

Test Solutions - Programming Manual

High Isolation Solid State Switches



 Mini-Circuits®

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

Important Notice

This guide is owned by Mini-Circuits and is protected by copyright, trademark and other intellectual property laws.

The information in this guide is provided by Mini-Circuits as an accommodation to our customers and may be used only to promote and accompany the purchase of Mini-Circuits' Parts. This guide may not be reproduced, modified, distributed, published, stored in an electronic database, or transmitted and the information contained herein may not be exploited in any form or by any means, electronic, mechanical recording or otherwise, without prior written permission from Mini-Circuits.

This guide is subject to change, qualifications, variations, adjustments or modifications without notice and may contain errors, omissions, inaccuracies, mistakes or deficiencies. Mini-Circuits assumes no responsibility for, and will have no liability on account of, any of the foregoing. Accordingly, this guide should be used as a guideline only.

Trademarks

Microsoft, Windows, Visual Basic, Visual C# and Visual C++ are registered trademarks of Microsoft Corporation. LabVIEW and CVI are registered trademarks of National Instruments Corporation. Delphi is a registered trademark of Delphi Technologies, Inc. MATLAB is a registered trademark of The MathWorks, Inc. Agilent VEE is a registered trademark of Agilent Technologies, Inc. Linux is a registered trademark of Linus Torvalds. Mac is a registered trademark of Apple Inc. Python is a registered trademark of Python Software Foundation Corporation.

All other trademarks cited within this guide are the property of their respective owners. Neither Mini-Circuits nor the Mini-Circuits PTE (portable test equipment) series are affiliated with or endorsed or sponsored by the owners of the above referenced trademarks.

Mini-Circuits and the Mini-Circuits logo are registered trademarks of Scientific Components Corporation.

Mini-Circuits

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

1 - Overview & Supported Models	5
1.1 - Control Methods.....	5
2 - Mini-Circuits' Daisy-Chain Control Concept	6
2.1 - Addressing Individual Switch Modules in the Daisy-Chain.....	6
2.2 - Addressing Standalone Switch Modules.....	6
3 - SCPI Commands for Switch Control	7
3.1 - System Identification Commands / Queries	7
3.1 (a) - Assign Addresses.....	8
3.1 (b) - Count Number of Slaves	9
3.1 (c) - Get Model Name	10
3.1 (d) - Get Serial Number	11
3.1 (e) - Get Firmware Version	12
3.2 - Switch State Commands / Queries	13
3.2 (a) - Set Switch State	13
3.2 (b) - Get Switch State.....	14
3.3 - Switch Sequence Configuration	15
3.3 (a) - Set Number of Steps	16
3.3 (b) - Query Number of Steps	17
3.3 (c) - Set Indexed Step.....	18
3.3 (d) - Query Indexed Step Number	19
3.3 (e) - Set State of Indexed Step.....	20
3.3 (f) - Query State of Indexed Step.....	21
3.3 (g) - Set Dwell Time for Indexed Step.....	22
3.3 (h) - Query Dwell Time for Indexed Step.....	23
3.3 (i) - Set Dwell Time Units	24
3.3 (j) - Query Dwell Time Units	25
3.3 (k) - Set Number of Cycles	26
3.3 (l) - Query Number of Cycles	27
3.3 (m) - Set Sequence Direction.....	28
3.3 (n) - Query Sequence Direction	29
3.3 (o) - Start / Stop Switch Sequence.....	30
4 - USB Control API for Microsoft Windows	31
4.1 - DLL API Options	31
4.1 (a) - .NET Framework 4.5 DLL (Recommended)	31
4.1 (b) - .NET Framework 2.0 DLL (Legacy Support).....	31
4.1 (c) - ActiveX COM Object DLL (Legacy Support)	32
4.2 - Referencing the DLL	34
4.3 - Note on DLL Use in Python / MatLab	35
4.4 - DLL Function Definitions	36
4.5 - DLL Function Explanations	37
4.5 (a) - Connect to Switch	37
4.5 (b) - Connect to Switch by Address	38
4.5 (c) - Disconnect from Switch	39
4.5 (d) - Read Model Name of Switch	40
4.5 (e) - Read Serial Number of Switch	41
4.5 (f) - Set SP4T Switch.....	42
4.5 (g) - Get SP4T Switch State	42
4.5 (h) - Send SCPI Switch Command	43
4.5 (i) - Set Address of Switch.....	44
4.5 (j) - Get Address of Switch.....	45
4.5 (k) - Get List of Connected Serial Numbers	46

4.5 (l) - Get List of Available Addresses	47
4.5 (m) - Get USB Connection Status.....	48
4.5 (n) - Get USB Device Name	49
4.5 (o) - Get Firmware	50
4.5 (p) - Get Firmware Version (Antiquated)	51
4.6 - Switching Sequence DLL Functions.....	52
4.6 (a) - Set Number of Steps	52
4.6 (b) - Get Number of Steps	53
4.6 (c) - Set Step	54
4.6 (d) - Get Step Switch State	56
4.6 (e) - Get Step Dwell Time	57
4.6 (f) - Get Step Dwell Time Units.....	58
4.6 (g) - Set Sequence Direction	59
4.6 (h) - Get Sequence Direction	60
4.6 (i) - Set Number of Cycles.....	61
4.6 (j) - Get Number of Cycles	62
4.6 (k) - Enable / Disable Continuous Mode.....	63
4.6 (l) - Check Continuous Mode State.....	64
4.6 (m) - Start Sequence.....	65
4.6 (n) - Stop Sequence	66
5 - USB Control via Direct Programming (Linux)	67
5.1 - USB Interrupt Code Concept.....	67
5.2 - Summary of Commands	67
5.2 (a) - Core Commands.....	67
5.2 (b) - Switching Sequence Commands.....	68
5.3 - Core Commands.....	69
5.3 (a) - Get Device Model Name	69
5.3 (b) - Get Device Serial Number.....	70
5.3 (d) - Send SCPI Switch Command	71
5.3 (e) - Get Firmware	73
5.4 - Switching Sequence Commands	74
5.4 (a) - Set Number of Steps	74
5.4 (b) - Get Number of Steps	75
5.4 (c) - Set Step	76
5.4 (d) - Get Step	78
5.4 (e) - Set Direction	80
5.4 (f) - Get Direction.....	81
5.4 (g) - Set Number of Cycles	82
5.4 (h) - Get Number of Cycles	83
5.4 (i) - Enable / Disable Continuous Mode	84
5.4 (j) - Check Continuous Mode State.....	85
5.4 (k) - Start / Stop Sequence	86

1 - Overview & Supported Models

This Programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' high isolation solid-state switches. The contents apply to:

Model Name	Switch Type	Frequency (MHz)	Switch Count	Control Interface
U2C-1SP2T-63VH	SPDT	10 - 6000 MHz	1	USB / I ² C / SPI
USB-4SP2T-63H		10 - 6000 MHz	4	USB
USB-2SP2T-DCH		DC - 8000 MHz	2	USB
USB-1SP2T-183		100 MHz - 18 GHz	1	USB
USB-1SP2T-34		100 MHz - 30 GHz	1	USB
USB-1SP2T-A44		100 MHz - 43.5 GHz	1	USB
U2C-1SP4T-63H	SP4T	2 - 6000 MHz	1	USB / I ² C
USB-2SP4T-63H		10 - 6000 MHz	2	USB
USB-1SP4T-183		100 MHz - 18 GHz	1	USB
USB-1SP4T-34		100 MHz - 30 GHz	1	USB
USB-1SP8T-63H	SP8T	10 - 6000 MHz	1	USB
USB-1SP8T-183		100 MHz - 18 GHz	1	USB
USB-1SP8T-34		100 MHz - 30 GHz	1	USB
USB-1SP16T-83H	SP16T	1 - 8000 MHz	1	USB / TTL

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/solidstate.html>

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

<https://www.minicircuits.com/WebStore/RF-Solid-State-Compact-Switch.html>

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

1.1 - Control Methods

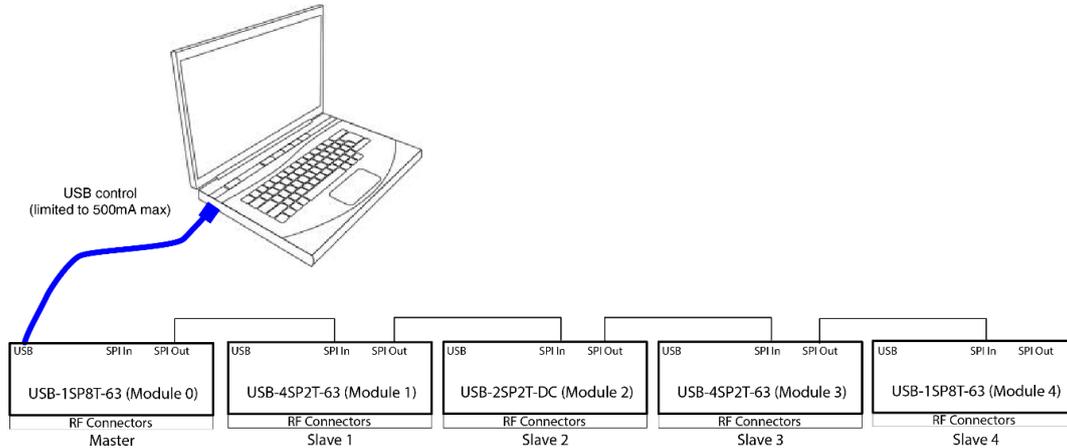
Setting and reading the switch states is accomplished using a series of ASCII / SCPI commands, sent using either of the following approaches:

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

Additionally, SPI, I²C and TTL interfaces are available on some models for logic-based control systems. Refer to the individual model datasheets for details and instructions on these approaches.

