Test Solutions - Programming Manual

USB-SP4T-63 Solid State Switch

USB-SP4T-63 Solid State SP4T Switch

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

This Programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' USB-SP4T-63 USB controlled solid-state switch.

For instructions on using the supplied GUI program, or connecting the PTE hardware, please see the User Guide at:
https://www.minicircuits.com/app/AN49-009.pdf

Mini-Circuits offers support over a variety of operating systems, programming environments and third party applications. Support for Windows® operating systems is provided through the Microsoft®.NET® and ActiveX® frameworks to allow the user to develop customized control applications. Support for Linux® operating systems is accomplished using the standard libhid and libusb libraries.

Mini-Circuits has experience with a wide variety of environments including (but not limited to):

- Visual Basic®, Visual C#®, Visual C++®
- Delphi®
- Borland C++®
- CVI®
- LabVIEW®
- MATLAB®
- Python®
- Agilent VEE®

The RF switch software package includes a GUI program, ActiveX and .NET DLL files, Linux support, project examples for third party software, and detailed user manuals. The latest package is available for download at:
https://www.minicircuits.com/softwaredownload/solidstate.html

For details on individual models, application notes, GUI installation instructions and user guides please see:
https://www.minicircuits.com/WebStore/PortableTestEquipment.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 - Switch Control Methods

Communication with the family of solid-state switches is accomplished using one of the API DLL files in a Windows environment (see Operating in a Windows Environment via USB) or the USB interrupt API in a Linux environment (see Operating in a Linux Environment via USB). These APIs define a series of functions to communicate with the switches and query various parameters.

2 - Operating in a Windows Environment via USB

2.1 - The DLL (Dynamic Link Library) Concept

The Dynamic Link Library concept is Microsoft's implementation of the shared library concept in the Windows environment.

DLLs provide a mechanism for shared code and data, intended to allow a developer to distribute applications without requiring code to be re-linked or recompiled.

Mini-Circuits' CD package provides DLL Objects designed to allow your own software application to interface with the functions of Mini-Circuits' USB controlled RF switches.

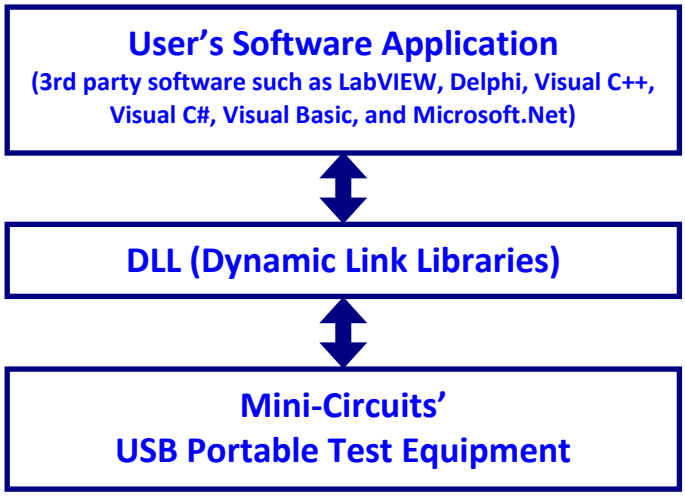

Fig 2.1-a: DLL Interface Concept

The software package provides two DLL files, the choice of which file to use is dictated by the user's operating system:

1. **ActiveX com object**
   Designed to be used in any programming environment that supports third party ActiveX COM (Component Object Model) compliant applications.
   The ActiveX file should be registered using RegSvr32 (see following sections for details).

2. **Microsoft.NET Class Library**
   A logical unit of functionality that runs under the control of the Microsoft.NET system.

2.1 (a) - ActiveX COM Object

ActiveX COM object DLL files are designed to be used with both 32-bit and 64-bit Windows operating systems. A 32-bit programming environment that is compatible with ActiveX is required. To develop 64-bit applications, the Microsoft.NET Class library should be used instead.

Supported Programming Environments

Mini-Circuits' USB controlled RF switches have been tested in the following programming environments. This is not an exhaustive list and the DLL file is designed to operate in most environments that support ActiveX functionality. Please contact Mini-Circuits for support.

