Test Solutions - Programming Manual

ZTS Series
Solid-State Switch Systems

ZTS Series Solid-State Switch Systems

Mini-Circuits®
PO Box 350166, Brooklyn, NY 11235-0003
+1 718-934-4500 | testsolutions@minicircuits.com
www.minicircuits.com

Important Notice

Trademarks

Mini-Circuits
13 Neptune Avenue
Brooklyn, NY 11235, USA
Phone: +1-718-934-4500
Email: sales@minicircuits.com
Web: www.minicircuits.com

1 - Overview 6
1.1 - Control Methods 6
1.2 - Switch Address Structure 6
2 - SCPI Commands for Control of ZTS Series Switch Systems 6
2.1 - System Queries 7
2.1 (a) - Get Model Name 7
2.1 (b) - Get Serial Number 8
2.1 (c) - Count Number of Switch Modules 9
2.1 (d) - Get Firmware 10
2.2 - Switch Control 11
2.2 (a) - Set Switch State 12
2.2 (b) - Query Switch State 13
2.2 (c) - Set All Switches 14
2.2 (d) - Set Multiple Switches 15
2.3 - Manual Trigger Control 16
2.3 (a) - Reset Trigger In 17
2.3 (b) - Query Trigger In 18
2.3 (c) - Set Trigger Out 19
2.4 - Switch Sequence Configuration 20
2.4 (a) - Set Number of Steps 21
2.4 (b) - Query Number of Steps 22
2.4 (c) - Set Indexed Step 23
2.4 (d) - Query Indexed Step Number 24
2.4 (e) - Set Switch Command for Indexed Step 25
2.4 (f) - Query Switch Command for Indexed Step 26
2.4 (g) - Set Dwell Time for Indexed Step 27
2.4 (h) - Query Dwell Time for Indexed Step 28
2.4 (i) - Set Continuous Mode 29
2.4 (j) - Query Continuous Mode 30
2.4 (k) - Set Number of Cycles 31
2.4 (l) - Query Number of Cycles 32
2.4 (m) - Set Trigger-In Mode 33
2.4 (n) - Query Trigger-In Mode 34
2.4 (o) - Set Trigger-Out Mode 35
2.4 (p) - Query Trigger-Out Mode 36
2.4 (q) - Start / Stop Switch Sequence 37
2.5 - SCPI - Ethernet Configuration Commands 38
2.5 (a) - Set Static IP Address 39
2.5 (b) - Get Static IP Address 40
2.5 (c) - Set Static Subnet Mask 41
2.5 (d) - Get Static Subnet Mask 42
2.5 (e) - Set Static Network Gateway 43
2.5 (f) - Get Static Network Gateway 44
2.5 (g) - Set HTTP Port 45
2.5 (h) - Get HTTP Port 46
2.5 (i) - Set Telnet Port 47
2.5 (j) - Get Telnet Port 48
2.5 (k) - Set Password Requirement 49
2.5 (l) - Get Password Requirement 50
2.5 (m) - Set Password 51
2.5 (n) - Get Password 52

2.5 (o) - Set DHCP Status 53
2.5 (p) - Get DHCP Status 54
2.5 (q) - Get MAC Address 55
2.5 (r) - Get Current Ethernet Configuration 56
2.5 (s) - Update Ethernet Settings 57
3 - Ethernet Control API 58
3.1 - Configuring Ethernet Settings 58
3.2 - HTTP Communication 59
3.3 - Telnet Communication 60
3.4 - Device Discovery Using UDP 61
4 - USB Control API for Microsoft Windows 62
4.1 - DLL API Options 62
4.1 (a) - .NET Framework 2.0 DLL (Recommended) 62
4.1 (b) - ActiveX COM Object DLL (Legacy Support) 63
4.2 - Referencing the DLL 65
4.3 - Note on DLL Use in Python / MatLab 66
4.4 - DLL Function Definitions 67
4.4 (a) - DLL Functions for USB Control 67
4.4 (b) - DLL Functions for Ethernet Configuration 67
4.5 - DLL - General Functions 68
4.5 (a) - Connect by Serial Number 68
4.5 (b) - Connect by Address 69
4.5 (c) - Disconnect 70
4.5 (d) - Send SCPI Command 71
4.6 - DLL Function Explanations - Ethernet Configuration 72
4.6 (a) - Get Ethernet Configuration 72
4.6 (b) - Get IP Address 74
4.6 (c) - Get MAC Address 76
4.6 (d) - Get Network Gateway 78
4.6 (e) - Get Subnet Mask 80
4.6 (f) - Get TCP/IP Port 82
4.6 (g) - Get SSH Port 83
4.6 (h) - Get Telnet Port 84
4.6 (i) - Get DHCP Status 85
4.6 (j) - Get Password Status 86
4.6 (k) - Get Password 87
4.6 (l) - Save IP Address 88
4.6 (m) - Save Network Gateway 89
4.6 (n) - Save Subnet Mask 90
4.6 (o) - Save TCP/IP Port 91
4.6 (p) - Save SSH Port 92
4.6 (q) - Save Telnet Port 93
4.6 (r) - Use DHCP 94
4.6 (s) - Use Password 95
4.6 (t) - Set Password 96
5 - USB Control via Direct Programming (Linux) 97
5.1 - USB Interrupt Code Concept 97
5.2 - Interrupts – General Commands 98
5.2 (a) - Send SCPI Command 98
5.3 - Interrupts - Ethernet Configuration Commands 100
5.3 (a) - Set Static IP Address 101

5.3 (b) - Set Static Subnet Mask 102
5.3 (c) - Set Static Network Gateway 103
5.3 (d) - Set HTTP Port 104
5.3 (e) - Set Telnet Port 105
5.3 (f) - Use Password 106
5.3 (g) - Set Password 107
5.3 (h) - Use DHCP 108
5.3 (i) - Get Static IP Address 109
5.3 (j) - Get Static Subnet Mask 110
5.3 (k) - Get Static Network Gateway 111
5.3 (l) - Get HTTP Port 112
5.3 (m) - Get Telnet Port 113
5.3 (n) - Get Password Status 114
5.3 (o) - Get Password 115
5.3 (p) - Get DHCP Status 116
5.3 (q) - Get Dynamic Ethernet Configuration 117
5.3 (r) - Get MAC Address 119
5.3 (s) - Reset Ethernet Configuration 120

1 - Overview

This programming manual is intended for customers wishing to create their own interface for Mini-Circuits' USB & Ethernet controlled, solid-state 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/multissw.html

For details and specifications on the individual models, please see:
https://www.minicircuits.com/WebStore/RF-Solid-State-Rack-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 HTTP or Telnet communication via an Ethernet TCP / IP connection (see Ethernet Control API), which is largely independent of the operating system
2. Using the provided API DLL files (ActiveX or .Net objects) 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))

Setting states and querying the system is achieved using a command set based on SCPI (see SCPI Commands for Control of ZTS Series Switch Systems), sent via one of the above methods.

1.2 - Switch Address Structure

The system is comprised of a controller and a series of solid-state switch modules, each of which can be controlled via its unique internal 2-digit address using the format `ADDRESS:COMMAND`. System queries can be sent to the controller at address 00, although the address component can be omitted for these commands. The switch modules are addressed from `01` to `nn`, depending on the specific ZTS model.

The special address `SL` can be used to send the commands to all switch modules in the system.

2 - SCPI Commands for Control of ZTS Series Switch Systems

The recommended method for setting states and querying the system is a series of 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.

2.1 - System Queries

These queries return information on the complete ZTS Series system. Since they apply to the system's internal controller, the address component (00) can be omitted from the queries.

	Description	Command/Query
a	Get Model Name	:MN?
b	Get Serial Number	:SN?
c	Count Number of Switch Modules	:NumberOfSlaves?
d	Get Firmware	:FIRMWARE?

2.1 (a) - Get Model Name

Description

Returns the Mini-Circuits part number of the ZTS Series system.