2 - Mini-Circuits' Daisy-Chain Control Concept

The daisy-chain control concept is designed to allow flexible switch systems to be configured and re-configured quickly and without the need to re-write the control software each time. Multiple switch modules of different types can be cascaded or "daisy-chained" together via the respective SPI In and Out connections. Only the first switch module in the chain (the "Master") is connected to a PC via its USB connection. All other modules (the "Slaves") receive their instructions and issue their replies through the Master.



The DC supply is also passed along the daisy-chain but it is important to note the maximum current available from the source and the maximum "pass-through" current supported by the individual switch modules (typically 500 mA). Additional power supplies can be connected to specific modules along the daisy-chain as required to ensure sufficient current to each module. Refer to the individual module datasheets for the current supply and pass-through requirements.

2.1 - Addressing Individual Switch Modules in the Daisy-Chain

Any supported switch module can assume the role of the Master, it is the first switch in the daisy-chain and the only one with a USB data connection. Each switch module in the chain is dynamically assigned an address when the USB control connection is established with the Master. The Master address is always 00, with each subsequent Slave in the chain being assigned a 2-digit address, in sequence from 01 to *nn*.

If new switch modules are subsequently added to the daisy-chain, or the order is changed, then the following commands can be issued to refresh the addresses and check the number of connected modules.

`:AssignAddresses` Re-assign the addresses for the complete daisy-chain, starting from the 00 for the Master
`:NumberOfSlaves` Query the number of slave devices connected in the daisy-chain

The full list of SCPI control commands / queries for identifying and controlling the switches are summarised in the following sections. These commands can be directed to any specific switch module by including the 2 digit address at the beginning of the string in the format shown below. Any commands sent without an address component will be directed to the Master (address 00).

For example:

`:MN?` Return the model name of the Master module (address 00)
`:00:MN?` Return the model name of the Master module (address 00)
`:01:MN?` Return the model name of the first Slave module (address 01)
`:02:MN?` Return the model name of the second Slave module (address 02)

2.2 - Addressing Standalone Switch Modules

Where only 1 switch module is connected by USB, with no additional daisy-chained switches, the address component is not required when sending the SCPI commands.

3 - SCPI Commands for Switch Control

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.

3.1 - System Identification Commands / Queries

	Description	Command/Query
a	Assign Address	:AssignAddresses
b	Count Number of Slaves	:NumberOfSlaves?
c	Get Model Name	:MN?
d	Get Serial Number	:SN?
e	Get Firmware	:FIRMWARE?

3.1 (a) - Assign Addresses

Applies when multiple switch modules are connected together in a daisy-chain configuration. The Master will automatically detect and issue addresses to all connected Slave units as soon as the USB connection to the Master is initiated. This command can be issued after making any changes to the order of SPI connections (adding or swapping switch modules) to reissue addresses to all connected switch modules.

Note: addresses are always issued in the order that Slaves are connected to the Master through the SPI ports. Therefore, changing the order of connection of Slave units also changes their addresses.

Command Syntax

`:AssignAddresses`

Return String

`[status]`

Variable	Value	Description
<code>[status]</code>	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
<code>:AssignAddresses</code>	1

See Also

DLL - [Send SCPI Switch Command](#)
Interrupt - [Send SCPI Switch Command](#)
[Count Number of Slaves](#)

3.1 (b) - Count Number of Slaves

Returns the number of Slave modules connected to the Master in a daisy-chain configuration. Each module can then be queried and controlled using its unique address.

Command Syntax

`:NumberOfSlaves?`

Return String

`[count]`

Variable	Description
<code>[count]</code>	The total number of Slaves connected into the Master

Examples

String to Send	String Returned
<code>:NumberOfSlaves</code>	1

See Also

DLL - [Send SCPI Switch Command](#)
Interrupt - [Send SCPI Switch Command](#)
[Assign Addresses](#)

3.1 (c) - Get Model Name

Returns the switch model name

Command Syntax

`:MN?`

Return String

`[model]`

Variable	Description
<code>[model]</code>	Model name of the connected switch

Examples

String to Send	String Returned
<code>:MN?</code>	<code>USB-1SP8T-63H</code>
<code>:00:MN?</code>	<code>00:USB-1SP8T-63H</code>
<code>:01:MN?</code>	<code>01:USB-1SP16T-83H</code>

See Also

[DLL - Send SCPI Switch Command](#)
[Interrupt - Send SCPI Switch Command](#)
[Get Serial Number](#)
[Get Firmware Version](#)

3.1 (d) - Get Serial Number

Returns the switch serial number.

Command Syntax

`:SN?`

Return String

`[serial]`

Variable	Description
<code>[serial]</code>	Serial number of the connected switch

Examples

String to Send	String Returned
<code>:SN?</code>	11807030001
<code>:00:SN?</code>	00:11807030001
<code>:01:SN?</code>	01:11807030002

See Also

[DLL - Send SCPI Switch Command](#)

[Interrupt - Send SCPI Switch Command](#)

[Get Model Name](#)

[Get Firmware Version](#)

3.1 (e) - Get Firmware Version

Description

Returns the internal firmware version of the connected switch.

Command Syntax

`:FIRMWARE?`

Return String

`[firmware]`

Variable	Description
<code>[firmware]</code>	Firmware version of the connected switch

Examples

String to Send	String Returned
<code>:FIRMWARE?</code>	A0
<code>:00:FIRMWARE?</code>	00:A0
<code>:01:FIRMWARE?</code>	01:A0

See Also

[DLL - Send SCPI Switch Command](#)

[Interrupt - Send SCPI Switch Command](#)

[Get Model Name](#)

[Get Serial Number](#)

3.2 - Switch State Commands / Queries

	Description	Command / Query
a	Set Switch State	[Sw_Type]:[Sw_Channel]:STATE:[Sw_State]
b	Get Switch State	[Sw_Type]:[Sw_Channel]:STATE?

3.2 (a) - Set Switch State

Sets the state of a specific switch.

Command Syntax

`: [Sw_Type] : [Sw_Channel] : STATE : [Sw_State]`

Variable	Description
[Sw_Type]	The switch type being addressed, "SP2T", "SP4T", "SP8T", or "SP16T"
[Sw_Channel]	For modules with multiple switches specify the individual switch channel to set (A to D)
[Sw_State]	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
<code>:SP2T:A:STATE:1</code>	1
<code>:SP4T:B:STATE:4</code>	1
<code>:SP16T:STATE:16</code>	1
<code>:00:SP16T:STATE:16</code>	00:1
<code>:01:SP16T:STATE:16</code>	01:1

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Get Switch State](#)

3.2 (b) - Get Switch State

Returns the state of a specific switch.