- Visual Studio® 6 (Visual C++ and Visual Basic)
- LabVIEW 8.0 or newer
- MATLAB 7 or newer
- Delphi
- Borland C++
- Agilent VEE
- Python

Installation

1. Copy the DLL file (MCL_SolidStateSwitch.dll) to the correct directory:
   For 32-bit Windows operating systems this is C:\WINDOWS\System32
   For 64-bit Windows operating systems this is C:\WINDOWS\SysWOW64
2. Open the Command Prompt:
   a. For Windows XP® (see *Fig 2.1-b*):
      i. Select "All Programs" and then "Accessories" from the Start Menu
      ii. Click on "Command Prompt" to open
   b. For later versions of the Windows operating system you will need to have Administrator privileges in order to run the Command Prompt in "Elevated" mode (see *Fig 2.1-c* for Windows 7 and Windows 8):
      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. You may be prompted to enter the log in details for an Administrator account if the current user does not have Administrator privileges on the local PC
3. Use regsvr32 to register the DLL:
   For 32-bit Windows operating systems type (see Fig 2.1-d):
   `\WINDOWS\System32\Regsvr32 \WINDOWS\System32\MCL_SolidStateSwitch.dll`
   For 64-bit Windows operating systems type (see Fig 2.1-e):
   `\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\MCL_SolidStateSwitch.dll`
4. Hit enter to confirm and a message box will appear to advise of successful registration.

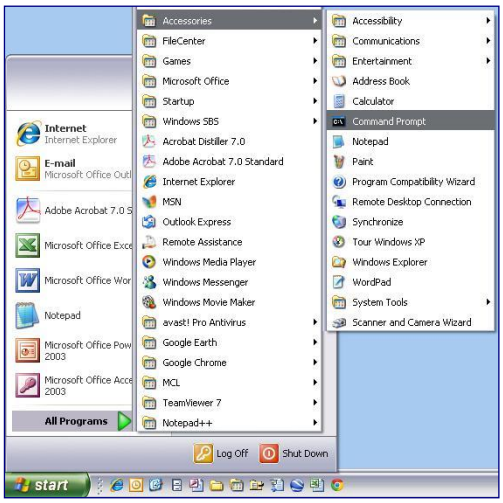

Fig 2.1-b: Opening the Command Prompt in Windows XP

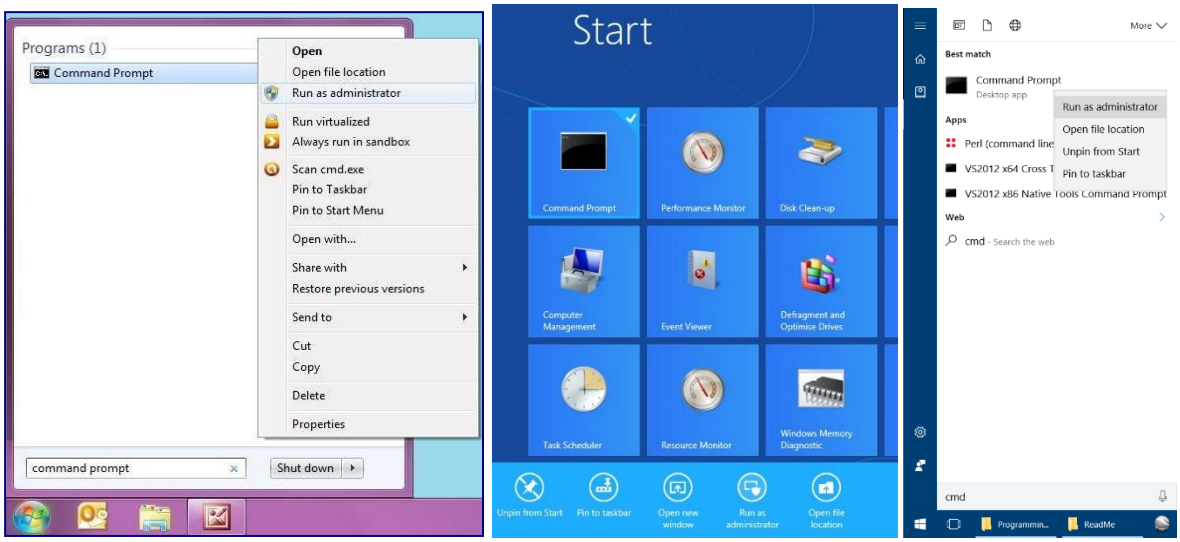

Fig 2.1-c: Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)