Command Syntax

```
:MN?
```

Return String

```
[model]
```

Variable	Description
[model]	Model name of the ZTS Series system

Examples

String to Send	String Returned
:MN?	ZTS-8SP8T-63

See Also

Get Serial Number
Count Number of Switch Modules
Get Firmware

2.1 (b) - Get Serial Number

Description

Returns the serial number of the ZTS Series system.

Command Syntax

```
:SN?
```

Return String

```
[serial]
```

Variable	Description
[serial]	Serial number of the ZTS Series system

Examples

String to Send	String Returned
:SN?	11706010001

See Also

Get Model Name
Count Number of Switch Modules
Get Firmware

2.1 (c) - Count Number of Switch Modules

Description

Returns the number of switch modules within the ZTS Series system.

Command Syntax

```
:NumberOfSlaves?
```

Return String

```
[count]
```

Variable	Description
[count]	The number of switch modules

Examples

String to Send	String Returned
: NumberOfSlaves?	8

See Also

Get Model Name
Get Serial Number
Get Firmware

2.1 (d) - Get Firmware

Description

Returns the firmware version of the ZTS Series system.

Command Syntax

:FIRMWARE?

Return String

[firmware]

Variable	Description
[firmware]	Firmware version name

Examples

String to Send	String Returned
:FIRMWARE?	A1-2

See Also

Get Model Name
Get Serial Number
Count Number of Switch Modules

2.2 - Switch Control

	Description	Command/Query
a	Set Switch State	:[address]:[type]:[channel]:STATE:[port]
b	Query Switch State	:[address]:[type]:[sw_name]:STATE?
e	Set All Switches	:SL:C-[sw_state]
f	Set Multiple Switches	:SL:[[sw_{x-y}]-[$state_{x-y}$]]

2.2 (a) - Set Switch State

Description

Sets the state of a specific switch.

Command Syntax

```
:[address]:[type]:[channel]:STATE:[port]
```

Variable	Description
[module]	The switch type at this address, either SP2T, SP4T, SP8T or SP16T (refer to the block diagram for details)
[channel]	For modules with multiple switches (2SP2T, 4SP2T or 2 SP4T) specify the specific switch channel to set (A to D). To set all switches within the module to the same state, do not specify the channel. This parameter does not apply for single switch modules (SP8T or SP16T).
[port]	The switch port to connect to COM, from 0 to 16 (depending on switch module)

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:01:SP2T:A:STATE:1	:01:1
:01:SP2T:STATE:2	:01:1
:02:SP4T:B:STATE:4	:02:1
:02:SP4T:STATE:1	:02:1
:03:SP8T:STATE:8	:03:1
:04:SP16T:STATE:16	:04:1

See Also

Query Switch State
Set All Switches
Set Multiple Switches

2.2 (b) - Query Switch State

Description

Returns the state of a specific switch.

Command Syntax

```
:[address]:[type]:[channel]:STATE?
```

Variable	Description
[type]	The switch module at this address, either SP2T, SP4T, SP8T or SP16T (refer to the block diagram for details)
[channel]	For modules with multiple switches (2SP2T, 4SP2T or 2 SP4T) specify the specific switch channel to set (A to D). This parameter does not apply for single switch modules (SP8T or SP16T).

Return String

```
[port]
```

Variable	Description
[port]	The switch port to which COM is connected, from 0 to 16 (depending on switch module)

Examples

String to Send	String Returned
:01:SP2T:A:STATE?	:01:1
:02:SP4T:B:STATE?	:02:4
:03:SP8T:STATE?	:03:8
:04:SP16T:STATE?	:04:16

See Also

Set Switch State
Set All Switches
Set Multiple Switches

2.2 (c) - Set All Switches

Description

Sets all switches to the same state.

Command Syntax

```
:SL:C-[port]
```

Variable	Description
[port]	The switch port to which COM should be set for all switches.

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SL:C-1	1
:SL:C-8	1

See Also

Set Switch State
Query Switch State
Set Multiple Switches

2.2 (d) - Set Multiple Switches

Description

Takes a string of switch number / state pairs to set multiple switches with a single command. Any combination of switches can be set so the command string does not need to be supplied in ascending switch address order.

Command Syntax

:SL:[sw_1]-[$port_1$]:[sw_2]-[$port_2$]:[sw_3]-[$port_3$] … :[sw_n]-[$port_n$]

Variable	Description
[sw_1]	The first switch number to set
$port_1$	The state to which the first switch is to be set
…	
[sw_n]	The *n*th switch number to set
$port_n$	The state to which the *n*th switch is to be set

Return String

[status]

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SL:1-2:2-2:3-7	1
:SL:1-3:2-7:3-5:4-4:5-1:6-3:7-4:8-6	1
:SL:1A-2:1B-2:1C-1:1D-1	

See Also

Set Switch State
Query Switch State
Set All Switches

2.3 - Manual Trigger Control

These functions apply to ZTS Series systems specified with external trigger ports.

	Description	Command/Query
a	Reset Trigger In	:ClearTriggerInStatus
b	Query Trigger In	:TriggerInStatus
c	Set Trigger Out	:InitTriggerOut

2.3 (a) - Reset Trigger In

Description

Only applicable to ZTS Series models specified with external trigger ports. Resets the internal Trigger In flag. Should be used after a trigger input has been processed so that the next trigger input signal can be detected.

Command Syntax

```
:ClearTriggerInStatus
```

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:ClearTriggerInStatus	1

See Also

Query Trigger In
Set Trigger Out

2.3 (b) - Query Trigger In

Description

Only applicable to ZTS Series models specified with external trigger ports. Returns the state of the internal Trigger In flag, to identify whether a trigger input signal has been received.

Command Syntax

```
:TriggerInStatus?
```

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Trigger In port at logic 0 (low)
	1	Trigger In port at logic 1 (high)

Examples

String to Send	String Returned
:TriggerInStatus?	0
:TriggerInStatus?	1

See Also

Reset Trigger In
Set Trigger Out

2.3 (c) - Set Trigger Out

Description

Only applicable to ZTS Series models specified with external trigger ports. Creates a trigger output signal (TTL high) of 100 µs duration, then resets to TLL low.

Command Syntax

```
:InitTriggerOut
```

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:InitTriggerOut	1

See Also

Reset Trigger In
Query Trigger In

2.4 - Switch Sequence Configuration

These commands allow a sequence of switch states to be programmed.

	Description	Command/Query
a	Set Number of Steps	:SEQ:STEPS:[steps]
b	Query Number of Steps	:SEQ:STEPS?
c	Set Indexed Step	:SEQ:STEP:[index]
d	Query Indexed Step Number	:SEQ:STEP?
e	Set Switch Command for Indexed Step	:SEQ:COMMAND:[command]
f	Query Switch Command for Indexed Step	:SEQ:COMMAND
g	Set Dwell Time for Indexed Step	:SEQ:DWELL:[time]
h	Query Dwell Time for Indexed Step	:SEQ:DWELL?
i	Set Continuous Mode	:SEQ:CONT:[mode]
j	Query Continuous Mode	:SEQ:CONT?
k	Set Number of Cycles	:SEQ:CYCLES:[count]
l	Query Number of Cycles	:SEQ:CYCLES?
m	Set Trigger-In Mode	:SEQ:TRIGGERIN:[mode]
n	Query Trigger-In Mode	:SEQ:TRIGGERIN?
o	Set Trigger-Out Mode	:SEQ:TRIGGEROUT:[mode]
p	Query Trigger-Out Mode	:SEQ:TRIGGEROUT?
q	Start / Stop Switch Sequence	:SEQ:MASTERMODE:[mode]

2.4 (a) - Set Number of Steps

Description

Sets the number of steps to be included in the switch sequence.