Command Syntax

```
: [Sw_Type] : [Sw_Channel] : STATE?
```

Variable	Description
[Sw_Type]	The switch type being addressed, "SP2T", "SP4T", "SP8T", or "SP16T"
[Sw_Channel]	For modules with multiple switches specify the individual switch channel (A to D)

Return String

```
[State]
```

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

Examples

String to Send	String Returned
: SP2T:A: STATE?	1
: SP4T:B: STATE?	4
: SP16T: STATE?	16
: 00: SP16T: STATE?	00:16
: 01: SP16T: STATE?	01:16

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Switch State](#)

3.3 - Switch Sequence Configuration

	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 State of Indexed Step	:SEQ:STATE:[state]
f	Query State of Indexed Step	:SEQ:STATE?
g	Set Dwell Time for Indexed Step	:SEQ:DWELLTIME:[time]
h	Query Dwell Time for Indexed Step	:SEQ:DWELLTIME?
i	Set Dwell Time Units	:SEQ:DWELLUNITS:[units]
j	Get Dwell Time Units	:SEQ:DWELLUNITS?
k	Set Number of Cycles	:SEQ:CYCLES:[count]
l	Query Number of Cycles	:SEQ:CYCLES?
m	Set Sequence Direction	:SEQ:DIRECTION:[mode]
n	Get Sequence Direction	:SEQ:DIRECTION?
o	Start / Stop Switch Sequence	:SEQ:MODE:[mode]

3.3 (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
<code>[steps]</code>	The number of steps to program in the sequence

Return String

`[status]`

Variable	Value	Description
<code>[status]</code>	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
<code>:SEQ:STEPS:10</code>	1

See Also

- DLL - [Send SCPI Switch Command](#)
- Interrupt - [Send SCPI Switch Command](#)
- [Query Number of Steps](#)
- [Set Indexed Step](#)
- [Query Indexed Step Number](#)

3.3 (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
<code>[steps]</code>	Number of steps to be included in the switch sequence

Examples

String to Send	String Returned
<code>:SEQ:STEPS?</code>	10

See Also

[DLL - Send SCPI Switch Command](#)
[Interrupt - Send SCPI Switch Command](#)
[Set Number of Steps](#)
[Set Indexed Step](#)
[Query Indexed Step Number](#)

3.3 (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

- DLL - [Send SCPI Switch Command](#)
- Interrupt - [Send SCPI Switch Command](#)
- [Set Number of Steps](#)
- [Query Number of Steps](#)
- [Query Indexed Step Number](#)

3.3 (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
<code>[index]</code>	The step number (1 to n) to be configured

Examples

String to Send	String Returned
<code>:SEQ:STEP?</code>	2

See Also

[DLL - Send SCPI Switch Command](#)
[Interrupt - Send SCPI Switch Command](#)
[Set Number of Steps](#)
[Query Number of Steps](#)
[Set Indexed Step](#)

3.3 (e) - Set State of Indexed Step

Description

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

Command Syntax

`:SEQ:STATE:[state]`

Variable	Description
[state]	The port number to set for the switch at this step (0 to 16, model dependent). For modules containing multiple switches (eg: USB-4SP2t-63H), the states for all switches should be listed, separated by colons.

Return String

[status]

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

Examples

String to Send	String Returned
<code>:SEQ:STATE:1:2:2:1</code>	1
<code>:SEQ:STATE:8</code>	1

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Indexed Step](#)
- [Query Indexed Step Number](#)
- [Query State of Indexed Step](#)
- [Set Dwell Time for Indexed Step](#)
- [Query Dwell Time for Indexed Step](#)

3.3 (f) - Query State of Indexed Step

Description

Returns the switch state to be set for the indexed step in the switch sequence.

Command Syntax

`:SEQ:STATE?`

Return String

`[state]`

Variable	Description
<code>[state]</code>	The port number to be set for the switch at this step (0 to 16, model dependent). For modules containing multiple switches (eg: USB-4SP2t-63H), the states for all switches should be listed, separated by colons.

Examples

String to Send	String Returned
<code>:SEQ:STATE?</code>	<code>1:2:2:1</code>
<code>:SEQ:STATE?</code>	<code>8</code>

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Indexed Step](#)
- [Query Indexed Step Number](#)
- [Set State of Indexed Step](#)
- [Set Dwell Time for Indexed Step](#)
- [Query Dwell Time for Indexed Step](#)

3.3 (g) - Set Dwell Time for Indexed Step

Description

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

Command Syntax

```
:SEQ:DWELLTIME:[time]
```

Variable	Description
[time]	The length of time in 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:DWELLTIME:250	1

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Indexed Step](#)
- [Query Indexed Step Number](#)
- [Query Dwell Time for Indexed Step](#)

3.3 (h) - Query Dwell Time for Indexed Step

Description

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

Command Syntax

```
:SEQ:DWELLTIME?
```

Return String

```
[time]
```

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

Examples

String to Send	String Returned
:SEQ:DWELLTIME?	250

See Also

[DLL - Send SCPI Switch Command](#)
[Interrupt - Send SCPI Switch Command](#)
[Set Indexed Step](#)
[Query Indexed Step Number](#)
[Set Dwell Time for Indexed Step](#)

3.3 (i) - Set Dwell Time Units

Description

Sets the units of time to be used for the dwell time at the indexed step.

Command Syntax

```
:SEQ:DWELLUNITS:[units]
```

Variable	Value	Description
[units]	U	Microseconds
	M	Milliseconds
	S	Seconds

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:SEQ:DWELLUNITS:U	1
:SEQ:DWELLUNITS:M	1
:SEQ:DWELLUNITS:S	1

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Dwell Time for Indexed Step](#)
- [Query Dwell Time for Indexed Step](#)
- [Query Dwell Time Units](#)

3.3 (j) - Query Dwell Time Units

Description

Returns the units of time to be used for the dwell time at the indexed step.

Command Syntax

`:SEQ:DWELLUNITS?`

Return String

`[units]`

Variable	Value	Description
<code>[units]</code>	U	Microseconds
	M	Milliseconds
	S	Seconds

Examples

String to Send	String Returned
<code>:SEQ:DWELLUNITS?</code>	U
<code>:SEQ:DWELLUNITS?</code>	M
<code>:SEQ:DWELLUNITS?</code>	S

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Dwell Time for Indexed Step](#)
- [Query Dwell Time for Indexed Step](#)
- [Set Dwell Time Units](#)

3.3 (k) - Set Number of Cycles

Description

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

Command Syntax

`:SEQ:CYCLES:[count]`

Variable	Description
<code>[count]</code>	The number of cycles / repetitions of the switch sequence to execute. Set the count to 0 to execute the sequence continuously.

Return String

`[status]`

Variable	Value	Description
<code>[status]</code>	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
<code>:SEQ:CYCLES:5</code>	1

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Query Number of Cycles](#)

3.3 (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. A count of 0 indicates that the sequence will run continuously.

Examples

String to Send	String Returned
:SEQ:CYCLES?	5

See Also

DLL - [Send SCPI Switch Command](#)

Interrupt - [Send SCPI Switch Command](#)

[Set Number of Cycles](#)

3.3 (m) - Set Sequence Direction

Description

Sets which way through the programmed list of steps to run.

Command Syntax

`:SEQ:DIRECTION:[mode]`

Variable	Value	Description
[mode]	0	Forward (from first to last step)
	1	Reverse (from last to first step)
	2	Bi-directional (forward then reverse)

Return String

[status]

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

Examples

String to Send	String Returned
<code>:SEQ:DIRECTION:0</code>	0
<code>:SEQ:DIRECTION:1</code>	1
<code>:SEQ:DIRECTION:1</code>	2

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Sequence Direction](#)
- [Query Sequence Direction](#)

3.3 (n) - Query Sequence Direction

Description

Queries which way the switch will run through the programmed list of steps.

Command Syntax

`:SEQ:DIRECTION?`

Return String

`[mode]`

Variable	Value	Description
<code>[mode]</code>	0	Forward (from first to last step)
	1	Reverse (from last to first step)
	2	Bi-directional (forward then reverse)

Examples

String to Send	String Returned
<code>:SEQ:DIRECTION?</code>	0
<code>:SEQ:DIRECTION?</code>	1
<code>:SEQ:DIRECTION?</code>	2

See Also

- [DLL - Send SCPI Switch Command](#)
- [Interrupt - Send SCPI Switch Command](#)
- [Set Sequence Direction](#)
- [Query Sequence Direction](#)

3.3 (o) - 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:MODE:[mode]`

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

Return String

`[status]`

Variable	Value	Description
<code>[status]</code>	0	Command failed
	1	Command completed successfully

Examples

String to Send	String Returned
<code>:SEQ:MODE:OFF</code>	1
<code>:SEQ:MODE:ON</code>	1

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/solidstate.html>

4.1 - DLL API Options

4.1 (a) - .NET Framework 4.5 DLL (Recommended)

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

Filename: mcl_SolidStateSwitch_NET45.dll

Requirements

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

Installation

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

4.1 (b) - .NET Framework 2.0 DLL (Legacy Support)

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

Filename: mcl_SolidStateSwitch64.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 (c) - ActiveX COM Object DLL (Legacy Support)

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

Filename: `mcl_SolidStateSwitch.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 (`MCL_SolidStateSwitch.dll`) 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_SolidStateSwitch.dll`
 - b. 64-bit PC (be sure to specify the path for `regsvr32` and the DLL):
`\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\MCL_SolidStateSwitch.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