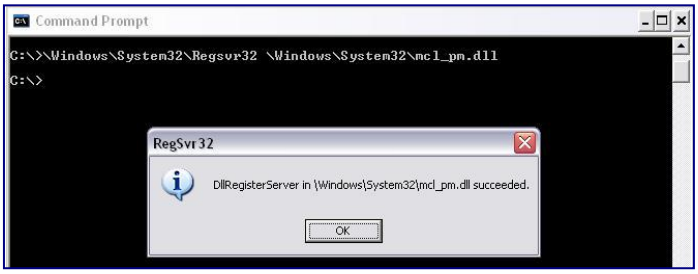

Fig 2.1-d: Registering the DLL in a 32-bit environment

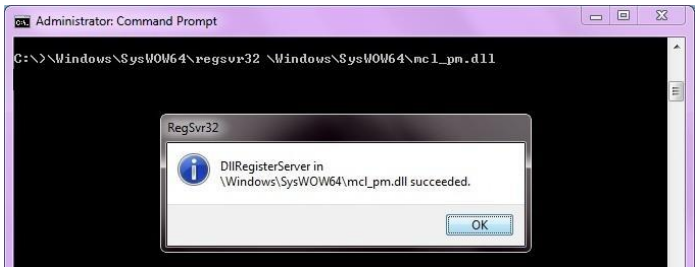

Fig 2.1-e: Registering the DLL in a 64-bit environment

2.1 (b) - Microsoft.NET Class Library

Microsoft.NET class libraries are designed to be used with both 32-bit and 64-bit Windows operating systems. To develop 64-bit applications the user must have both a 64-bit operating system and 64-bit programming environment. However, the Microsoft.NET class library is also compatible with 32-bit programming environments.

Supported Programming Environments

Mini-Circuits' USB controlled RF switches have been tested in the following programming environments. This is not an exhaustive list and the DLL file is designed to operate in most environments that support Microsoft.NET functionality. Please contact Mini-Circuits for support.

- National Instruments CVI
- Microsoft.NET (Visual C++, Visual Basic.NET, Visual C# 2003 or newer)
- LabVIEW 2009 or newer
- MATLAB 2008 or newer
- Delphi
- Borland C++

Installation

1. Copy the DLL file (MCL_SolidStateSwitch64.dll) to the correct directory
   a. For 32 bit Windows operating systems this is `C:\WINDOWS\System32`
   b. For 64 bit Windows operating systems this is `C:\WINDOWS\SysWOW64`
2. **No registration is required**

2.2 - Referencing the DLL Library

In order to use the DLL functionality, some programming environments will require the user to set a reference to the relevant DLL file. Once this is done, the user just needs to declare a new instance of the USB control class (defined within the DLL) for each switch module to be controlled. The class is assigned to a variable which is used to call the DLL functions as needed. In the following examples, the variable names MyPTE1 and MyPTE2 have been used to represent 2 connected switch systems.

Example Declarations using the ActiveX DLL (MCL_SolidStateSwitch.dll)

Visual Basic

```
Public MyPTE1 As New MCL_SolidStateSwitch.USB_Control
        ' Declare new switch object, assign to MyPTE1
Public MyPTE2 As New MCL_SolidStateSwitch.USB_Control
        ' Declare new switch object, assign to MyPTE2
```

Visual C++

```
MCL_SolidStateSwitch::USB_Control ^MyPTE1 = gcnew
                                        _ MCL_SolidStateSwitch::USB_Control();
        // Declare new switch object, assign to MyPTE1
MCL_SolidStateSwitch::USB_Control ^MyPTE2 = gcnew
                                        _ MCL_SolidStateSwitch::USB_Control();
        // Declare new switch object, assign to MyPTE2
```

Visual C#

```
public MCL_SolidStateSwitch.USB_Control MyPTE1 = new
                                        _ MCL_SolidStateSwitch.USB_Control();
        // Declare new switch object, assign to MyPTE1
public MCL_SolidStateSwitch.USB_Control MyPTE2 = new
                                        _ MCL_SolidStateSwitch.USB_Control();
        // Declare new switch object, assign to MyPTE2
```

Matlab

```
MyPTE1 = actxserver('MCL_SolidStateSwitch.USB_Control')
MyPTE2 = actxserver('MCL_SolidStateSwitch.USB_Control')
        % Declare new switch objects, MyPTE1 & MyPTE2
```

Example Declarations using the .NET DLL (MCL_SolidStateSwitch64.dll)

Visual Basic

```
Public MyPTE1 As New MCL_SolidStateSwitch64.USB_Digital_Switch
        ' Declare new switch object, assign to MyPTE1
Public MyPTE2 As New MCL_SolidStateSwitch64.USB_Digital_Switch
        ' Declare new switch object, assign to MyPTE2
```

