Test Solutions - Programming Manual

Programmable Attenuators

ZVVA Series USB & RS232 Programmable Attenuators
RUDAT Series USB & RS232 Programmable Attenuators
RCDAT Series USB & Ethernet Programmable Attenuators
RC4DAT Series USB & Ethernet Multi-Channel Programmable Attenuators

Mini-Circuits®

www.minicircuits.com | PO Box 350166, Brooklyn, NY 11235-0003 | +1 718-934-4500 | sales@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: testsolutions@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 and Ethernet controlled, programmable attenuators. The contents apply to:

- ZVVA Series (USB controlled) variable attenuators
- RUDAT Series (USB & RS232 controlled) single channel attenuators
- RCDAT Series (USB & Ethernet controlled) single channel attenuators
- RC4DAT Series (USB & Ethernet controlled) multi-channel attenuators

For instructions on using the supplied GUI program, or connecting the PTE hardware, please see the User Guide at:
http://www.minicircuits.com/support/softwaredownload.html.

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 programmable attenuator 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:
http://www.minicircuits.com/support/software_download.html

For details on individual models, application notes, GUI installation instructions and user guides please see:
http://www.minicircuits.com/products/PortableTestEquipment.shtml

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

2 - Operating in a Windows Environment via USB

When connected by USB, the computer will recognize the programmable attenuator as a Human Interface Device (HID). In this mode of operation the DLL file provides the method of control. Alternatively, the RUDAT series can be operated over a serial RS232 connection and the RCDAT series can be operated over an Ethernet TCP/IP Network (please see Serial Control Using RS232 Communication and Ethernet Control over IP Networks for details).

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 the Mini-Circuits programmable attenuator.

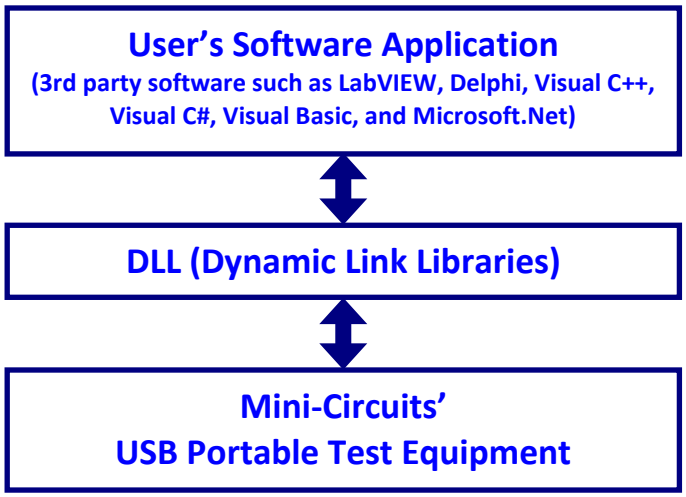


Fig 2.1-a: DLL Interface Concept

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' programmable attenuators 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_rudat.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_rudat.dll`
   For 64-bit Windows operating systems type (see Fig 2.1-e):
   `\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\mcl_rudat.dll`
4. Hit enter to confirm and a message box will appear to advise of successful registration.