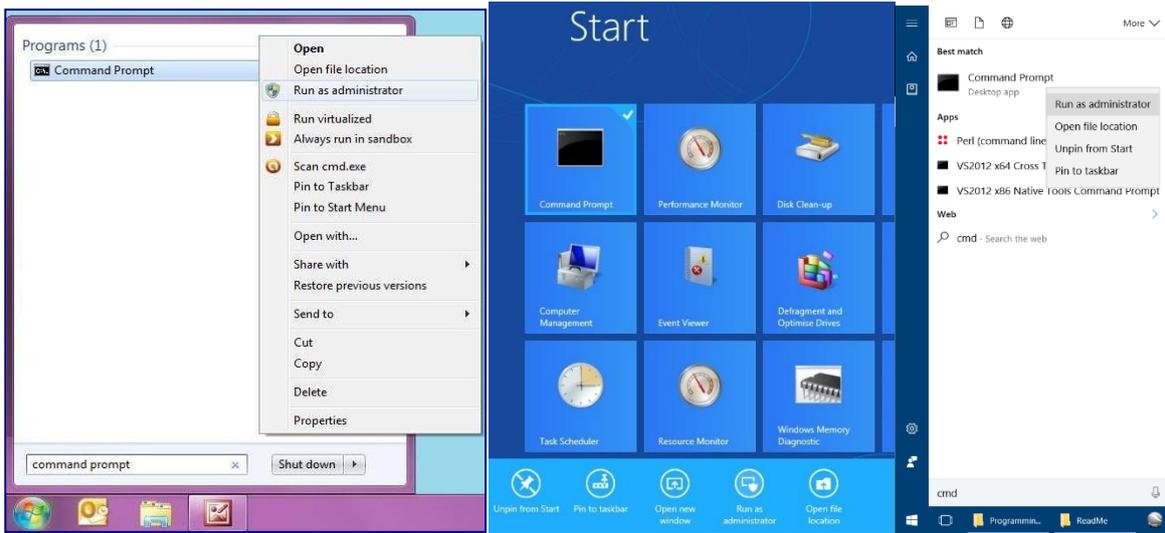


Fig 4.1-b: Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)

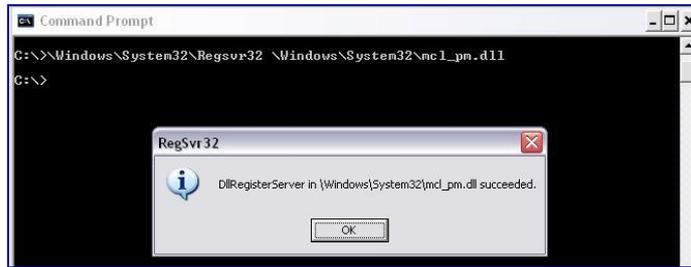


Fig 4.1-c: Registering the DLL in a 32-bit environment

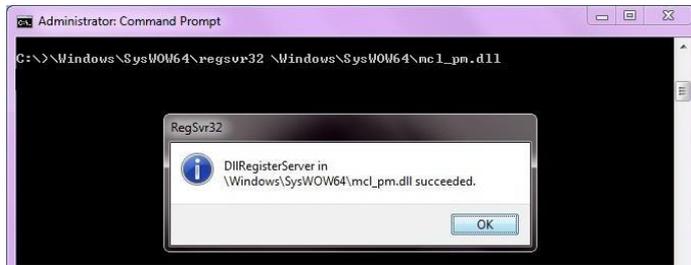


Fig 4.1-d: Registering the DLL in a 64-bit environment

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 4.5 DLL (mcl_SolidStateSwitch_NET45.dll)

(For operation with the .Net 2.0 DLL, replace "mcl_SolidStateSwitch_NET45" with "mcl_SolidStateSwitch64")

Python	<pre>import clr # Import the pythonnet CLR library clr.AddReference('C:\Windows\SysWOW64\mcl_SolidStateSwitch_NET45.dll') from mcl_SolidStateSwitch_NET45 import USB_Digital_Switch MyPTE1 = USB_Digital_Switch() MyPTE2 = USB_Digital_Switch()</pre>
Visual Basic	<pre>Public MyPTE1 As New mcl_SolidStateSwitch_NET45.USB_Digital_Switch Public MyPTE2 As New mcl_SolidStateSwitch_NET45.USB_Digital_Switch</pre>
Visual C++	<pre>mcl_SolidStateSwitch_NET45::USB_Digital_Switch ^MyPTE1 = gcnew mcl_SolidStateSwitch_NET45::USB_Digital_Switch(); mcl_SolidStateSwitch_NET45::USB_Digital_Switch ^MyPTE2 = gcnew mcl_SolidStateSwitch_NET45::USB_Digital_Switch();</pre>
Visual C#	<pre>public mcl_SolidStateSwitch_NET45.USB_Digital_Switch MyPTE1 = new mcl_SolidStateSwitch_NET45.USB_Digital_Switch(); public mcl_SolidStateSwitch_NET45.USB_Digital_Switch MyPTE2 = new mcl_SolidStateSwitch_NET45.USB_Digital_Switch();</pre>
MatLab	<pre>MCL_SW = NET.addAssembly('C:\Windows\SysWOW64\mcl_SolidStateSwitch_NET45.dll') MyPTE1 = mcl_SolidStateSwitch_NET45.USB_Digital_Switch MyPTE2 = mcl_SolidStateSwitch_NET45.USB_Digital_Switch</pre>

Example Declarations using the ActiveX DLL (mcl_SolidStateSwitch.dll)

Visual Basic	<pre>Public MyPTE1 As New mcl_SolidStateSwitch.USB_Control Public MyPTE2 As New mcl_SolidStateSwitch.USB_Control</pre>
Visual C++	<pre>mcl_SolidStateSwitch::USB_Control ^MyPTE1 = gcnew mcl_SolidStateSwitch::USB_Control(); mcl_SolidStateSwitch::USB_Control ^MyPTE2 = gcnew mcl_SolidStateSwitch::USB_Control();</pre>
Visual C#	<pre>public mcl_SolidStateSwitch.USB_Control MyPTE1 = new mcl_SolidStateSwitch.USB_Control(); public mcl_SolidStateSwitch.USB_Control MyPTE2 = new mcl_SolidStateSwitch.USB_Control();</pre>
MatLab	<pre>MyPTE1 = actxserver('mcl_SolidStateSwitch.USB_Control') MyPTE2 = actxserver('mcl_SolidStateSwitch.USB_Control')</pre>

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 operation:
 - 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 operation:
 - 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	<code>int Read_SN(ByRef string SN)</code>
Visual C#	<pre>status = MyPTE1.Read_SN(ref(SN)); if(status > 0) { MessageBox.Show("The connected device is " + SN); }</pre>
Python	<pre>status = MyPTE1.Read_SN("") if status[0] > 0: SN = str(status[1]) print('The connected device is ', SN)</pre>
MatLab	<pre>[status, SN] = MyPTE1.Read_SN('') if status > 0 h = msgbox('The connected device is ', SN) end</pre>

4.4 - DLL Function Definitions

The following functions are defined in all DLL file options. Please see the following sections for a full description of their use and implementation.

- a) int `Connect`(Optional ByRef string `SN`)
- b) int `ConnectByAddress`(Optional ByRef int `Address`)
- c) void `Disconnect`()
- d) int `Read_ModelName`(ByRef string `ModelName`)
- e) int `Read_SN`(ByRef string `SN`)
- f) int `Set_SP4T_COM_To`(Byte `Port`)
- g) int `Get_SP4T_State`()
- h) int `Send_SCPI`(ByRef string `SendStr`, ByRef string `RetStr`)
- i) int `Set_Address`(int `Address`)
- j) int `Get_Address`()
- k) int `Get_Available_SN_List`(ByRef string `SN_List`)
- l) int `Get_Available_Address_List`(ByRef string `Add_List`)
- m) int `GetUSBConnectionStatus`()
- n) string `GetUSBDeviceName`()
- o) int `GetExtFirmware`(ByRef int `A0`, ByRef int ByRef `A1`, int ByRef `A2`,
ByRef string `Firmware`)
- p) int `GetFirmware`()

4.5 - DLL Function Explanations

These functions apply to all Mini-Circuits solid state switch models and provide a means to control the device over a USB connection.

4.5 (a) - Connect to Switch

```
int Connect(Optional ByRef string SN)
```

Description

Initializes the USB connection to a switch. If multiple switches are connected to the same computer, then the serial number should be included, otherwise this can be omitted. The switch 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 USB switch. Can be omitted if only one switch is connected.

Return Values

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

Examples

Python	status = MyPTE1.Connect()
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 to Switch by Address](#)
[Disconnect from Switch](#)
[Get List of Connected Serial Numbers](#)

4.5 (b) - Connect to Switch by Address

```
int ConnectByAddress (Optional ByRef int Address)
```

Description

This function is called to initialize the USB connection to a switch by referring to a user defined 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 connection process can take a few milliseconds so it is recommended that the connection be made once at the beginning of the routine and left open until the switch is no longer needed. The switch should be disconnected on completion of the program using the [Disconnect](#) function.