Visual C++

```
MCL_SolidStateSwitch64::USB_Digital_Switch ^MyPTE1 = gcnew
                                    _ MCL_SolidStateSwitch64::USB_Digital_Switch();
        // Declare new switch object, assign to MyPTE1
MCL_SolidStateSwitch64::USB_Digital_Switch ^MyPTE2 = gcnew
                                    _ MCL_SolidStateSwitch64::USB_Digital_Switch();
        // Declare new switch object, assign to MyPTE2
```

Visual C#

```
public MCL_SolidStateSwitch64.USB_Digital_Switch MyPTE1 = new
                                    _ MCL_SolidStateSwitch64.USB_Digital_Switch();
        // Declare new switch object, assign to MyPTE1
public MCL_SolidStateSwitch64.USB_Digital_Switch MyPTE2 = new
                                    _ MCL_SolidStateSwitch64.USB_Digital_Switch();
        // Declare new switch object, assign to MyPTE2
```

Matlab

```
MCL_SW = NET.addAssembly('C:\Windows\SysWOW64\MCL_SolidStateSwitch64.dll')
MyPTE1 = MCL_SolidStateSwitch64.USB_Digital_Switch
MyPTE2 = MCL_SolidStateSwitch64.USB_Digital_Switch
        % Declare new switch objects, MyPTE1 & MyPTE2
```

2.3 - Summary of DLL Functions

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

2.3 (a) - Core DLL Functions

```
a) int Connect(Optional string SN)
b) int ConnectByAddress(Optional 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(string SN_List)
l) int Get_Available_Address_List(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()
```

2.3 (b) - Switching Sequence DLL Functions

These functions apply to USB-SP4T-63 with firmware version A3 or later:

```
a) int SetSequence_NoOfSteps(int NoOfSteps)
b) int GetSequence_NoOfSteps()
c) int SetSequence_Step(int StepNo, int SwitchTo, int Dwell,
                                                        int DwellUnits)
d) int GetSequence_SwitchTo(ByRef int StepNo)
e) int GetSequence_Dwell(ByRef int StepNo)
f) int GetSequence_DwellUnits(ByRef int StepNo)
g) int SetSequence_Direction(int Direction)
h) int GetSequence_Direction()
i) int SetSequence_NoOfCycles(int NoOfCycles)
j) int GetSequence_NoOfCycles()
k) int SetSequence_ContinuousMode(int ContinuousMode)
l) int GetSequence_ContinuousMode()
m) int SetSequence_ON()
n) int SetSequence_OFF()
```

2.4 - Core DLL Functions

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

2.4 (a) - Connect to Switch

Declaration

```
int Connect(Optional 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

Visual Basic

```
status = MyPTE1.Connect(SN)
```

Visual C++

```
status = MyPTE1->Connect(SN);
```

Visual C#

```
status = MyPTE1.Connect(SN);
```

Matlab

```
status = MyPTE1.Connect(SN)
```

See Also

Connect to Switch by Address
Disconnect from Switch
Get List of Connected Serial Numbers

2.4 (b) - Connect to Switch by Address

Declaration

```
int ConnectByAddress(Optional 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

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 to Switch
Disconnect from Switch
Set Address of Switch
Get Address of Switch

2.4 (c) - Disconnect from Switch

Declaration

```
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

Visual Basic

```
MyPTE1.Disconnect()
```

Visual C++

```
MyPTE1->Disconnect();
```

Visual C#

```
MyPTE1.Disconnect();
```

Matlab

```
MyPTE1.Disconnect
```

See Also

Connect to Switch
Connect to Switch by Address

2.4 (d) - Read Model Name of Switch

Declaration

```
int Read_ModelName(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	Required. A string variable that will 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

Visual Basic

```
If MyPTE1.Read_ModelName(ModelName) > 0 Then
        MsgBox ("The connected switch is " & ModelName)
                ' Display a message stating the model name
End If
```

Visual C++

```
if (MyPTE1->Read_ModelName(ModelName) > 0 )
{
        MessageBox::Show("The connected switch is " + ModelName);
                // Display a message stating the model name
}
```

Visual C#

```
if (MyPTE1.Read_ModelName(ref(ModelName)) > 0 )
{
        MessageBox.Show("The connected switch is " + ModelName);
                // Display a message stating the model name
}
```

Matlab

```
[status, ModelName]=MyPTE1.Read_ModelName(ModelName)
If status > 0 then
{
        msgbox('The connected switch is ', ModelName)
                % Display a message stating the model name
}
```