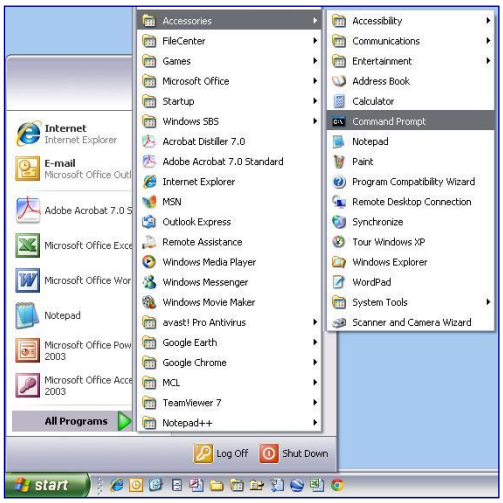


Fig 2.1-b: Opening the Command Prompt in Windows XP

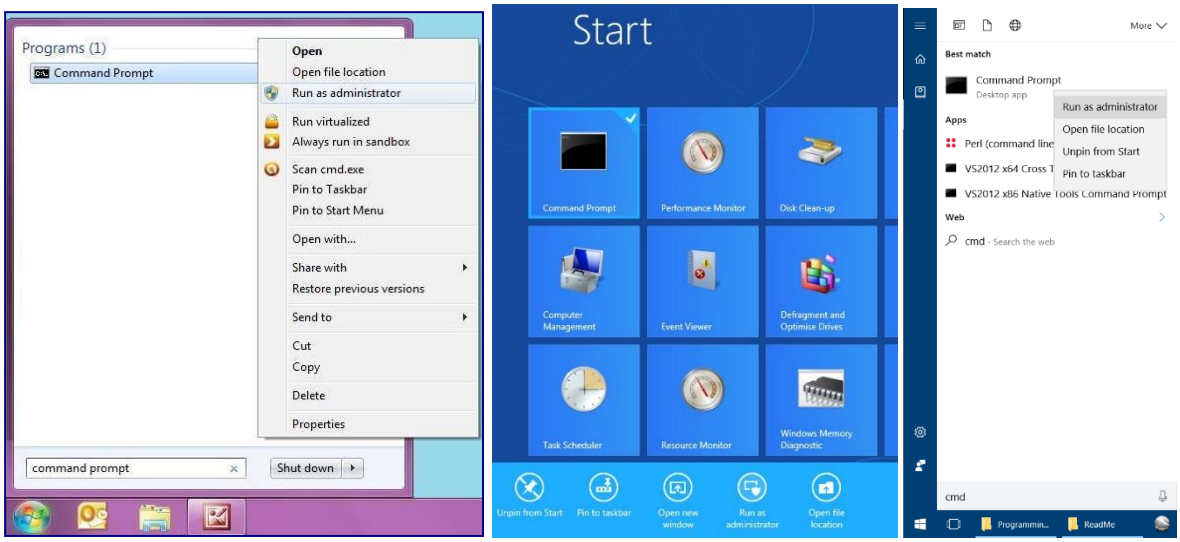


Fig 2.1-c: Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)

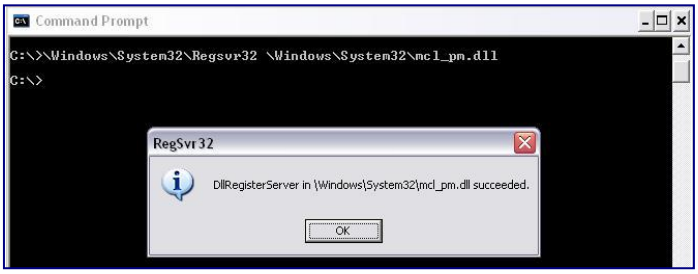


Fig 2.1-d: Registering the DLL in a 32-bit environment

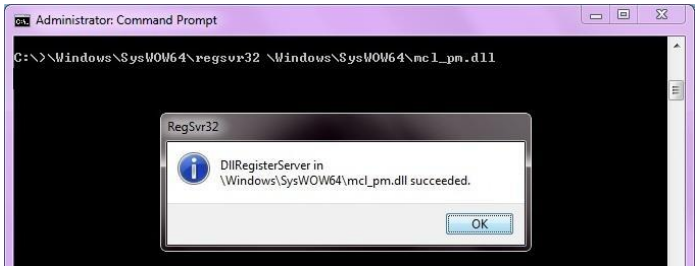


Fig 2.1-e: Registering the DLL in a 64-bit environment

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' programmable attenuators 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_rudat64.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 (Dynamic Linked Library)

The DLL file is installed in the host PC's system folders using the steps outlined above. Most programming environments will require a reference to be set to the DLL. Within the program, a new instance of the DLL's USB attenuator control class can be created for each programmable attenuator to control. The details of this vary between programming environments and languages but Mini-Circuits can provide detailed support on request. In the following examples, MyPTE1 and MyPTE2 will be used as names of 2 declared attenuator objects.

2.2 (a) - Example Declarations using the ActiveX DLL (mcl_rudat.dll)

Visual Basic

```
Public MyPTE1 As New mcl_RUDAT.USB_DAT
        ' Initialize new attenuator object, assign to MyPTE1
Public MyPTE2 As New mcl_RUDAT.USB_DAT
        ' Initialize new attenuator object, assign to MyPTE2
```

Visual C++

```
mcl_RUDAT::USB_DAT ^MyPTE1 = gcnew mcl_RUDAT::USB_DAT();
        // Initialize new attenuator instance, assign to MyPTE1
mcl RUDAT::USB DAT ^MyPTE2 = gcnew mcl RUDAT::USB DAT();
        // Initialize new attenuator instance, assign to MyPTE2
```