Command Syntax

```
:SEQ:STEPS:[steps]
```

Variable	Description
[steps]	The number of steps to program in the sequence

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:STEPS:10	1

See Also

Query Number of Steps
Set Indexed Step
Query Indexed Step Number

2.4 (b) - Query Number of Steps

Description

Returns the number of steps set to be included in the switch sequence.

Command Syntax

```
:SEQ:STEPS?
```

Return String

```
[steps]
```

Variable	Description
[steps]	Number of steps to be included in the switch sequence

Examples

String to Send	String Returned
:SEQ:STEPS?	10

See Also

Set Number of Steps
Set Indexed Step
Query Indexed Step Number

2.4 (c) - Set Indexed Step

Description

Indexes a specific step number (1 to n) so that its parameters (such as dwell time and switch state) can be set.

Command Syntax

```
:SEQ:STEP:[index]
```

Variable	Description
[index]	The step number to be configured

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:STEP:2	1

See Also

Set Number of Steps
Query Number of Steps
Query Indexed Step Number

2.4 (d) - Query Indexed Step Number

Description

Returns the index number (1 to n) of the step currently being configured.

Command Syntax

```
:SEQ:STEPS?
```

Return String

```
[index]
```

Variable	Description
[index]	The step number (1 to n) to be configured

Examples

String to Send	String Returned
:SEQ:STEP?	2

See Also

Set Number of Steps
Query Number of Steps
Set Indexed Step

2.4 (e) - Set Switch Command for Indexed Step

Description

The switch command to be sent for the indexed step in the switch sequence.

Command Syntax

```
:SEQ:COMMAND:[command]
```

Variable	Description
[command]	Any valid switch state command (see Switch Control)

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:STEP:COMMAND:01:SP2T:D:STATE:2	1

See Also

Set Indexed Step
Query Indexed Step Number
Query Switch Command for Indexed Step
Set Dwell Time for Indexed Step
Query Dwell Time for Indexed Step

2.4 (f) - Query Switch Command for Indexed Step

Description

Returns the switch command to be sent for the indexed step in the switch sequence.

Command Syntax

```
:SEQ:COMMAND?
```

Return String

```
[command]
```

Variable	Description
[command]	The switch command to be executed at this step in the switch sequence

Examples

String to Send	String Returned
:SEQ:COMMAND?	:01:SP2T:D:STATE:2

See Also

Set Indexed Step
Query Indexed Step Number
Set Switch Command for Indexed Step
Set Dwell Time for Indexed Step
Query Dwell Time for Indexed Step

2.4 (g) - Set Dwell Time for Indexed Step

Description

Sets the length of time (microseconds) for which the system will pause at the latest switch state, before proceeding with the switch sequence.

Note: If Trigger-In mode is enabled the system will wait for a trigger input signal before loading a switch state, then pause for the specified dwell before checking for the next trigger signal. Alternatively, dwell time can be set to 0 in this mode if each step is to be loaded based only on the trigger signal.

Command Syntax

```
:SEQ:DWELL:[time]
```

Variable	Description
[time]	The length of time in µs to pause at this step before proceeding with the switch sequence

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:DWELL:250	1

See Also

Set Indexed Step
Query Indexed Step Number
Set Switch Command for Indexed Step
Query Switch Command for Indexed Step
Query Dwell Time for Indexed Step
Set Trigger-In Mode

2.4 (h) - Query Dwell Time for Indexed Step

Description

Returns the length of time (microseconds) for which the system will pause at the latest switch state, before proceeding with the switch sequence.

Note: If Trigger-In mode is enabled the system will wait for a trigger input signal before loading a switch state, then pause for the specified dwell before checking for the next trigger signal. Alternatively, dwell time can be set to 0 in this mode if each step is to be loaded based only on the trigger signal.

Command Syntax

```
:SEQ:DWELL?
```

Return String

```
[time]
```

Variable	Description
[time]	The length of time in µs for which the system will pause at this step before proceeding with the switch sequence

Examples

String to Send	String Returned
:SEQ:DWELL?	250

See Also

Set Indexed Step
Query Indexed Step Number
Set Switch Command for Indexed Step
Query Switch Command for Indexed Step
Set Dwell Time for Indexed Step
Set Trigger-In Mode

2.4 (i) - Set Continuous Mode

Description

Sets whether the system should cycle continuously through the programmed switch sequence or execute it once only.

Command Syntax

```
:SEQ:CONT:[mode]
```

Variable	Value	Description
[mode]	0	Continuous mode disabled (sequence will execute once only)
	1	Continuous mode enabled (sequence will repeat continuously)

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:CONT:0	1
:SEQ:CONT:1	1

See Also

Query Continuous Mode
Set Number of Cycles
Query Number of Cycles

2.4 (j) - Query Continuous Mode

Description

Indicates whether the system has been set to cycle continuously through the programmed switch sequence or execute it once only.

Command Syntax

```
:SEQ:CONT?
```

Return String

```
[mode]
```

Variable	Value	Description
[mode]	0	Continuous mode disabled (sequence will execute once only)
	1	Continuous mode enabled (sequence will repeat continuously)

Examples

String to Send	String Returned
:SEQ:CONT?	1

See Also

Set Continuous Mode
Set Number of Cycles
Query Number of Cycles

2.4 (k) - Set Number of Cycles

Description

Sets the number of times that the programmed switch sequence is to be executed.

Note: If Continuous Mode is enabled then the setting for number of cycles will be ignored and the sequence will run continuously. Disable Continuous mode in order to set a number of cycles.

Command Syntax

```
:SEQ:CYCLES:[count]
```

Variable	Description
[count]	The number of cycles / repetitions of the switch sequence to execute

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:CYCLES:5	1

See Also

Set Continuous Mode
Query Continuous Mode
Query Number of Cycles

2.4 (I) - Query Number of Cycles

Description

Queries the number of times that the programmed switch sequence is set to be executed.

Command Syntax

```
:SEQ:CYCLES?
```

Return String

```
[COUNT]
```

Variable	Description
[count]	The number of cycles / repetitions of the switch sequence to be executed

Examples

String to Send	String Returned
:SEQ:CYCLES?	5

See Also

Set Continuous Mode
Query Continuous Mode
Set Number of Cycles

2.4 (m) - Set Trigger-In Mode

Description

Sets how the switch sequence responds to an external trigger input signal:

- Trigger-In enabled:
  - Each step in the sequence is executed when a trigger input is detected
  - The system will then pause for the specified dwell time before checking for the next trigger input signal (dwell time can be set to 0)
- Trigger-In disabled:
  - Each step in the sequence is executed based only on the specified dwell times
  - Trigger input signals are ignored

Command Syntax

```
:SEQ:TRIGGERIN:[mode]
```

Variable	Value	Description
[mode]	0	Trigger-In mode disabled
	1	Trigger-In mode enabled

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:TRIGGERIN:0	1
:SEQ:TRIGGERIN:1	1

See Also

Query Trigger-In Mode
Set Trigger-Out Mode
Query Trigger-Out Mode

2.4 (n) - Query Trigger-In Mode

Description

Queries how the switch sequence responds to an external trigger input signal:

- Trigger-In enabled:
  - Each step in the sequence is executed when a trigger input is detected
  - The system will then pause for the specified dwell time before checking for the next trigger input signal (dwell time can be set to 0)
- Trigger-In disabled:
  - Each step in the sequence is executed based only on the specified dwell times
  - Trigger input signals are ignored

Command Syntax

```
:SEQ:TRIGGERIN?
```

Return String

```
[mode]
```

Variable	Value	Description
[mode]	0	Trigger-In mode disabled
	1	Trigger-In mode enabled

Examples

String to Send	String Returned
:SEQ:TRIGGERIN?	1

See Also

Set Trigger-In Mode
Set Trigger-Out Mode
Query Trigger-Out Mode

2.4 (o) - Set Trigger-Out Mode

Description

Sets whether a trigger output signal is to be set after each step in the switch sequence has been set.