Parameters

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

Return Values

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

Examples

Python	<code>status = MyPTE1.ConnectByAddress(5)</code>
Visual Basic	<code>status = MyPTE1.ConnectByAddress(5)</code>
Visual C++	<code>status = MyPTE1->ConnectByAddress(5);</code>
Visual C#	<code>status = MyPTE1.ConnectByAddress(ref(5));</code>
MatLab	<code>status = MyPTE1.ConnectByAddress(5);</code>

See Also

[Connect to Switch](#)
[Disconnect from Switch](#)
[Set Address of Switch](#)
[Get Address of Switch](#)

4.5 (c) - Disconnect from Switch

```
void Disconnect ()
```

Description

This function is called to close the connection to the switch after completion of the switching routine. 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 switch from the computer, then reconnect the switch before attempting to start again.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
none		

Examples

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

See Also

[Connect to Switch](#)

[Connect to Switch by Address](#)

4.5 (d) - Read Model Name of Switch

```
int Read_ModelName (ByRef string ModelName)
```

Description

This function is called to determine the Mini-Circuits part number of the connected switch. The user passes a string variable which is updated with the part number.

Parameters

Data Type	Variable	Description
string	ModelName	String variable, passed by reference, to be updated with the Mini-Circuits part number for the switch.

Return Values

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

Examples

Python	<pre>status = MyPTE1.Read_ModelName("") if status[0] > 0: ModelName = str(status[1]) print('The connected device is ', ModelName)</pre>
Visual Basic	<pre>If MyPTE1.Read_ModelName(ModelName) > 0 Then MsgBox ("The connected device is " & ModelName) End If</pre>
Visual C++	<pre>if (MyPTE1->Read_ModelName(ModelName) > 0) { MessageBox::Show("The connected device is " + ModelName); }</pre>
Visual C#	<pre>if (MyPTE1.Read_ModelName(ref(ModelName)) > 0) { MessageBox.Show("The connected device is " + ModelName); }</pre>
MatLab	<pre>[status, ModelName] = MyPTE1.Read_ModelName('') if status > 0 h = msgbox('The connected device is ', ModelName) end</pre>

See Also

[Read Serial Number of Switch](#)

4.5 (e) - Read Serial Number of Switch

```
int Read_SN (ByRef string SN)
```

Description

This function is called to determine the serial number of the connected switch. The user passes a string variable which is updated with the serial number.

Parameters

Data Type	Variable	Description
string	ModelName	String variable, passed by reference, to be updated with the Mini-Circuits serial number for the switch.

Return Values

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

Examples

Python	<pre>status = MyPTE1.Read_SN("") if status[0] > 0: SN = str(status[1]) print('The connected device is ', SN)</pre>
Visual Basic	<pre>If MyPTE1.Read_SN(SN) > 0 Then MsgBox ("The connected device is " & SN) End If</pre>
Visual C++	<pre>if (MyPTE1->Read_SN(SN) > 0) { MessageBox::Show("The connected device is " + SN); }</pre>
Visual C#	<pre>if (MyPTE1.Read_SN(ref(SN)) > 0) { MessageBox.Show("The connected device is " + SN); }</pre>
MatLab	<pre>[status, SN] = MyPTE1.Read_SN('') if status > 0 h = msgbox('The connected device is ', SN) end</pre>

See Also

[Connect to Switch](#)

[Read Model Name of Switch](#)

4.5 (f) - Set SP4T Switch

```
int Set_SP4T_COM_To(Byte Port)
```

This function is not applicable to this switch series. See [Send SCPI Switch Command](#) instead.

4.5 (g) - Get SP4T Switch State

```
int Get_SP4T_State ()
```

This function is not applicable to this switch series. See [Send SCPI Switch Command](#) instead.

4.5 (h) - Send SCPI Switch Command

```
int Send_SCPI(ByRef String SendStr, ByRef String RetStr)
```

Description

Sends a SCPI command to set or read the switch state and other properties. Refer to [Summary of SCPI Commands / Queries](#) for the syntax of the SCPI commands.

Parameters

Data Type	Variable	Description
String	SendStr	The SCPI command / query to send to the switch module
String	RetStr	Passed by reference in order to be updated with the response from the switch module

Return Values

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

Examples

Python	<pre>status = MyPTE1.Send_SCPI(":SP2T:A:STATE:2", "") response = str(status[1])</pre>
Visual Basic	<pre>status = MyPTE1.Send_SCPI(":SP2T:A:STATE:2", response)</pre>
Visual C++	<pre>status = MyPTE1->Send_SCPI(":SP2T:A:STATE:2", response);</pre>
Visual C#	<pre>status = MyPTE1.Send_SCPI(ref(":SP2T:A:STATE:2"), ref(response));</pre>
MatLab	<pre>[status, command, response] = MyPTE1.Send_SCPI(":SP2T:A:STATE:2", response)</pre>

See Also

[Summary of SCPI Commands / Queries](#)

4.5 (i) - Set Address of Switch

```
int Set_Address (int Address)
```

Description

This function allows the internal address of the connected switch to be changed from the factory default of 255. The switch can be referred to by the address instead of the serial number (see [Connect to Switch by Address](#)).

Parameters

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

Return Values

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

Example

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

See Also

[Connect to Switch by Address](#)
[Get Address of Switch](#)
[Get List of Available Addresses](#)

4.5 (j) - Get Address of Switch

```
int Get_Address ()
```

Description

This function returns the address of the connected device.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	0	Command failed
int	1-255	Address of the switch

Examples

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

See Also

[Connect to Switch by Address](#)
[Set Address of Switch](#)
[Get List of Available Addresses](#)

4.5 (k) - Get List of Connected Serial Numbers

```
int Get_Available_SN_List(ByRef string SN_List)
```

Description

This function takes a user defined variable and updates it with a list of serial numbers for all available (currently connected) switches.

Parameters

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

Return Values

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

Example

Python	<pre>status = MyPTE1.Get_Available_SN_List("") if status[0] > 0: SN_List = str(status[1]) print("Connected devices:", SN_List)</pre>
Visual Basic	<pre>If MyPTE1.Get_Available_SN_List(SN_List) > 0 Then MsgBox ("Connected devices: " & SN_List) End If</pre>
Visual C++	<pre>if (MyPTE1->Get_Available_SN_List(SN_List) > 0) { MessageBox::Show("Connected devices: " + SN_List); }</pre>
Visual C#	<pre>if (MyPTE1.Get_Available_SN_List(ref(SN_List)) > 0) { MessageBox.Show("Connected devices: " + SN_List); }</pre>
MatLab	<pre>[status, SN_List] = MyPTE1.Get_Available_SN_List('') if status > 0 h = msgbox('Connected devices: ', SN_List) end</pre>

See Also

[Connect to Switch](#)
[Get List of Available Addresses](#)

4.5 (I) - Get List of Available Addresses

```
int Get_Available_Address_List (ByRef string Add_List)
```

Description

This function takes a user defined variable and updates it with a list of addresses of all connected switches.

Parameters

Data Type	Variable	Description
string	Add_List	String variable passed by reference to be updated with a list of addresses separated by a single space character, for example, "5 101 254 255"

Return Values

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

Example

Python	<pre>status = MyPTE1.Get_Available_Add_List("") if status[0] > 0: Address_List = str(status[1]) print("Connected devices:", Address_List)</pre>
Visual Basic	<pre>If MyPTE1.Get_Available_Add_List(SN_List) > 0 Then MsgBox ("Connected devices: " & Address_List) End If</pre>
Visual C++	<pre>if (MyPTE1->Get_Available_Add_List(Address_List) > 0) { MessageBox::Show("Connected devices: " + Address_List); }</pre>
Visual C#	<pre>if (MyPTE1.Get_Available_Add_List(ref(Address_List)) > 0) { MessageBox.Show("Connected devices: " + Address_List); }</pre>
MatLab	<pre>[status, Address_List] = MyPTE1.Get_Available_Add_List('') if status > 0 h = msgbox('Connected devices: ', Address_List) end</pre>

See Also

[Connect to Switch by Address](#)

[Get List of Connected Serial Numbers](#)

4.5 (m) - Get USB Connection Status

```
int GetUSBConnectionStatus ()
```

Description

This function checks whether the USB connection to the switch is still active.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	0	No connection
int	1	USB connection to switch is active

Examples

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

4.5 (n) - Get USB Device Name

```
string GetUSBDeviceName ()
```

Description

This function is for advanced users wishing to identify the device name of the switch for direct USB communication.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
string	Name	Device name of the switch matrix

Examples

Python	<code>status = MyPTE1.GetUSBDeviceName()</code>
Visual Basic	<code>status = MyPTE1.GetUSBDeviceName()</code>
Visual C++	<code>status = MyPTE1->GetUSBDeviceName();</code>
Visual C#	<code>status = MyPTE1.GetUSBDeviceName();</code>
MatLab	<code>status = MyPTE1.GetUSBDeviceName();</code>

4.5 (o) - Get Firmware

```
int GetExtFirmware(ByRef int A0, ByRef int A1, ByRef int A2,
                  ByRef string Firmware)
```

Description

This function returns the internal firmware version of the switch along with three reserved variables (for factory use).