See Also

Read Serial Number of Switch

2.4 (e) - Read Serial Number of Switch

Declaration

```
int Read_SN(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	Required. string variable that will 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

Visual Basic

```
If MyPTE1.Read_SN(SN) > 0 Then
        MsgBox ("The connected switch is " & SN)
                ' Display a message stating the serial number
End If
```

Visual C++

```
if (MyPTE1->Read_SN(SN) > 0 )
{
        MessageBox::Show("The connected switch is " + SN);
                // Display a message stating the serial number
}
```

Visual C#

```
if (MyPTE1.Read_SN(ref(SN)) > 0 )
{
        MessageBox.Show("The connected switch is " + SN);
                // Display a message stating the serial number
}
```

Matlab

```
[status, SN]= MyPTE1.Read_SN(SN)
If status > 0 then
{
        msgbox('The connected switch is ', SN)
                % Display a message stating the serial number
}
```

See Also

Connect to Switch
Read Model Name of Switch

2.4 (f) - Set SP4T Switch

Declaration

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

Description

Sets the state of the SP4T switch, connecting the Com (common) port to one of ports 1, 2, 3 or 4.

Parameters

Data Type	Variable	Description
Byte	Port	Required. Byte value corresponding to the SP4T switch connection to be made. The 4 options for are: 1 = Com connected to port 1 2 = Com connected to port 2 3 = Com connected to port 3 4 = Com connected to port 4

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Set_SP4T_COM_To(3)
        ' connect COM to port 3
```

Visual C++

```
Status = MyPTE1->Set_SP4T_COM_To(3);
        // connect COM to port 3
```

Visual C#

```
Status = MyPTE1.Set_SP4T_COM_To(3);
        // connect COM to port 3
```

Matlab

```
MyPTE1.Set_SP4T_COM_To(3)
        % connect COM to port 3
```

See Also

Get SP4T Switch State

2.4 (g) - Get SP4T Switch State

Declaration

```
int Get_SP4T_State()
```

Description

Returns the state of an SP4T switch.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	0	Command failed
	1	Switch has Com port connected to port 1
	2	Switch has Com port connected to port 2
	3	Switch has Com port connected to port 3
	4	Switch has Com port connected to port 4

Examples

Visual Basic

```
SwState = MyPTE1.Get_SP4T_State()
If SwState > 0 Then
        MsgBox ('Switch state is Com to port ' & SwState)
End if
```

Visual C++

```
SwState = MyPTE1->Get_SP4T_State();
if (SwState > 0)
{
        MessageBox::Show("Switch state is Com to port " + SwState);
}
```

Visual C#

```
SwState = MyPTE1.Get_SP4T_State();
if (SwState > 0)
{
        MessageBox.Show("Switch state is Com to port " + SwState);
}
```

Matlab

```
[SwState] = MyPTE1.Get_SP4T_State()
If SwState > 0 then
{
        MsgBox ('Switch state is Com to port ', SwState)
}
```

See Also

Set SP4T Switch

2.4 (h) - Send SCPI Switch Command

Declaration

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

Description

This function does not apply to USB-SP4T-63.

2.4 (i) - Set Address of Switch

Declaration

```
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

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

2.4 (j) - Get Address of Switch

Declaration

```
int Get_Address()
```

Description

This function returns the address of the connected switch.

Parameters

Data Type	Variable	Description
none		

Return Values

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

Examples

Visual Basic

```
addr = MyPTE1.Get_Address()
```

Visual C++

```
addr = MyPTE1->Get_Address();
```

Visual C#

```
addr = MyPTE1.Get_Address();
```

Matlab

```
addr = MyPTE1.Get_Address
```

See Also

Connect to Switch by Address
Set Address of Switch
Get List of Available Addresses

2.4 (k) - Get List of Connected Serial Numbers

Declaration

```
int Get_Available_SN_List(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	Required. string variable which will be updated with a list of all available serial numbers, separated by a single space character; for example “11301020001 11301020002 11301020003”.