Command Syntax

```
:SEQ:TRIGGEROUT:[mode]
```

Variable	Value	Description
[mode]	0	Trigger-Out mode disabled (no trigger output signal is set)
	1	Trigger-Out mode enabled (trigger output signal is set after each step in the switch sequence)

Return String

```
[status]
```

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:TRIGGEROUT:0	1
:SEQ:TRIGGEROUT:1	1

See Also

Set Trigger-In Mode
Query Trigger-In Mode
Query Trigger-Out Mode

2.4 (p) - Query Trigger-Out Mode

Description

Queries whether a trigger output signal is to be set after each step in the switch sequence has been set.

Command Syntax

```
:SEQ:TRIGGEROUT?
```

Return String

```
[mode]
```

Variable	Value	Description
[mode]	0	Trigger-Out mode disabled (no trigger output signal is to be set)
	1	Trigger-Out mode enabled (trigger output signal is to be set after each step in the switch sequence)

Examples

String to Send	String Returned
:SEQ:TRIGGEROUT?	1

See Also

Set Trigger-In Mode
Query Trigger-In Mode
Set Trigger-Out Mode

2.4 (q) - Start / Stop Switch Sequence

Description

Starts or stops the switch sequence based on the previously defined parameters.

Note: Any command / query sent to the system while a switch sequence is in process will stop the sequence.

Command Syntax

:SEQ:MASTERMODE:[mode]

Variable	Value	Description
[mode]	OFF	Disables / stops the switch sequence
	ON	Enables / starts the switch sequence

Return String

[status]

Variable	Value	Description
[status]	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
:SEQ:MASTERMODE:OFF	1
:SEQ:MASTERMODE:ON	1

2.5 - SCPI - Ethernet Configuration Commands

These functions apply ZTS Series models with firmware A2 or later.

	Description	Command/Query
a	Set Static IP Address	:ETHERNET:CONFIG:IP:[ip]
b	Get Static IP Address	:ETHERNET:CONFIG:IP?
c	Set Static Subnet Mask	:ETHERNET:CONFIG:SM:[mask]
d	Get Static Subnet Mask	:ETHERNET:CONFIG:SM?
e	Set Static Network Gateway	:ETHERNET:CONFIG:NG:[gateway]
f	Get Static Network Gateway	:ETHERNET:CONFIG:NG?
g	Set HTTP Port	:ETHERNET:CONFIG:HTPORT:[port]
h	Get HTTP Port	:ETHERNET:CONFIG:HTPORT?
i	Set Telnet Port	:ETHERNET:CONFIG:TELNETPORT:[port]
j	Get Telnet Port	:ETHERNET:CONFIG:TELNETPORT?
k	Set Password Requirement	:ETHERNET:CONFIG:PWDENABLED:[enabled]
l	Get Password Requirement	:ETHERNET:CONFIG:PWDENABLED?
m	Set Password	:ETHERNET:CONFIG:PWD:[pwd]
n	Get Password	:ETHERNET:CONFIG:PWD?
o	Set DHCP Status	:ETHERNET:CONFIG:DHCPENABLED:[enabled]
p	Get DHCP Status	:ETHERNET:CONFIG:DHCPENABLED?
q	Get MAC Address	:ETHERNET:CONFIG:MAC?
r	Get Current Ethernet Configuration	:ETHERNET:CONFIG:LISTEN?
s	Update Ethernet Settings	:ETHERNET:CONFIG:INIT

2.5 (a) - Set Static IP Address

Description

Sets the IP address to be used by the system for Ethernet communication when using static IP settings. DHCP must be disabled for this setting to apply, otherwise a dynamic IP address will be in use. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:IP:[ip]
```

Variable	Description
[ip]	The static IP address to be used by the system; must be valid and available on the network

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Get Static IP Address
Set Static Subnet Mask
Set Static Network Gateway
Update Ethernet Settings

2.5 (b) - Get Static IP Address

Description

Returns the IP address to be used by the system 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.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:IP?
```

Return String

```
[ip]
```

Variable	Description
[ip]	The static IP address to be used by the system

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:IP?
```

See Also

Set Static IP Address
Get Static Subnet Mask
Get Static Network Gateway
Get Current Ethernet Configuration

2.5 (c) - Set Static Subnet Mask

Description

Sets the subnet mask to be used by the system for Ethernet communication when using static IP settings. DHCP must be disabled for this setting to apply, otherwise a dynamic IP address will be in use. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:SM:[mask]
```

Variable	Description
[mask]	The subnet mask for communication on the network

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Set Static IP Address
Get Static Subnet Mask
Set Static Network Gateway
Update Ethernet Settings

2.5 (d) - Get Static Subnet Mask

Description

Returns the subnet mask to be used by the system 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.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:SM?
```

Return String

```
[mask]
```

Variable	Description
[mask]	The subnet mask for communication on the network

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:SM?
```

See Also

Get Static IP Address
Set Static Subnet Mask
Get Static Network Gateway
Get Current Ethernet Configuration

2.5 (e) - Set Static Network Gateway

Description

Sets the IP address of the network gateway to be used by the system for Ethernet communication when using static IP settings. DHCP must be disabled for this setting to apply, otherwise a dynamic IP address will be in use. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:NG:[gateway]
```

Variable	Description
[gateway]	IP address of the network gateway

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Set Static IP Address
Set Static Subnet Mask
Get Static Network Gateway
Update Ethernet Settings

2.5 (f) - Get Static Network Gateway

Description

Returns the IP address of the network gateway to be used by the system 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.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:NG?
```

Return String

```
[gateway]
```

Variable	Description
[gateway]	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?
```

See Also

Get Static IP Address
Get Static Subnet Mask
Set Static Network Gateway
Get Current Ethernet Configuration

2.5 (g) - Set HTTP Port

Description

Sets the IP port to be used for HTTP communication. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:HTPORT:[port]
```

Variable	Description
[port]	IP port to be used for HTTP communication. The port will need to be included in all HTTP commands if any other than the default port 80 is selected.

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Get HTTP Port
Set Telnet Port
Update Ethernet Settings

2.5 (h) - Get HTTP Port

Description

Gets the IP port to be used for HTTP communication.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:HTPORT?
```

Return String

```
[port]
```

Variable	Description
[port]	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?
```

See Also

Set HTTP Port
Get Telnet Port

2.5 (i) - Set Telnet Port

Description

Sets the IP port to be used for Telnet communication. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:TELNETPORT:[port]
```

Variable	Description
[port]	IP port to be used for Telnet communication. The port will need to be included when initiating a Telnet session if other than the default port 23 is selected.

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Set HTTP Port
Get Telnet Port
Update Ethernet Settings

2.5 (j) - Get Telnet Port

Description

Gets the IP port to be used for Telnet communication.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:TELNETPORT?
```

Return String

```
[port]
```

Variable	Description
[port]	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?
```

See Also

Get HTTP Port
Set Telnet Port

2.5 (k) - Set Password Requirement

Description

Sets whether or not a password is required for Ethernet communication. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:PWDENABLED:[enabled]
```

Variable	Value	Description
[enabled]	0	Password not required for Ethernet communication
	1	Password required for Ethernet communication

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Get Password Requirement
Set Password
Get Password
Update Ethernet Settings

2.5 (I) - Get Password Requirement

Description

Indicates whether or not a password is required for Ethernet communication.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:PWDENABLED?
```

Return String

```
[enabled]
```

Variable	Value	Description
[enabled]	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?
```

See Also

Set Password Requirement
Set Password
Get Password

2.5 (m) - Set Password

Description

Sets the password for Ethernet communication. The password will only be required for communication with the device when password security is enabled. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:PWD:[pwd]
```

Variable	Description
[pwd]	Password to set for Ethernet communication (not case sensitive)

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Set Password Requirement
Get Password Requirement
Get Password
Update Ethernet Settings

2.5 (n) - Get Password

Description

Returns the password for Ethernet communication. The password will only be required for communication with the device when password security is enabled

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:PWD?
```