Parameters

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

Return Values

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

Examples

Python	<pre>status = MyPTE1.GetExtFirmware(A0, A1, A2, Firmware) if status[0] > 0: Firmware = str(status[4]) print("Firmware Version:", Firmware)</pre>
Visual Basic	<pre>If MyPTE1.GetExtFirmware(A0, A1, A2, Firmware) > 0 Then MsgBox ("Firmware Version: " & Firmware) End If</pre>
Visual C++	<pre>if (MyPTE1->GetExtFirmware(A0, A1, A2, Firmware) > 0) { MessageBox::Show("Firmware Version: " + Firmware); }</pre>
Visual C#	<pre>if(MyPTE1.GetExtFirmware(ref(A0),ref(A1),ref(A1), ref(Firmware)) > 0) { MessageBox.Show("Firmware Version: " + Firmware); }</pre>
MatLab	<pre>[status,A0, A1, A2, Firmware] = MyPTE1.GetExtFirmware('', '', '', '') if status > 0 h = msgbox('Firmware Version: ', Firmware) end</pre>

4.5 (p) - Get Firmware Version (Antiquated)

```
string GetFirmware()
```

This function is obsolete. See [Get Firmware](#) instead.

4.6 - Switching Sequence DLL Functions

Mini-Circuits' high isolation solid-state switches have a "switching sequence mode" which allows the user to program a timed sequence of switch states into the switch's internal microcontroller, allowing very fast switching sequences to be triggered with no further USB communication.

4.6 (a) - Set Number of Steps

```
int SetSequence_NoOfSteps (int NoOfSteps)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Sets the number of steps to be configured for the pre-defined switching sequence.

Parameters

Data Type	Variable	Description
int	NoOfSteps	Number of steps to configure (1 to 100)

Return Values

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

Examples

Configure 5 steps in the switching sequence:

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

See Also

[Get Number of Steps](#)

4.6 (b) - Get Number of Steps

```
int GetSequence_NoOfSteps ()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the number of steps in the switching sequence.

Return Values

Data Type	Value	Description
int	NoOfSteps	Number of steps in the switching sequence (1 to 100)

Examples

Python	<code>result = MyPTE1.GetSequence_NoOfSteps()</code>
Visual Basic	<code>result = MyPTE1.GetSequence_NoOfSteps()</code>
Visual C++	<code>result = MyPTE1->GetSequence_NoOfSteps();</code>
Visual C#	<code>result = MyPTE1.GetSequence_NoOfSteps();</code>
MatLab	<code>result = MyPTE1.GetSequence_NoOfSteps();</code>

See Also

[Set Number of Steps](#)

4.6 (c) - Set Step

Declaration

```
int SetSequence_Step(int StepNo, int SwitchTo, int Dwell, int DwellUnits)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Configures the state and dwell time of a single step within the pre-defined switching sequence.

Parameters

Data Type	Variable	Description
int	StepNo	Step index number, indexed from 0 to n - 1
int	SwitchTo	Switch state expressed as an integer, 0 to 16 (model dependent, the port which is to be connected to the Com port)
int	Dwell	Dwell time
int	DwellUnits	Dwell time units: 0 = microseconds (μ s) 1 = milliseconds (ms) 2 = seconds (s)

Return Values

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

Examples

Set step 3 of 5 in the sequence (index number 2) so that Com will be connected to port 3, with a dwell time of 5 μ s:

Python	<code>status = MyPTE1.SetSequence_Step(2, 3, 5, 0)</code>
Visual Basic	<code>status = MyPTE1.SetSequence_Step(2, 3, 5, 0)</code>
Visual C++	<code>status = MyPTE1->SetSequence_Step(2, 3, 5, 0);</code>
Visual C#	<code>status = MyPTE1.SetSequence_Step(2, 3, 5, 0);</code>
MatLab	<code>status = MyPTE1.SetSequence_Step(2, 3, 5, 0);</code>

See Also

- [Set Number of Steps](#)
- [Get Step Switch State](#)
- [Get Step Dwell Time](#)
- [Get Step Dwell Time Units](#)

4.6 (d) - Get Step Switch State

Declaration

```
int GetSequence_SwitchTo(int StepNo)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the switch state for a single step within the pre-defined switching sequence.

Parameters

Data Type	Variable	Description
int	StepNo	Step index number, indexed from 0 to n - 1

Return Values

Data Type	Value	Description
int	State	The switch state for the specified step in the sequence

Examples

Python	result = MyPTE1.GetSequence_SwitchTo(2)
Visual Basic	result = MyPTE1.GetSequence_SwitchTo(2)
Visual C++	result = MyPTE1->GetSequence_SwitchTo(2);
Visual C#	result = MyPTE1.GetSequence_SwitchTo(2);
MatLab	result = MyPTE1.GetSequence_SwitchTo(2);

See Also

[Get Number of Steps](#)
[Set Step](#)
[Get Step Dwell Time](#)
[Get Step Dwell Time Units](#)

4.6 (e) - Get Step Dwell Time

Declaration

```
int GetSequence_Dwell (int StepNo)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the dwell time (without units) for a single step within the pre-defined switching sequence.

Parameters

Data Type	Variable	Description
int	StepNo	Step index number, indexed from 0 to n - 1

Return Values

Data Type	Value	Description
int	Dwell	The dwell time (without units) for the specified step in the sequence

Examples

Python	result = MyPTE1.GetSequence_Dwell(2)
Visual Basic	result = MyPTE1.GetSequence_Dwell(2)
Visual C++	result = MyPTE1->GetSequence_Dwell(2);
Visual C#	result = MyPTE1.GetSequence_Dwell(2);
MatLab	result = MyPTE1.GetSequence_Dwell(2);

See Also

- [Get Number of Steps](#)
- [Set Step](#)
- [Get Step Switch State](#)
- [Get Step Dwell Time Units](#)

4.6 (f) - Get Step Dwell Time Units

Declaration

```
int GetSequence_DwellUnits (int StepNo)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the dwell time units for a single step within the pre-defined switching sequence.

Parameters

Data Type	Variable	Description
int	StepNo	Step index number, indexed from 0 to n - 1

Return Values

Data Type	Value	Description
int	DwellUnits	The dwell time units for the specified step in the sequence

Examples

Python	<code>result = MyPTE1.GetSequence_DwellUnits(2)</code>
Visual Basic	<code>result = MyPTE1.GetSequence_DwellUnits(2)</code>
Visual C++	<code>result = MyPTE1->GetSequence_DwellUnits(2);</code>
Visual C#	<code>result = MyPTE1.GetSequence_DwellUnits(2);</code>
MatLab	<code>result = MyPTE1.GetSequence_DwellUnits(2);</code>

See Also

[Get Number of Steps](#)
[Set Step](#)
[Get Step Switch State](#)
[Get Step Dwell Time](#)

4.6 (g) - Set Sequence Direction

Declaration

```
int SetSequence_Direction(int Direction)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Sets the direction in which the sequence of switch states which will be executed.

Parameters

Data Type	Variable	Description
int	Direction	Direction in which execute the sequence of switch states: 0 = Forward - ascending order from index point 0 to (n - 1) 1 = Reverse - descending order index point (n - 1) to 0 2 = Bi-Directional - ascending then descending order

Return Values

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

Examples

Configure the sequence to execute in the forward direction:

Python	status = MyPTE1.SetSequence_Direction(0)
Visual Basic	status = MyPTE1.SetSequence_Direction(0)
Visual C++	status = MyPTE1->SetSequence_Direction(0);
Visual C#	status = MyPTE1.SetSequence_Direction(0);
MatLab	status = MyPTE1.SetSequence_Direction(0);

See Also

[Get Sequence Direction](#)

4.6 (h) - Get Sequence Direction

Declaration

```
int GetSequence_Direction ()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the direction in which the sequence of switch states will be executed.