Return Values

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

Example

Visual Basic

```
If MyPTE1.Get_Available_SN_List(SN_List) > 0 Then
        array_SN() = Split(SN_List, " ")
                ' Split the list into an array of serial numbers
        For i As Integer = 0 To array_SN.Length - 1
                ' Loop through the array and use each serial number
        Next
End If
```

Visual C++

```
if (MyPTE1 ->Get_Available_SN_List(SN_List) > 0)
{
        // split the List into array of SN's
}
```

Visual C#

```
if (MyPTE1.Get_Available_SN_List(ref(SN_List)) > 0)
{
        // split the List into array of SN's
}
```

Matlab

```
[status, SN_List]= MyPTE1.Get_Available_SN_List(SN_List)
If status > 0 then
{
        % split the List into array of SN's
}
```

See Also

Connect to Switch
Get List of Available Addresses

2.4 (I) - Get List of Available Addresses

Declaration

```
int Get_Available_Address_List(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	Required. string variable which the function will update 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

Visual Basic

```
If MyPTE1.Get_Available_Add_List(st_Ad_List) > 0 Then
                ' Get list of available addresses
        array_Ad() = Split(st_Ad_List, " ")
                ' Split the list into an array of addresses
        For i As Integer = 0 To array_Ad.Length - 1
                ' Loop through the array and use each address
        Next
End If
```

Visual C++

```
if (MyPTE1->Get_Available_Address_List(Add_List) > 0);
{       // split the List into array of Addresses
}
```

Visual C#

```
if (MyPTE1.Get_Available_Address_List(ref(Add_List)) > 0)
{       // split the List into array of Addresses
}
```

Matlab

```
[status, Add_List]= MyPTE1.Get_Available_Address_List(Add_List)
If status > 0 then
{       % split the List into array of Addresses
}
```

See Also

Connect to Switch by Address
Get List of Connected Serial Numbers

2.4 (m) - Get USB Connection Status

Declaration

```
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

Visual Basic

```
If MyPTE1.GetUSBConnectionStatus = 1 Then
        ' switch is connected
End If
```

Visual C++

```
if (MyPTE1->GetUSBConnectionStatus() == 1)
{
        // switch is connected
}
```

Visual C#

```
if (MyPTE1.GetUSBConnectionStatus() == 1)
{
        // switch is connected
}
```

Matlab

```
usbstatus = MyPTE1.GetUSBConnectionStatus
If usbstatus == 1 then
{
        % switch is connected
}
```

2.4 (n) - Get USB Device Name

Declaration

```
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

Visual Basic

```
usbname = MyPTE1.GetUSBDeviceName()
        ' Return the USB device name as a string
```

Visual C++

```
usbname = MyPTE1->GetUSBDeviceName();
        // Return the USB device name as a string
```

Visual C#

```
usbname = MyPTE1.GetUSBDeviceName();
        // Return the USB device name as a string
```

Matlab

```
usbname = MyPTE1.GetUSBDeviceName
        % Return the USB device name as a string
```

2.4 (o) - Get Firmware

Declaration

```
int GetExtFirmware(int A0, int A1, int A2, 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

Visual Basic

```
If MyPTE1.GetExtFirmware(A0, A1, A2, Firmware) > 0 Then
        MsgBox ("Firmware version is " & Firmware)
End If
```

Visual C++

```
if (MyPTE1->GetExtFirmware(A0, A1, A2, Firmware) > 0 )
{
        MessageBox::Show("Firmware version is " + Firmware);
}
```

Visual C#

```
if (MyPTE1.GetExtFirmware(ref(A0, A1, A2, Firmware)) > 0 )
{
        MessageBox.Show("Firmware version is " + Firmware);
}
```

Matlab

```
[status, A0, A1, A2, Firmware]=MyPTE1.GetExtFirmware(A0, A1, A2, Firmware)
If status > 0 then
{
        msgbox('Firmware version is ', Firmware)
}
```

2.4 (p) - Get Firmware Version (Antiquated)

Declaration

```
string GetFirmware()
```

Description

This function returns an internal code for the firmware version of the switch.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
string	Firmware	The firmware version, eg: "A3"

Examples

Visual Basic

```
FW = MyPTE1.GetFirmware()
```

Visual C++

```
FW = MyPTE1->GetFirmware();
```

Visual C#

```
FW = MyPTE1.GetFirmware();
```

Matlab

```
FW = MyPTE1.GetFirmware()
```

See Also

Get Firmware

2.5 - Switching Sequence DLL Functions

USB-SP4T-63 supports 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. Firmware A3 or later is required.