Return String

```
[pwd]
```

Variable	Description
[pwd]	Password for Ethernet communication (not case sensitive)

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:PWD?
```

See Also

Set Password Requirement
Get Password Requirement
Set Password

2.5 (o) - Set DHCP Status

Description

Enables or disables DHCP (Dynamic Host Control Protocol). When enabled the system will request a valid IP address from the network's DHCP server. When disabled, the system's static IP settings will be used. Changes to the Ethernet configuration only take effect after the Update Ethernet Settings command has been issued.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:DHCPENABLED:[enabled]
```

Variable	Value	Description
[enabled]	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

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Set Static IP Address
Get DHCP Status
Update Ethernet Settings

2.5 (p) - Get DHCP Status

Description

Indicates whether or not DHCP (Dynamic Host Control Protocol) is enabled. When enabled the system will request a valid IP address from the network's DHCP server. When disabled, the system's static IP settings will be used.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:DHCPENABLED?
```

Return String

```
[enabled]
```

Variable	Value	Description
[enabled]	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?
```

See Also

Set Static IP Address
Set DHCP Status
Get Current Ethernet Configuration

2.5 (q) - Get MAC Address

Description

Returns the MAC (Media Access Control) address of the system (a physical hardware address).

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:MAC?
```

Return String

```
[mac]
```

Variable	Description
[mac]	MAC address of the system

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?
```

See Also

Get Static IP Address
Get Static Subnet Mask
Get Static Network Gateway
Get Current Ethernet Configuration

2.5 (r) - Get Current Ethernet Configuration

Description

Returns the Ethernet configuration (IP address, subnet mask and network gateway) that is currently active for the device. If DHCP is enabled this will be the settings issued dynamically by the network's DHCP server. If DHCP is disabled this will be the user configured static IP settings.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:LISTEN?
```

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?
```

See Also

Get Static IP Address
Get Static Subnet Mask
Get Static Network Gateway
Update Ethernet Settings

2.5 (s) - Update Ethernet Settings

Description

Resets the Ethernet controller so that any recently applied changes to the Ethernet configuration can be loaded. Any subsequent commands / queries to the system will need to be issued using the new Ethernet configuration.

Note: If a connection cannot be established after the INIT command has been issued 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.

Requirements

ZTS Series models with firmware A2 or later.