Visual C#

```
mcl RUDAT.USB DAT MyPTE1 = new mcl RUDAT.USB DAT();
        // Initialize new attenuator instance, assign to MyPTE1
mcl_RUDAT.USB_DAT MyPTE2 = new mcl_RUDAT.USB_DAT();
        // Initialize new attenuator instance, assign to MyPTE2
```

Matlab

```
MyPTE1 = actxserver('mcl_RUDAT.USB_DAT')
        % Initialize new attenuator instance, assign to MyPTE1
MyPTE2 = actxserver('mcl RUDAT.USB DAT')
        % Initialize new attenuator instance, assign to MyPTE2
```

2.2 (b) - Example Declarations using the .NET DLL (mcl_rudat64.dll)

Visual Basic

```
Public MyPTE1 As New mcl RUDAT64.USB RUDAT
        ' Initialize new attenuator object, assign to MyPTE1
Public MyPTE2 As New mcl_RUDAT64.USB_RUDAT
        ' Initialize new attenuator object, assign to MyPTE2
```

Visual C++

```
mcl_RUDAT64::USB_RUDAT^MyPTE1 = gcnew mcl_RUDAT64::USB_RUDAT();
        // Initialize new attenuator instance, assign to MyPTE1
mcl RUDAT64::USB RUDAT^MyPTE2 = gcnew mcl RUDAT64::USB RUDAT();
        // Initialize new attenuator instance, assign to MyPTE2
```

Visual C#

```
mcl RUDAT64.USB RUDATMyPTE1 = new mcl RUDAT64.USB RUDAT();
        // Initialize new attenuator instance, assign to MyPTE1
mcl_RUDAT64.USB_RUDATMyPTE2 = new mcl_RUDAT64.USB_RUDAT();
        // Initialize new attenuator instance, assign to MyPTE2
```

Matlab

```
MCL_ATT=NET.addAssembly('C:\Windows\SysWOW64\mcl_RUDAT64.dll')
MyPTE1=mcl RUDAT64.USB RUDAT  % Initialize new attenuator instance
MyPTE2=mcl_RUDAT64.USB_RUDAT  % Initialize new attenuator instance
```

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) - DLL - General Functions

```
a) int Get_Available_SN_List(ByRef string SN_List)
b) int Get_Available_Address_List(ByRef string Add_List)
c) int Connect(Optional string SN)
d) int ConnectByAddress(Optional int Address)
e) void Disconnect()
f) int Read_ModelName(ByRef string ModelName)
g) int Read_SN(ByRef string SN)
h) int Set_Address(int Address)
i) int Get_Address()
j) int Set_StartUpAttIndicator(int Indicator)
k) int Get_StartUpAttIndicator()
l) int InitiateStoreLastAtt()
m) int Send_SCPI(string SndSTR, ByRef string RetSTR)
n) int GetUSBConnectionStatus()
o) int GetStatus()
p) int GetExtFirmware(ByRef int A0, ByRef int A1, ByRef int A2,
                                                  ByRef string Firmware)
```

2.3 (b) - DLL - ZVVA / RUDAT / RCDAT (Single Channel) Attenuation Functions

These functions apply to ZVVA, RUDAT & RCDAT models only:

```
a) int SetAttenuation(ByRef float TotalAtt)      (ActiveX)
   int SetAttenuation(float TotalAtt)            (.NET)
b) int Read_Att(ByRef float CAtt1)
c) int Set_StartUpAtt(single AttVal)
d) single Get_StartUpAtt()
```

2.3 (c) - DLL - RC4DAT (Multi-Channel) Attenuation Functions

These functions apply to RC4DAT models only:

```
a) int SetChannelAtt(int Channel, float TotalAtt)
b) int SetChannelsAtt(float Att1, int CH1, int CH2,
                                                  int CH3, int CH4)
c) float ReadChannelAtt(int Channel)
d) int Read4ChannelsAtt(ByRef float C1Att, ByRef float C2Att,
                                  ByRef float C3Att, ByRef float C4Att)
e) int Set_ChannelStartUpAtt(int Channel, int StartUpAtt)
f) int Get_ChannelStartUpAtt(int Channel)
```

2.3 (d) - DLL - Ethernet Configuration Functions

These functions apply to RCDAT & RC4DAT models only:

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

2.3 (e) - DLL - Attenuation Hopping Functions

These functions require firmware version B1 or later.

```
a) int Hop_SetNoOfPoints(int HopNoOfPoints)
b) int Hop_GetNoOfPoints()
c) int Hop_GetMaxNoOfPoints()
d) int Hop_SetDirection(int HopDirection)
e) int Hop_GetDirection()
f) int Hop_GetMaxDwell()
g) int Hop_GetMinDwell()
h) int Hop_SetPoint(int PointNo, float HopPower, int HopDwT,
                                                        int HopDwTUnits)
i) int Hop_GetPoint(int PointNo, ByRef float HopPower,
                                ByRef int HopDwT, ByRef int HopDwTUnits)
j) int Hop_SetActiveChannels(int CH1_YesNO, int CH2_YesNO,
                                          int CH3_YesNO, int CH4_YesNO)
k) int Hop_GetActiveChannels(ByRef int CH1_YesNO,
        ByRef int CH2_YesNO, ByRef int CH3_YesNO, ByRef int CH4_YesNO)
l) int Hop_SetPoint4Channels(int PointNo, float HopAtt1,
                           float HopAtt2, float HopAtt3, float HopAtt4,
                                          int HopDwT , int HopDwTUnits)
m) int Hop_GetPoint4Channels(int PointNo, ByRef float HopAtt1,
                              ByRef float HopAtt2, ByRef float HopAtt3,
                              ByRef float HopAtt4, ByRef string HopDwT)
n) int Hop_SetMode(int On_Off)
```

2.3 (f) - DLL - Attenuation Sweeping / Fading Functions

These functions require firmware version B1 or later.

```
a) int Sweep_SetDirection(int SweepDirection)
b) int Sweep_GetDirection()
c) int  Sweep_SetDwell(int Dwell, int Dwell_Units)
d) int Sweep_GetDwell()
e) int Sweep_GetMaxDwell()
f) int Sweep_GetMinDwell()
g) int Sweep_SetStartAtt(single Att)
h) single Sweep_GetStartAtt()
i) int Sweep_SetStopAtt(single Att)
j) single Sweep_GetStopAtt()
k) int Sweep_SetStepSize(single Att)
l) single Sweep_GetStepSize()
m) int Sweep_SetActiveChannels(int CH1_YesNO, int CH2_YesNO,
                                    int CH3_YesNO, int CH4_YesNO)
n) int Sweep_GetActiveChannels(ByRef int CH1_YesNO,
       ByRef int CH2_YesNO, ByRef int CH3_YesNO, ByRef int CH4_YesNO)
o) int Sweep_SetChannelStartAtt(int Channel, float Att)
p) float Sweep_GetChannelStartAtt(int Channel)
q) int Sweep_SetChannelStopAtt(int Channel, float Att)
r) float Sweep_GetChannelStopAtt(int Channel)
s) int Sweep_SetChannelStepSize(int Channel, float Att)
t) float Sweep_GetChannelStepSize(int Channel)
u) int Sweep_SetMode(int On_Off)
```

2.4 - DLL - General Functions

2.4 (a) - Get List of Connected Serial Numbers

Declaration

```
int Get_Available_SN_List(ByRef String SN_List)
```

Description

Returns a list of serial numbers for all connected programmable attenuators.

Parameters

Data Type	Variable	Description
String	SN_List	Variable passed by reference, to be updated with a list of all connected serial numbers, separated by a single space, for example “11301210001 11301210002 11301210003”.

Return Values

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

Examples

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

Get Device Serial Number

2.4 (b) - Get List of Available Addresses

Declaration

```
int Get_Available_Address_List(ByRef String Add_List)
```

Description

Returns a list of USB addresses for all connected programmable attenuators.

Parameters

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

Return Values

Data Type	Value	Description
int	0	Command failed
int	Non zero	The number of devices connected

Examples

Visual Basic

```
If MyPTE1.Get_Available_Add_List(Add_List) > 0 Then
        ' Get list of available addresses
    array_Ad() = Split(Add_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 Attenuator by Address
Set USB Address
Get USB Address

2.4 (c) - Connect to Attenuator

Declaration