2.5 (a) - Set Number of Steps

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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:

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

2.5 (b) - Get Number of Steps

Declaration

```
int GetSequence_NoOfSteps()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic

```
steps = MyPTE1.GetSequence_NoOfSteps()
```

Visual C++

```
steps = MyPTE1->GetSequence_NoOfSteps();
```

Visual C#

```
steps = MyPTE1.GetSequence_NoOfSteps();
```

Matlab

```
steps = MyPTE1.GetSequence_NoOfSteps
```

See Also

Set Number of Steps

2.5 (c) - Set Step

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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, 1 to 4 (the port which is to be connected to the Com port)
int	Dwell	Dwell time
int	DwellUnits	Dwell time units: 0 = microseconds (µs) 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 µs:

Visual Basic

```
status = MyPTE1.SetSequence_Step(2, 3, 5, 0)
```

Visual C++

```
status = MyPTE1->SetSequence_Step(2, 3, 5, 0);
```

Visual C#

```
status = MyPTE1.SetSequence_Step(2, 3, 5, 0);
```

Matlab

```
status = MyPTE1.SetSequence_Step(2, 3, 5, 0)
```

See Also

Set Number of Steps
Get Step Switch State
Get Step Dwell Time
Get Step Dwell Time Units

2.5 (d) - Get Step Switch State

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic
```
state = MyPTE1.GetSequence_SwitchTo(2)
```
Visual C++
```
state = MyPTE1->GetSequence_SwitchTo(2);
```
Visual C#
```
state = MyPTE1.GetSequence_SwitchTo(2);
```
Matlab
```
state = MyPTE1.GetSequence_SwitchTo(2)
```

See Also

Get Number of Steps
Set Step
Get Step Dwell Time
Get Step Dwell Time Units

2.5 (e) - Get Step Dwell Time

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic

```
dwell = MyPTE1.GetSequence_Dwell(2)
```

Visual C++

```
dwell = MyPTE1->GetSequence_Dwell(2);
```

Visual C#

```
dwell = MyPTE1.GetSequence_Dwell(2);
```

Matlab

```
dwell = MyPTE1.GetSequence_Dwell(2)
```

See Also

Get Number of Steps
Set Step
Get Step Switch State
Get Step Dwell Time Units

2.5 (f) - Get Step Dwell Time Units

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic
```
dwell = MyPTE1.GetSequence_DwellUnits(2)
```
Visual C++
```
dwell = MyPTE1->GetSequence_DwellUnits(2);
```
Visual C#
```
dwell = MyPTE1.GetSequence_DwellUnits(2);
```
Matlab
```
dwell = MyPTE1.GetSequence_DwellUnits(2)
```

See Also

Get Number of Steps
Set Step
Get Step Switch State
Get Step Dwell Time

2.5 (g) - Set Sequence Direction

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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:

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

2.5 (h) - Get Sequence Direction

Declaration

```
int GetSequence_Direction()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic
```
direction = MyPTE1.GetSequence_Direction()
```
Visual C++
```
direction = MyPTE1->GetSequence_Direction();
```
Visual C#
```
direction = MyPTE1.GetSequence_Direction();
```
Matlab
```
direction = MyPTE1.GetSequence_Direction()
```

See Also

Set Sequence Direction

2.5 (i) - Set Number of Cycles

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic

```
status = MyPTE1.SetSequence_NoOfCycles(400)
```

Visual C++

```
status = MyPTE1->SetSequence_NoOfCycles(400);
```

Visual C#

```
status = MyPTE1.SetSequence_NoOfCycles(400);
```

Matlab

```
status = MyPTE1.SetSequence_NoOfCycles(400)
```

See Also

Set Number of Cycles
Enable / Disable Continuous Mode
Check Continuous Mode State

2.5 (j) - Get Number of Cycles

Declaration

```
int GetSequence_NoOfCycles()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic
```
cycles = MyPTE1.GetSequence_NoOfCycles()
```
Visual C++
```
cycles  = MyPTE1->GetSequence_NoOfCycles();
```
Visual C#
```
cycles = MyPTE1.GetSequence_NoOfCycles();
```
Matlab
```
cycles = MyPTE1.GetSequence_NoOfCycles
```

See Also

Set Number of Cycles
Enable / Disable Continuous Mode
Check Continuous Mode State

2.5 (k) - Enable / Disable Continuous Mode

Declaration

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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:

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

2.5 (I) - Check Continuous Mode State

Declaration

```
int GetSequence_ContinuousMode()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

Visual Basic
```
mode = MyPTE1.GetSequence_ContinuousMode()
```
Visual C++
```
mode = MyPTE1->GetSequence_ContinuousMode();
```
Visual C#
```
mode = MyPTE1.GetSequence_ContinuousMode();
```
Matlab
```
mode = MyPTE1.GetSequence_ContinuousMode()
```