Command Syntax

```
:ETHERNET:CONFIG:INIT
```

Return String

```
[status]
```

Variable	Value	Description
[status]	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
```

See Also

Get Current Ethernet Configuration

3 - Ethernet Control API

Control of the system via Ethernet TCP / IP networks involves sending the SCPI commands / queries detailed above via HTTP or Telnet.

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.

3.1 - Configuring Ethernet Settings

The system can be configured manually with a static IP address or automatically by the network using DHCP (Dynamic Host Control Protocol):

- Dynamic IP (factory default setting)
  - Subnet Mask, Network Gateway and local IP Address are assigned by the network server on each connection
  - The only user controllable parameters are:
    - TCP/IP port (used for HTTP communication), default is port 80
    - Telnet port, default is port 23
    - Password (up to 20 characters; default is no password)
- Static IP
  - All parameters must be specified by the user:
    - IP Address (must be a legal and unique address on the local network)
    - Subnet Mask (subnet mask of the local network)
    - Network gateway (the IP address of the network gateway/router)
    - TCP/IP port (used for HTTP communication), default is port 80
    - Telnet port, default is port 23
    - Password (up to 20 characters; default is no password)

The Ethernet connection details can be configured using the commands summarized in SCPI - Ethernet Configuration Commands, whilst connected by USB or Ethernet.

3.2 - HTTP Communication

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 to send to the switch

Example 1:

http://192.168.100.100:800/PWD=123;:01:1SP2T:A:STATE:1

Explanation:

- The system has IP address 192.168.100.100 and uses port 800
- Password security is enabled and set to “123”

Example 2:

http://10.10.10.10/:01:1SP2T:A:STATE?

Explanation:

- The system has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled

The system will return the result of the command/query as a string of ASCII characters.

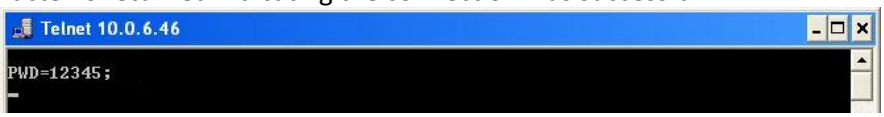
3.3 - Telnet Communication

Communication is started by creating a Telnet connection to the system's IP address. On successful connection the "line feed" character will be returned. If the system has a password enabled then 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:

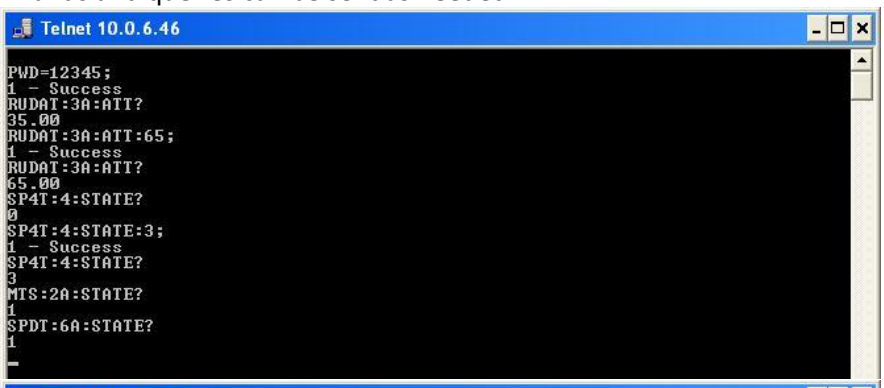
1) Set up Telnet connection to a system with IP address 10.0.6.46:

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 - Success" indicates success:

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



3.4 - 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 solid-state switch systems connected on the network; full control of those units is then accomplished using HTTP or Telnet, as detailed previously.

Alternatively, the IP configuration can be identified or changed by connecting the system with the USB interface (see DLL - Ethernet Configuration Functions).

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

UDP Ports

Mini-Circuits' ZTS Series systems 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 test system's IP address is already known it is not necessary to use UDP.

Transmission

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

Receipt

All systems 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:

MCL_MULTI_SWITCH_CONTROLLER?

Received Data:

Model Name: ZTS-8SP8T-63
Serial Number: 11702120001
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: ZTS-6SP8T-63
Serial Number: 11702120002
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

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
    a. This is the recommended API for most modern operating systems
2) .Net Framework 2.0 DLL
    a. 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
    a. Provided for legacy support of older systems and programming 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/softwaredownload/multissw.html

4.1 - DLL API Options

4.1 (a) - .NET Framework 2.0 DLL (Recommended)

Provided for support of systems with .Net framework version 2.0 or later installed.

Filename: mcl_MultiSwitchController_64.dll

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 (b) - ActiveX COM Object DLL (Legacy Support)

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

Filename: mcl_MultiSwitchController.dll

Requirements

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

Installation

1) Download the latest DLL file from the Mini-Circuits website
2) Copy the .dll file to the correct directory:
   a. 32-bit PC: `C:\WINDOWS\System32`
   b. 64-bit PC: `C:\WINDOWS\SysWOW64`
3) Open the Command Prompt:
   a. For Windows XP®:
      i. Select “All Programs” and then “Accessories” from the Start Menu
      ii. Click on “Command Prompt” to open
   b. For later Windows versions the Command Prompt will need to be run in “Elevated” mode (as an administrator):
      i. Open the Start Menu/Start Screen and type “Command Prompt”
      ii. Right-click on the shortcut for the Command Prompt
      iii. Select “Run as Administrator”
      iv. Enter the administrator credentials if requested
4) Register the DLL using regsvr32:
   a. 32-bit PC: `Regsvr32 mcl_MultiSwitchController.dll`
   b. 64-bit PC (be sure to specify the path for regsvr32 and the DLL):
      `\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\mcl_MultiSwitchController.dll`
5) Hit enter to confirm, a message box will appear to advise of successful registration

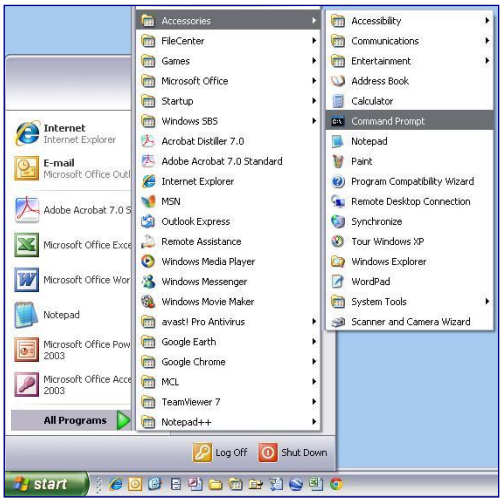


Fig 4.1-a: Opening the Command Prompt in Windows XP

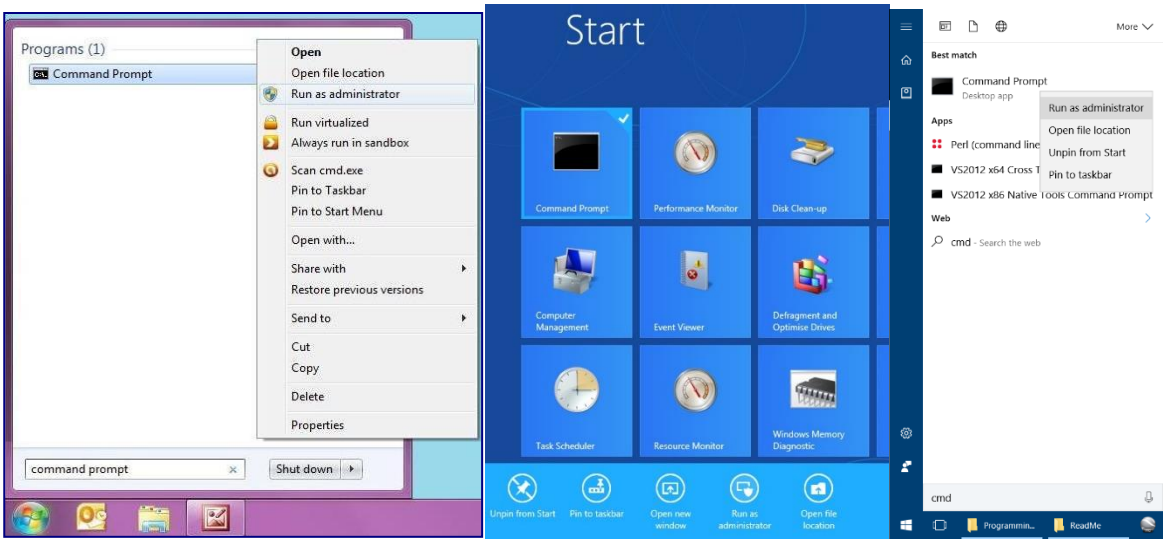


Fig 4.1-b: Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)

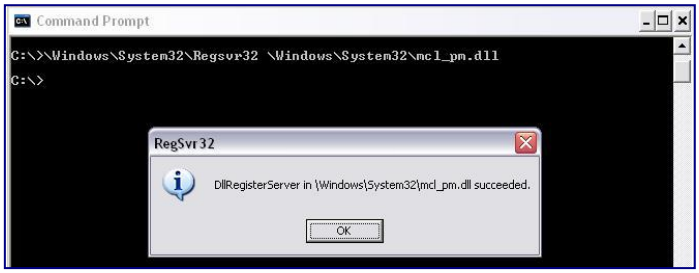


Fig 4.1-c: Registering the DLL in a 32-bit environment

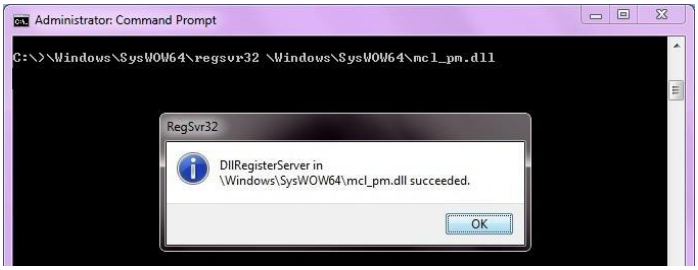


Fig 4.1-d: Registering the DLL in a 64-bit environment

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 DLL (mcl_MultiSwitchController_64.dll)

Python	`import clr  # Import the pythonnet CLR library`
	`clr.AddReference('mcl_MultiSwitchController_64')`
	`from mcl_MultiSwitchController_64 import USB_Control`
	`MyPTE1 = USB_Control()`
	`MyPTE2 = USB_Control()`

Visual Basic	`Public MyPTE1 As New mcl_MultiSwitchController_64.USB_Control` `Public MyPTE2 As New mcl_MultiSwitchController_64.USB_Control`

Visual C++	`mcl_MultiSwitchController_64::USB_Control ^MyPTE1 = gcnew`
	`mcl_MultiSwitchController_64::USB_Control();`
	`mcl_MultiSwitchController_645::USB_Control ^MyPTE2 = gcnew`
	`mcl_MultiSwitchController_64::USB_Control();`

Visual C#	`mcl_MultiSwitchController_64.USB_Control MyPTE1 = new`
	`mcl_MultiSwitchController_64.USB_Control();`
	`mcl_MultiSwitchController_64.USB_Control MyPTE2 = new`
	`mcl_MultiSwitchController_64.USB_Control();`

MatLab	`MCL_SW=NET.addAssembly('C:\Windows\SysWOW64\mcl_MultiSwitchController_64.dll')`
	`MyPTE1 = mcl_MultiSwitchController_64.USB_Control`
	`MyPTE2 = mcl_MultiSwitchController_64.USB_Control`

Example Declarations using the ActiveX DLL (mcl_MultiSwitchController.dll)

Visual Basic	`Public MyPTE1 As New MCL_MultiSwitchController.USB_Control` `Public MyPTE2 As New MCL_MultiSwitchController.USB_Control`

Visual C++	`MCL_MultiSwitchController::USB_Control ^MyPTE1 = gcnew`
	`MCL_MultiSwitchController::USB_Control();`
	`MCL_MultiSwitchController::USB_Control ^MyPTE2 = gcnew`
	`MCL_MultiSwitchController::USB_Control();`

Visual C#	`public MCL_MultiSwitchController.USB_Control MyPTE1 = new`
	`MCL_MultiSwitchController.USB_Control();`
	`public MCL_MultiSwitchController.USB_Control MyPTE2 = new`
	`MCL_MultiSwitchController.USB_Control();`

MatLab	`MyPTE1 = actxserver(MCL_MultiSwitchController.USB_Control')` `MyPTE2 = actxserver(MCL_MultiSwitchController.USB_Control')`

4.3 - Note on 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`

4.4 - DLL Function Definitions

The following functions are defined in both the ActiveX and .Net DLL files. Please see the following sections for a full description of their structure and implementation.

4.4 (a) - DLL Functions for USB Control

a) Short Connect (Optional String SN)
b) Short ConnectByAddress (Optional Short Address)
c) Void Disconnect ()
d) Short Send_SCPI (String SndSTR , Ref String RetSTR)

4.4 (b) - DLL Functions for Ethernet Configuration

a) Short GetEthernet_CurrentConfig (Int IP1, Int IP2, Int IP3, Int IP4, Int Mask1, Int Mask2, Int Mask3, Int Mask4, Int Gateway1, Int Gateway2, Int Gateway3, Int Gateway4)
b) Short GetEthernet_IPAddress (Int b1, Int b2, Int b3, Int b4)
c) Short GetEthernet_MACAddress (Int MAC1 , Int MAC2, Int MAC3, Int MAC4, Int MAC5, Int MAC6)
d) Short GetEthernet_NetworkGateway (Int b1, Int b2, Int b3, Int b4)
e) Short GetEthernet_SubNetMask (Int b1, Int b2, Int b3, Int b4)
f) Short GetEthernet_TCPIPPort (Int port)
g) Short GetEthernet_UseDHCP ()
h) Short GetEthernet_UsePWD ()
i) Short GetEthernet_PWD (string Pwd)
j) Short SaveEthernet_IPAddress (Int b1, Int b2, Int b3, Int b4)
k) Short SaveEthernet_NetworkGateway (Int b1, Int b2, Int b3, Int b4)
l) Short SaveEthernet_SubnetMask (Int b1, Int b2, Int b3, Int b4)
m) Short SaveEthernet_TCPIPPort (Int port)
n) Short SaveEthernet_UseDHCP (Int UseDHCP)
o) Short SaveEthernet_UsePWD (Int UsePwd)
p) Short SaveEthernet_PWD (String Pwd)

4.5 - DLL - General Functions

4.5 (a) - Connect by Serial Number

Declaration