Return Values

Data Type	Value	Description
int	Direction	Direction in which the sequence of switch states will be executed: 0 = Forward - ascending order from index point 0 to (n - 1) 1 = Reverse - descending order index point (n - 1) to 0 2 = Bi-Directional - ascending then descending order

Examples

Python	result = MyPTE1.GetSequence_Direction()
Visual Basic	result = MyPTE1.GetSequence_Direction()
Visual C++	result = MyPTE1->GetSequence_Direction();
Visual C#	result = MyPTE1.GetSequence_Direction();
MatLab	result = MyPTE1.GetSequence_Direction();

See Also

[Set Sequence Direction](#)

4.6 (i) - Set Number of Cycles

Declaration

```
int SetSequence_NoOfCycles (int NoOfCycles)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Sets the number of times that the complete switching sequence will be executed. This setting will be ignored if the switch sequence has been configured to execute in continuous mode.

Parameters

Data Type	Variable	Description
int	NoOfSteps	Number of cycles, from 1 to 65,535

Return Values

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

Examples

Configure the switching sequence to be executed 400 times

Python	<code>status = MyPTE1.SetSequence_NoOfCycles(400)</code>
Visual Basic	<code>status = MyPTE1.SetSequence_NoOfCycles(400)</code>
Visual C++	<code>status = MyPTE1->SetSequence_NoOfCycles(400);</code>
Visual C#	<code>status = MyPTE1.SetSequence_NoOfCycles(400);</code>
MatLab	<code>status = MyPTE1.SetSequence_NoOfCycles(400);</code>

See Also

[Set Number of Cycles](#)
[Enable / Disable Continuous Mode](#)
[Check Continuous Mode State](#)

4.6 (j) - Get Number of Cycles

Declaration

```
int GetSequence_NoOfCycles ()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Returns the number of times that the complete switching sequence will be executed. This setting will be ignored if the switch sequence has been configured to execute in continuous mode.

Return Values

Data Type	Value	Description
int	NoOfCycles	Number of cycles, from 1 to 65,535

Examples

Python	result = MyPTE1.GetSequence_NoOfCycles()
Visual Basic	result = MyPTE1.GetSequence_NoOfCycles()
Visual C++	result = MyPTE1->GetSequence_NoOfCycles();
Visual C#	result = MyPTE1.GetSequence_NoOfCycles();
MatLab	result = MyPTE1.GetSequence_NoOfCycles();

See Also

[Set Number of Cycles](#)
[Enable / Disable Continuous Mode](#)
[Check Continuous Mode State](#)

4.6 (k) - Enable / Disable Continuous Mode

Declaration

```
int SetSequence_ContinuousMode (int Mode)
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Configures whether the switch sequence will be executed continuously or for a pre-defined number of cycles. With continuous mode enabled, the sequence will repeat from the time the sequence is enabled by the user until the time it is disabled by the user; the setting for number of cycles will be ignored. With continuous mode disabled the sequence will be repeat according to the setting for number of cycles.

Parameters

Data Type	Variable	Description
int	Mode	Setting for continuous mode: 0 = Continuous mode disabled 1 = Continuous mode enabled

Return Values

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

Examples

Configure the sequence to execute in continuous mode:

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

See Also

- [Set Number of Cycles](#)
- [Get Number of Cycles](#)
- [Check Continuous Mode State](#)

4.6 (I) - Check Continuous Mode State

Declaration

```
int GetSequence_ContinuousMode ()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Indicates whether or not the switching sequence is configured to operate in continuous mode. With continuous mode enabled, the sequence will repeat from the time the sequence is enabled by the user until the time it is disabled by the user; the setting for number of cycles will be ignored. With continuous mode disabled the sequence will repeat according to the setting for number of cycles.

Return Values

Data Type	Value	Description
int	Mode	Setting for continuous mode: 0 = Continuous mode disabled 1 = Continuous mode enabled

Examples

Python	result = MyPTE1.GetSequence_ContinuousMode()
Visual Basic	result = MyPTE1.GetSequence_ContinuousMode()
Visual C++	result = MyPTE1->GetSequence_ContinuousMode();
Visual C#	result = MyPTE1.GetSequence_ContinuousMode();
MatLab	result = MyPTE1.GetSequence_ContinuousMode();

See Also

[Set Number of Cycles](#)
[Get Number of Cycles](#)
[Enable / Disable Continuous Mode](#)

4.6 (m) - Start Sequence

Declaration

```
int SetSequence_ON()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Starts the pre-defined switching sequence. The sequence will not operate unless all required parameters have been configured correctly.

Return Values

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

Examples

Start the switching sequence:

Python	<code>status = MyPTE1.SetSequence_ON()</code>
Visual Basic	<code>status = MyPTE1.SetSequence_ON()</code>
Visual C++	<code>status = MyPTE1->SetSequence_ON();</code>
Visual C#	<code>status = MyPTE1.SetSequence_ON();</code>
MatLab	<code>status = MyPTE1.SetSequence_ON();</code>

See Also

[Stop Sequence](#)

4.6 (n) - Stop Sequence

Declaration