See Also

Set Number of Cycles
Get Number of Cycles
Enable / Disable Continuous Mode

2.5 (m) - Start Sequence

Declaration

```
int SetSequence_ON()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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:

Visual Basic

```
status = MyPTE1.SetSequence_ON()
```

Visual C++

```
status = MyPTE1->SetSequence_ON();
```

Visual C#

```
status = MyPTE1.SetSequence_ON();
```

Matlab

```
status = MyPTE1.SetSequence_ON()
```

See Also

Stop Sequence

2.5 (n) - Stop Sequence

Declaration

```
int SetSequence_OFF()
```

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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:

Visual Basic
```
status = MyPTE1.SetSequence_OFF()
```
Visual C++
```
status = MyPTE1->SetSequence_OFF();
```
Visual C#
```
status = MyPTE1.SetSequence_OFF();
```
Matlab
```
status = MyPTE1.SetSequence_OFF()
```

See Also

Start Sequence

3 - Operating in a Linux Environment via USB

To open a connection to Mini-Circuits' solid state, USB controlled RF switches, the Vendor ID and Product ID are required:

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

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

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

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

Worked examples can be found in the Programming Examples & Troubleshooting Guide, downloadable from the Mini-Circuits website. The examples use the libhid and libusb libraries to interface with the programmable attenuator as a USB HID (Human Interface Device).

3.1 - Summary of Commands

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

3.1 (a) - Core Commands

	Description	Command Code (Byte 0)	Comments
a	Get Device Model Name	40	
b	Get Device Serial Number	41	
c	Set SP4T Switch	1 2 3 4	Com to port 1 Com to port 2 Com to port 3 Com to port 4
d	Get SP4T Switch State	15	
f	Get Firmware	99	

3.1 (b) - Switching Sequence Commands

These functions allow a pre-defined switching sequence to be programmed into the switch, USB-SP4T-63 with firmware A3 or later is required.

	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

3.2 - Core Commands

3.2 (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-SP4T-63 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	83	ASCII character code for S
6	80	ASCII character code for P
7	52	ASCII character code for 4
8	24	ASCII character code for T
9	45	ASCII character code for -
10	54	ASCII character code for 6
11	51	ASCII character code for 3
12	0	Zero value byte to indicate end of string

See Also

Get Device Serial Number

3.2 (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

3.2 (c) - Set SP4T Switch

Description

Sets an SP4T switch.

Applies To

USB-SP4T-63. For all other models refer to Send SCPI Switch Command.

Transmit Array

Byte	Data	Description
0	1 - 4	Interrupt code for Set SP4T Switch state: 1 = Com to port 1 2 = Com to port 2 3 = Com to port 3 4 = Com to port 4
1 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	1 - 4	Interrupt code for Set SP4T Switch state: 1 = Com to port 1 2 = Com to port 2 3 = Com to port 3 4 = Com to port 4
1 - 63	Not significant	"Don't care" bytes, can be any value

Example

The following transmit array will set the SP4T switch to position 3 (Com connected to port 3):

Byte	Data	Description
0	3	Set SP4T to state 3

See Also

Get SP4T Switch State

3.2 (d) - Get SP4T Switch State

Description

Returns the state of the SP4T switch.

Applies To

USB-SP4T-63. For all other models refer to Send SCPI Switch Command.

Transmit Array

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

Returned Array

Byte	Data	Description
0	15	Interrupt code for Get SP4T Switch State
1	Switch State	Numeric value indicating the switch state: 1 = Com to port 1 2 = Com to port 2 3 = Com to port 3 4 = Com to port 4
2 - 63	Not significant	"Don't care" bytes, can be any value

Example

The below returned array indicates the switch is set as com to port 3:

Byte	Data	Description
0	15	Interrupt code for Get All SP4T Switch States
1	3	Switch set to state 3 (Com to port 3)

See Also

Set SP4T Switch

3.2 (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

3.3 - Switching Sequence Commands

USB-SP4T-63 supports 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. Firmware A3 or later is required.

3.3 (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
USB-SP4T-63	A3 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

3.3 (b) - Get Number of Steps

Description

Returns the number of steps in the switching sequence.

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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 (µs) 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 µs:

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 (µs)

See Also

Get Step

3.3 (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
USB-SP4T-63	A3 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 (µs) 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 µs:

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 (µs)

See Also

Set Step

3.3 (e) - Set Direction

Description

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

3.3 (f) - Get Direction

Description

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

Applies To

Supported Models	Required Firmware
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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

3.3 (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
USB-SP4T-63	A3 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