```
Short Connect(Optional String SN)
```

Description

Initializes the USB connection. If multiple ZTS Series systems are connected to the same host computer by USB then the serial number should be included, otherwise this can be omitted. The system should be disconnected on completion of the program using the Disconnect function.

Parameters

Data Type	Variable	Description
String	SN	Optional. The serial number of the test system. Can be omitted if only one switch is connected by USB.

Return Values

Data Type	Value	Description
Short	0	No connection was possible
	1	Connection successfully established
	2	Connection already established (Connect has been called more than once). The system will continue to operate normally.

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);`

See Also

Connect by Address
Disconnect

4.5 (b) - Connect by Address

Declaration

```
Short ConnectByAddress(Optional Short Address)
```

Description

Initialize the USB connection to a ZTS Series switch system 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 system should be disconnected on completion of the program using the Disconnect function.

Parameters

Data Type	Variable	Description
Short	Address	Optional. The address of the system. Can be omitted if only one switch is connected by USB.

Return Values

Data Type	Value	Description
Short	0	No connection was possible
	1	Connection successfully established
	2	Connection already established (Connect has been called more than once)

Examples

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

See Also

Connect by Serial Number
Disconnect

4.5 (c) - Disconnect

Declaration

```
Void Disconnect()
```

Description

This function is called to close the connection to the switch system after completion of the test sequence. It is strongly recommended that this function is used prior to ending the program. Failure to do so may result in a connection problem with the device. Should this occur, shut down the program and unplug the system from the computer, then reconnect to start again.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
None		

Examples

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

See Also

Connect by Serial Number
Connect by Address

4.5 (d) - Send SCPI Command

Declaration

```
Short Send_SCPI(String SndSTR, Ref String RetSTR)
```

Description

Sends a SCPI command / query to set or read the attenuator states and other properties. Refer to SCPI Commands for Control of ZTS Series Switch Systems for the syntax of the SCPI commands.

Parameters

Data Type	Variable	Description
String	SndSTR	Required. The SCPI command to send.
String	RetSTR	Required. User defined string which will be updated with the value returned from the test system.

Return Values

Data Type	Value	Description
Short	0	Command failed
	1	Command completed successfully

Examples

Python	`status = MyPTE1.Send_SCPI(":01:SP2T:A:STATE:1", "")` `response = str(status[1])`
Visual Basic	`status = MyPTE1.Send_SCPI(":01:SP2T:A:STATE:1", response)`
Visual C++	`status = MyPTE1->Send_SCPI(":01:SP2T:A:STATE:1", response);`
Visual C#	`status = MyPTE1.Send_SCPI(":01:SP2T:A:STATE:1", ref(response));`
MatLab	`[status, response] = MyPTE1.Send_SCPI(":01:SP2T:A:STATE:1", '')`

See Also

SCPI Commands for Control of ZTS Series Switch Systems

4.6 - DLL Function Explanations - Ethernet Configuration

4.6 (a) - Get Ethernet Configuration

Declaration

```
Short GetEthernet_CurrentConfig(Ref Int IP1, Ref Int IP2, Ref Int IP3,
          Ref Int IP4, Ref Int Mask1, Ref Int Mask2, Ref Int Mask3, Ref Int Mask4,
           Ref Int Gateway1, Ref Int Gateway2, Ref Int Gateway3, Ref Int Gateway4)
```

Description

This function returns the current IP configuration of the connected multi-channel attenuator system in a series of user defined variables. The settings checked are IP address, subnet mask and network gateway.

Parameters

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

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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
```

See Also

Get MAC Address
Get TCP/IP Port

4.6 (b) - Get IP Address

Declaration

```
Short GetEthernet_IPAddress(Ref Int b1, Ref Int b2, Ref Int b3, Ref Int b4)
```

Description

This function returns the current IP address of the connected system in a series of user defined variables (one per octet).

Parameters

Data Type	Variable	Description
Int	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”).
Int	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”).
Int	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”).
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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`

See Also

Get Ethernet Configuration
Get TCP/IP Port
Save IP Address
Save TCP/IP Port

4.6 (c) - Get MAC Address

Declaration

```
Short GetEthernet_MACAddress(Ref Int MAC1, Ref Int MAC2, Ref Int MAC3,
                             Ref Int MAC4, Ref Int MAC5, Ref Int MAC6)
```

Description

This function returns the MAC (media access control) address, the physical address, of the connected system as a series of decimal values (one for each of the 6 numeric groups).

Parameters

Data Type	Variable	Description
Int	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
Int	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
Int	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
Int	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
Int	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
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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
```

See Also

Get Ethernet Configuration

4.6 (d) - Get Network Gateway

Declaration

```
Short GetEthernet_NetworkGateway(Ref Int b1, Ref Int b2, Ref Int b3, Ref Int b4)
```

Description

This function returns the IP address of the network gateway to which the system is currently connected. A series of user defined variables are passed to the function to be updated with the IP address (one per octet).

Parameters

Data Type	Variable	Description
Int	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”).
Int	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”).
Int	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”).
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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`

See Also

Get Ethernet Configuration
Save Network Gateway

4.6 (e) - Get Subnet Mask

Declaration

```
Short GetEthernet_SubNetMask(Ref Int b1, Ref Int b2, Ref Int b3, Ref Int b4)
```

Description

This function returns the subnet mask used by the network gateway to which the system is currently connected. A series of user defined variables are passed to the function to be updated with the subnet mask (one per octet).

Parameters

Data Type	Variable	Description
Int	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”).
Int	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”).
Int	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”).
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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`

See Also

Get Ethernet Configuration
Save Subnet Mask

4.6 (f) - Get TCP/IP Port

Declaration

```
Short GetEthernet_TCPIPPort(Ref Int port)
```

Description

This function returns the TCP/IP port used by the test system for HTTP communication. The default is port 80.

Note: Port 23 is reserved for Telnet communication and cannot be set as the HTTP port.

Parameters

Data Type	Variable	Description
Int	port	Integer variable which will be updated with the TCP/IP port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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
```

See Also

Save TCP/IP Port
Get SSH Port
Get Telnet Port

4.6 (g) - Get SSH Port

Declaration

```
Short GetEthernet_SSHPort(Ref Int port)
```

Description

This function returns the port 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

Data Type	Variable	Description
Int	port	Integer variable which will be updated with the SSH port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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
```

See Also

Save SSH Port
Get TCP/IP Port
Get Telnet Port

4.6 (h) - Get Telnet Port

Declaration

```
Short GetEthernet_TelnetPort(Ref Int port)
```

Description

This function returns the port used for Telnet communication. The default is port 23.

Parameters

Data Type	Variable	Description
Int	port	Integer variable which will be updated with the Telnet port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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
```

See Also

Save Telnet Port
Get TCP/IP Port
Get SSH Port

4.6 (i) - Get DHCP Status

Declaration

```
Short GetEthernet_UseDHCP()
```

Description

This function indicates whether the test system is using DHCP (dynamic host control protocol), in which case the IP configuration is derived from a network server; or user defined “static” IP settings.