```
int SetSequence_OFF()
```

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Description

Stops the pre-defined switching sequence. The sequence will not operate unless all required parameters have been configured correctly.

Return Values

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

Examples

Start the switching sequence:

Python	<code>status = MyPTE1.SetSequence_OFF()</code>
Visual Basic	<code>status = MyPTE1.SetSequence_OFF()</code>
Visual C++	<code>status = MyPTE1->SetSequence_OFF();</code>
Visual C#	<code>status = MyPTE1.SetSequence_OFF();</code>
MatLab	<code>status = MyPTE1.SetSequence_OFF();</code>

See Also

[Start Sequence](#)

5 - USB Control via Direct Programming (Linux)

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

5.1 - USB Interrupt Code Concept

To open a USB connection to the system, the Vendor ID and Product ID are required:

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

The transmitted and received buffer sizes for the interrupt commands 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 - Summary of Commands

The commands that can be sent to the switch are summarized in the table below and detailed on the following pages.

5.2 (a) - Core Commands

	Description	Command Code (Byte 0)	Comments
a	Get Device Model Name	40	
b	Get Device Serial Number	41	
c	Send SCPI Switch Command	42	
d	Get Firmware	99	

5.2 (b) - Switching Sequence Commands

These functions allow a pre-defined switching sequence to be programmed into the switch.

	Description	Byte 0	Byte 1
a	Set Number of Steps	204	0
b	Get Number of Steps	205	0
c	Set Step	204	1
d	Get Step	205	1
e	Set Direction	204	2
f	Get Direction	205	2
g	Set Number of Cycles	204	4
h	Get Number of Cycles	205	4
i	Enable / Disable Continuous Mode	204	3
j	Check Continuous Mode State	205	3
k	Start / Stop Sequence	204	5

5.3 - Core Commands

5.3 (a) - Get Device Model Name

Description

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

Transmit Array

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

Returned Array

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

Example

The following array would be returned for Mini-Circuits’ USB-2SP2T-63H switch (see the [Programming Examples & Troubleshooting Guide](#) for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	40	Interrupt code for Get Device Model Name
1	85	ASCII character code for U
2	83	ASCII character code for S
3	42	ASCII character code for B
4	45	ASCII character code for -
5	50	ASCII character code for 2
6	83	ASCII character code for S
7	80	ASCII character code for P
8	50	ASCII character code for 2
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	72	ASCII character code for H
14	0	Zero value byte to indicate end of string

See Also

[Get Device Serial Number](#)

5.3 (b) - Get Device Serial Number

Description

Returns the serial number of the connected switch.

Transmit Array

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

Returned Array

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

Example

The following example indicates that the connected switch box has serial number 1130922011 (see the [Programming Examples & Troubleshooting Guide](#) for conversions between decimal, binary and ASCII characters):

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

See Also

[Get Device Model Name](#)

5.3 (d) - Send SCPI Switch Command

Description

Allows a SCPI command to be sent to the models listed below in order to set or read the switch state. Refer to [Summary of SCPI Commands / Queries](#) for the syntax of the SCPI commands.

Transmit Array

Byte	Data	Description
0	42	Interrupt code for Send SCPI Switch 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	42	Interrupt code for Send SCPI Switch 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

The SCPI command to check the state of USB-SP8T-63H is :SP8T:STATE? (see [Set SP8T Switch State](#)).

The ASCII character codes representing the 12 characters in this command should be sent in bytes 1 to 12 of the transmit array as follows (see the [Programming Examples & Troubleshooting Guide](#) for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1	58	ASCII character code for :
2	83	ASCII character code for S
3	80	ASCII character code for P
4	56	ASCII character code for 8
5	84	ASCII character code for T
6	58	ASCII character code for :
7	83	ASCII character code for S
8	84	ASCII character code for T
9	65	ASCII character code for A
10	84	ASCII character code for T
11	69	ASCII character code for E
12	63	ASCII character code for ?

The returned array below would indicate the switch is in state 8 (COM <> port 8):

Byte	Data	Description
0	42	Interrupt code for Send SCPI Command
1	56	ASCII character code for 8
2	0	Zero value byte to indicate end of string

See Also

[Summary of SCPI Commands / Queries](#)

5.3 (e) - Get Firmware

Description

Returns the internal firmware version of the switch box.

Transmit Array

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1- 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1	Reserved	Internal code for factory use only
2	Reserved	Internal code for factory use only
3	Reserved	Internal code for factory use only
4	Reserved	Internal code for factory use only
5	Firmware Letter	ASCII code for the first character in the firmware revision identifier
6	Firmware Number	ASCII code for the second character in the firmware revision identifier
7-63	Not significant	"Don't care" bytes, could be any value

Example

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

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1	55	Internal code for factory use only
2	52	Internal code for factory use only
3	83	Internal code for factory use only
4	87	Internal code for factory use only
5	67	ASCII code for the letter "C"
6	51	ASCII code for the number 3
7-63	Not significant	"Don't care" bytes, could be any value

5.4 - Switching Sequence Commands

Mini-Circuits' high isolation solid-state switches have a "switching sequence mode" which allows the user to program a timed sequence of switch states into the switch's internal microcontroller, allowing very fast switching sequences to be triggered with no further USB communication.

5.4 (a) - Set Number of Steps

Description

Sets the number of steps to be configured for the pre-defined switching sequence.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	0	Command for Number of Steps
2	Steps	Number of steps to configure (1 to 100)
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will set 5 points in the switching sequence:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	0	Command for Number of Steps
2	5	Set 5 steps in the switching sequence

See Also

[Get Number of Steps](#)

5.4 (b) - Get Number of Steps

Description

Returns the number of steps in the switching sequence.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	0	Command for Number of Steps
2 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	Steps	Number of steps in the switching sequence
2 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following returned array indicates that there are 5 steps in the pre-defined switching sequence:

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	5	5 steps in the switching sequence

See Also

[Set Number of Steps](#)

5.4 (c) - Set Step

Description

Configures the state and dwell time of a single step within the pre-defined switching sequence.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	1	Command for Step Configuration
2	Step_Index	Step index number, indexed from 0 to n - 1
3	Switch_State	Switch state expressed as an integer, 1 to 4 (the port which is to be connected to the Com port)
4	Dwell_0	Dwell time split into 2 bytes: Dwell_0 = INT (Dwell_Time / 256)
5	Dwell_1	Dwell time split into 2 bytes: Dwell_1 = Dwell_Time - (Dwell_0 * 256)
6	Dwell_Units	Dwell time units: 0 = microseconds (μ s) 1 = milliseconds (ms) 2 = seconds (s)
7 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will set step 3 of 5 in the sequence (index number 2) so that Com will be connected to port 3, with a dwell time of 5 μ s:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	1	Command for Step Configuration
2	2	Step index number 2 for the third step in the sequence
3	3	Connect Com to port 3
4	0	Dwell_0 = INT (5 / 256) = 0
5	5	Dwell_1 = 5 - (0 * 256) = 5
6	0	Dwell time units are microseconds (μ s)

See Also

[Get Step](#)

5.4 (d) - Get Step

Description

Returns the state and dwell time of a single step within the pre-defined switching sequence.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	1	Command for Step Configuration
2	Step_Index	Step index number, indexed from 0 to n - 1
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Command for Set Switch Sequence Property
1	Step_Index	Step index number, indexed from 0 to n - 1
2	Switch_State	Switch state expressed as an integer, 1 to 4 (the port which is to be connected to the Com port)
3	Dwell_0	Dwell time split into 2 bytes: $Dwell_Time = (256 * Dwell_0) + Dwell_1$
4	Dwell_1	Dwell time split into 2 bytes
5	Dwell_Units	Dwell time units: 0 = microseconds (μ s) 1 = milliseconds (ms) 2 = seconds (s)
6 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following returned array indicates step 3 of 5 in the sequence (index number 2) is configured so that Com will be connected to port 3, with a dwell time of 5 μ s:

Byte	Data	Description
0	205	Command for Set Switch Sequence Property
1	2	Step index number 2 for the third step in the sequence
2	3	Connect Com to port 3
3	0	Dwell_Time = (256 * 0) + 5 = 5
4	5	Dwell_Time (calculated above)
5	0	Dwell time units are microseconds (μ s)

See Also

[Set Step](#)

5.4 (e) - Set Direction

Description

Sets the direction in which the sequence of switch states which will be executed.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	2	Command for Direction
2	Direction	Direction in which execute the sequence of switch states: 0 = Forward - ascending order from index point 0 to (n - 1) 1 = Reverse - descending order index point (n - 1) to 0 2 = Bi-Directional - ascending then descending order
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will configure the sequence to execute in the forward direction:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	2	Command for Direction
2	0	Set the forward direction

See Also

[Get Direction](#)

5.4 (f) - Get Direction

Description

Returns the direction in which the sequence of switch states will be executed.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	2	Command for Direction
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	Direction	Direction in which the sequence of switch states will be executed: 0 = Forward - ascending order from index point 0 to (n - 1) 1 = Reverse - descending order index point (n - 1) to 0 2 = Bi-Directional - ascending then descending order
2 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following returned array indicates that the sequence will be executed in the forward direction:

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	0	Sequence will execute in the forward direction

See Also

[Set Direction](#)

5.4 (g) - Set Number of Cycles

Description

Sets the number of times that the complete switching sequence will be executed. This setting will be ignored if the switch sequence has been configured to execute in continuous mode.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	4	Command for Number of Cycles
2	Cycles_0	Number of cycles (from 1 to 65,535) split into 2 bytes: Cycles_0 = INT (Cycles / 256)
3	Cycles_1	Number of cycles (from 1 to 65,535) split into 2 bytes: Cycles_1 = Cycles - (Cycles_0 * 256)
4 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will configure the switching sequence to be executed 400 times:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	4	Command for Step Configuration
2	1	Cycles_0 = INT (400 / 256) = 1
3	144	Cycles_1 = 400 - (1 * 256) = 144

See Also

[Get Number of Cycles](#)
[Enable / Disable Continuous Mode](#)
[Check Continuous Mode State](#)
[Start / Stop Sequence](#)

5.4 (h) - Get Number of Cycles

Description

Returns the number of times that the complete switching sequence will be executed. This setting will be ignored if the switch sequence has been configured to execute in continuous mode.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	4	Command for Number of Cycles
2 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	Cycles_0	Number of cycles (from 1 to 65,535) split into 2 bytes: Cycles = (256 * Cycles_0) + Cycles_1
2	Cycles_1	Number of cycles (from 1 to 65,535) split into 2 bytes
3 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following returned array indicates that the switching sequence has been configured to execute 400 times:

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	1	Number of cycles split into 2 bytes: Cycles = (256 * 1) + 144 = 400
2	144	Number of cycles split into 2 bytes (calculated above)

See Also

[Set Number of Cycles](#)
[Enable / Disable Continuous Mode](#)
[Check Continuous Mode State](#)
[Start / Stop Sequence](#)

5.4 (i) - Enable / Disable Continuous Mode

Description

Configures whether the switch sequence will be executed continuously or for a pre-defined number of cycles. With continuous mode enabled, the sequence will repeat from the time the sequence is enabled by the user until the time it is disabled by the user; the setting for number of cycles will be ignored. With continuous mode disabled the sequence will repeat according to the setting for number of cycles.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	3	Command for Continuous Mode
2	Mode	Enable / disable continuous mode: 0 = Continuous mode disabled 1 = Continuous mode enabled
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will configure the sequence to execute in continuous mode:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	3	Command for Direction
2	1	Operate in continuous mode

See Also

[Set Number of Cycles](#)
[Get Number of Cycles](#)
[Check Continuous Mode State](#)
[Start / Stop Sequence](#)

5.4 (j) - Check Continuous Mode State

Description

Indicates whether or not the switching sequence is configured to operate in continuous mode. With continuous mode enabled, the sequence will repeat from the time the sequence is enabled by the user until the time it is disabled by the user; the setting for number of cycles will be ignored. With continuous mode disabled the sequence will repeat according to the setting for number of cycles.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	3	Command for Continuous Mode
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	Mode	Setting for continuous mode: 0 = Continuous mode disabled 1 = Continuous mode enabled
2 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following returned array indicates that the sequence has been configured to operate in continuous mode:

Byte	Data	Description
0	205	Command for Get Switch Sequence Property
1	1	Sequence will operate in continuous mode

See Also

[Set Number of Cycles](#)
[Get Number of Cycles](#)
[Enable / Disable Continuous Mode](#)
[Start / Stop Sequence](#)

5.4 (k) - Start / Stop Sequence

Description

Starts or stops the pre-defined switching sequence. The sequence will not operate unless all required parameters have been configured correctly.

Note: Sending any command to the switch whilst the pre-defined sequence is running will cause the sequence to stop.

Applies To

Supported Models	Required Firmware
U2C-1SP2T-63VH / U2C-1SP4T-63H	B9 or later
USB-1SP16T-83H / USB-1SP8T-63H / USB-2SP2T-DCH / USB-2SP4T-63H / USB-4SP2T-63H	A5 or later

Transmit Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	5	Command for Start / Stop Sequence
2	Mode	Start or stop the switching sequence: 0 = Stop the sequence 1 = Start the sequence
3 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

The following transmit array will start the switching sequence according to the pre-defined parameters:

Byte	Data	Description
0	204	Command for Set Switch Sequence Property
1	5	Command for Start / Stop Sequence
2	1	Start the sequence