini-Circuits is not able to provide formal software support (GUI & API) for MacOS users but it is possible to control Mini-Circuits' Ethernet enabled devices without any software installation, including from MacOS.

The key steps to get started would be as follows.

6.1. Connect & Identify Initial IP Address

For connection into a network supporting DHCP:

1. DHCP is enabled by default so an IP address should be assigned automatically when the device is connected to the network
2. Identify the assigned IP address by referring to the network administrator or router.
3. Alternatively, a broadcast query can be sent to the network using UDP so that all Mini-Circuits devices respond with their IP (refer to Device Discovery Using UDP)
4. Once identified, the dynamic IP can be used to connect and control the device, including to set a new static IP configuration if required

For a direct connection between the Mac and Mini-Circuits device:

1. For devices with the latest firmware, a default "link-local" IP of 169.254.10.10 will be set if no response is received from a DHCP server (which will be the case for a direct computer connection)
2. This IP can be used to connect to the device and update the Ethernet configuration as needed
3. Refer to Link-Local / Auto IP Address for details of supported models

6.2. Updating the Ethernet Configuration

Once the initial IP address has been identified using the above steps, the device can be connected in order to set a new static IP address configuration. This can be achieved by writing an automation program based on the ASCII / SCPI commands detailed in this manual.

Alternatively, for a one-time step as part of initial commissioning, it may be simpler to use Mini-Circuits' HTML tool which can be downloaded from:

https://www.minicircuits.com/softwaredownload/MCL_PTE_Ethernet_Config.zip

The tool is an HTML file which can be downloaded and opened on the computer, it provides a simple form which the user populates with the current IP address and the updated configuration to load. The HTML file connects and updates the Ethernet configuration as specified.

With a valid IP address, the full list of ASCII / SCPI commands summarized in this programming manual can be used to control the device.

The fallback in the event of an unknown or invalid IP configuration would be to connect the device by USB in order to overwrite the configuration.

7. Contact

Mini-Circuits

13 Neptune Avenue

Brooklyn, NY 11235

Phone: +1-718-934-4500

Email: sales@minicircuits.com

Web: www.minicircuits.com