Parameters

Data Type	Variable	Description
None		

Return Values

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

Example

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()`

See Also

Get Ethernet Configuration
Use DHCP

4.6 (j) - Get Password Status

Declaration

```
Short GetEthernet_UsePWD()
```

Description

Indicates whether or not a password is required for HTTP and Telnet communication. The password is always required for SSH communication.

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

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Password not required
Short	1	Password required

Example

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()`

See Also

Get Password
Use Password
Set Password

4.6 (k) - Get Password

Declaration

```
Short GetEthernet_PWD(Ref String Pwd)
```

Description

Returns the password for Ethernet communication. The password is always required for SSH communication but only for HTTP and Telnet when password security is enabled.

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

Parameters

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

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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`

See Also

Get Password Status
Use Password
Set Password

4.6 (I) - Save IP Address

Declaration

```
Short SaveEthernet_IPAddress(Int b1, Int b2, Int b3, Int b4)
```

Description

This function sets a static IP address to be used by the connected test system.

Note: this could subsequently be overwritten automatically if DHCP is enabled (see Use DHCP).

Parameters

Data Type	Variable	Description
Int	IP1	Required. First (highest order) octet of the IP address to set (for example “192” for the IP address “192.168.1.0”).
Int	IP2	Required. Second octet of the IP address to set (for example “168” for the IP address “192.168.1.0”).
Int	IP2	Required. Third octet of the IP address to set (for example “1” for the IP address “192.168.1.0”).
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get Ethernet Configuration
Get IP Address

4.6 (m) - Save Network Gateway

Declaration

```
Short SaveEthernet_NetworkGateway(Int b1, Int b2, Int b3, Int b4)
```

Description

This function sets the IP address of the network gateway to which the system should connect.

Note: this could subsequently be overwritten automatically if DHCP is enabled (see Use DHCP).

Parameters

Data Type	Variable	Description
Int	IP1	Required. First (highest order) octet of the network gateway IP address (for example “192” for the IP address “192.168.1.0”).
Int	IP2	Required. Second octet of the network gateway IP address (for example “168” for the IP address “192.168.1.0”).
Int	IP2	Required. Third octet of the network gateway IP address (for example “1” for the IP address “192.168.1.0”).
Int	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

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get Ethernet Configuration
Get Network Gateway

4.6 (n) - Save Subnet Mask

Declaration

```
Short SaveEthernet_SubnetMask(Int b1, Int b2, Int b3, Int b4)
```

Description

This function sets the subnet mask of the network to which the system should connect.

Note: this could subsequently be overwritten automatically if DHCP is enabled (see Use DHCP).

Parameters

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

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get Ethernet Configuration
Get Subnet Mask

4.6 (o) - Save TCP/IP Port

Declaration

```
Short SaveEthernet_TCPIPPort(Int port)
```

Description

This function sets the TCP/IP port used by the system for HTTP communication. The default is port 80.

Note: Port 23 is reserved for Telnet communication and cannot be set as the HTTP port.

Parameters

Data Type	Variable	Description
Int	port	Numeric value of the TCP/IP port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get TCP/IP Port
Save TCP/IP Port
Save SSH Port
Save Telnet Port

4.6 (p) - Save SSH Port

Declaration

```
Short SaveEthernet_SSHPort(Int port)
```

Description

This function sets the 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

Data Type	Variable	Description
Int	port	Numeric value of the SSH port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get SSH Port
Save TCP/IP Port
Save Telnet Port

4.6 (q) - Save Telnet Port

Declaration

```
Short SaveEthernet_TelnetPort(Int port)
```

Description

This function sets the port to be used for Telnet communication. The default is port 23.

Parameters

Data Type	Variable	Description
Int	port	Numeric value of the Telnet port.

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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

See Also

Get Telnet Port
Save TCP/IP Port
Save SSH Port
Save Telnet Port

4.6 (r) - Use DHCP

Declaration

```
Short SaveEthernet_UseDHCP(Int UseDHCP)
```

Description

This function enables or disables DHCP (dynamic host control protocol). When enabled the IP configuration of the system is assigned automatically by the network server; when disabled the user defined “static” IP settings apply.

Parameters

Data Type	Variable	Description
Int	UseDHCP	Required. Integer value to set the DHCP mode: 0 - DHCP disabled (static IP settings used) 1 - DHCP enabled (IP setting assigned by network)

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get DHCP Status

4.6 (s) - Use Password

Declaration

```
Short SaveEthernet_UsePWD(Int UsePwd)
```

Description

Sets whether or not a password is required for HTTP and Telnet communication. The password is always required for SSH communication.

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

Parameters

Data Type	Variable	Description
Int	UseDHCP	Required. Integer value to set the password mode: 0 – Password not required 1 – Password required

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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);`

See Also

Get Password Status
Get Password
Set Password

4.6 (t) - Set Password

Declaration

```
Short SaveEthernet_PWD(String Pwd)
```

Description

Sets the password for Ethernet communication. The password is always required for SSH communication but only for HTTP and Telnet when password security is enabled.

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

Parameters

Data Type	Variable	Description
String	Pwd	Required. The password to set (20 characters maximum).

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1	Command completed successfully

Example

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");`

See Also

Get Password Status
Get Password
Use Password

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 a ZTS Series switch system, the Vendor ID and Product ID are required:

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

Communication with the system 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 system.

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 – General Commands

5.2 (a) - Send SCPI Command

Description

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

Transmit Array

Byte	Data	Description
0	1 or *	Interrupt code for Send SCPI Command
1 - 63	SCPI Transmit String	The SCPI command to send represented as a series of ASCII character codes, one character code per byte

Returned Array

Byte	Data	Description
0	1 or *	Interrupt code for Send SCPI Command
1 to (n-1)	SCPI Return String	The 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	“Don’t care” bytes, can be any value

Example (Get Model Name of Master)

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	1	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 USB-1SP8T-63 would be as follows:

Byte	Data	Description
0	1	Interrupt code for Send SCPI Command
1	90	ASCII character code for Z
2	84	ASCII character code for T
3	83	ASCII character code for S
4	45	ASCII character code for -
5	49	ASCII character code for 8
6	83	ASCII character code for S
7	80	ASCII character code for P
8	56	ASCII character code for 8
9	84	ASCII character code for T
10	45	ASCII character code for -
11	54	ASCII character code for 6
12	51	ASCII character code for 3
13	0	Zero value byte to indicate end of string

See Also

SCPI Commands for Control of ZTS Series Switch Systems

5.3 - Interrupts - Ethernet Configuration Commands

	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 (a) - 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 (b) - 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 (c) - 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 (d) - 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 (e) - 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 (f) - 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 (g) - Set Password

Description

Sets the password to be used for Ethernet communication (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 (h) - Use DHCP

Description

Enables or disables DHCP (dynamic host control protocol). With DHCP enabled, the 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	207	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	207	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 (i) - 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 (j) - 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 (k) - 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 (I) - 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 (m) - 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 (n) - Get Password Status

Description

Checks whether the system 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 (o) - Get Password

Description

Gets the password to be used for Ethernet communication (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 (p) - Get DHCP Status

Description

Checks whether DHCP (dynamic host control protocol) is enabled or disabled. With DHCP enabled, the 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 (q) - Get Dynamic Ethernet Configuration

Description

Returns the IP address, subnet mask and default gateway currently used by the system. 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 (r) - Get MAC Address

Description

Returns the MAC address.

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 (s) - Reset Ethernet Configuration

Description

Resets the controller so that the latest Ethernet configuration is adopted. Must be sent after any changes are made to the Ethernet 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