```
int Connect(Optional String SN)
```

Description

This function is called to initialize the connection to a programmable attenuator. If multiple attenuators are connected to the same computer, then the serial number should be included, otherwise this can be omitted. 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 attenuator is no longer needed. The attenuator 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 programmable attenuator. Can be omitted if only one attenuator 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 attenuator 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 Attenuator by Address
Disconnect from Attenuator

2.4 (d) - Connect to Attenuator by Address

Declaration

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

Description

This function is called to initialize the USB connection to a programmable attenuator 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 attenuator is no longer needed. The attenuator should be disconnected on completion of the program using the Disconnect function.

Parameters

Data Type	Variable	Description
int	Address	Optional. A short containing the address of the attenuator. Can be omitted if only one device is connected but must be included otherwise.

Return Values

Data Type	Value	Description
int	0	No connection was possible
	1	Connection successfully established
	2	Device already connected

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 Attenuator
Disconnect from Attenuator

2.4 (e) - Disconnect from Attenuator

Declaration

```
Void Disconnect()
```

Description

This function is called to close the connection to the programmable attenuator. 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 attenuator from the computer, then reconnect the attenuator 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 Attenuator
Connect to Attenuator by Address

2.4 (f) - Read Model Name

Declaration

```
int Read_ModelName(ByRef String ModelName)
```

Description

This function is called to determine the Mini-Circuits part number of the connected programmable attenuator. 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 programmable attenuator.

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 attenuator is " & ModelName)
                ' Display a message stating the model name
End If
```

Visual C++

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

Visual C#

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

Matlab

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

See Also

Read Serial Number

2.4 (g) - Read Serial Number

Declaration

```
int Read_SN(ByRef String SN)
```

Description

This function is called to determine the serial number of the connected programmable attenuator. 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 serial number for the programmable attenuator.

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 generator is " & SN)
                'Display a message stating the serial number
End If
```

Visual C++

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

Visual C#

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

Matlab

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

See Also

Read Model Name
Get List of Connected Serial Numbers

2.4 (h) - Set USB Address

Declaration

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

Description

This function sets the internal address of the attenuator connected via USB (the factory default address is 255). This allows the user to connect by a short address rather than serial number in future.

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
	Non zero	Command completed successfully

Examples

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

Get USB Address
Get List of Available Addresses
Connect to Attenuator by Address

2.4 (i) - Get USB Address

Declaration

```
int Get_Address()
```

Description

This function returns the USB address of the connected attenuator.

Parameters

Data Type	Variable	Description
None		

Return Values

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

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

Set USB Address
Get List of Available Addresses
Connect to Attenuator by Address

2.4 (j) - Set Start-Up Attenuation Mode

Declaration

```
int Set_StartUpAttIndicator(int Indicator)
```

Description

Sets the start-up mode to be used by the attenuator, this specifies how the initial attenuation value will be chosen when DC power is applied.

Note: See Store Last Attenuation Value if operating in "Last Attenuation" mode.

Requirements

Firmware version A6 or later

Parameters

Data Type	Variable	Description
int	Indicator	A numeric code indicating the start-up attenuation mode: 76 = Last Attenuation - The attenuation will be set to the same level as when the device was last powered off 70 = Fixed Attenuation - The attenuation will be set to a pre-defined value 78 = Default - The attenuator will assume the factory default state (maximum attenuation)

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.Set_StartUpAttIndicator(70)
```
Visual C++
```
Status = MyPTE1->Set_StartUpAttIndicator(70);
```
Visual C#
```
Status = MyPTE1.Set_StartUpAttIndicator(70);
```
Matlab
```
Status = MyPTE1.Set_StartUpAttIndicator(70)
```

See Also

Get Start-Up Attenuation Mode
Set Start-Up Attenuation Value
Get Start-Up Attenuation Value
Store Last Attenuation Value

2.4 (k) - Get Start-Up Attenuation Mode

Declaration

```
int Get_StartUpAttIndicator()
```

Description

Returns the start-up mode to be used by the attenuator, this specifies how the initial attenuation value will be chosen when DC power is applied.

Requirements

Firmware version A6 or later

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	76	Last Attenuation - The attenuation will be set to the same level as when the device was last powered off
	70	Fixed Attenuation - The attenuation will be set to a pre-defined value
	78	Default - The attenuator will assume the factory default state (maximum attenuation)

Examples

Visual Basic

```
Mode = MyPTE1.Get_StartUpAttIndicator()
```

Visual C++

```
Mode = MyPTE1->Get_StartUpAttIndicator();
```

Visual C#

```
Mode = MyPTE1.Get_StartUpAttIndicator();
```

Matlab

```
Mode = MyPTE1.Get_StartUpAttIndicator()
```

See Also

Set Start-Up Attenuation Mode
Set Start-Up Attenuation Value
Get Start-Up Attenuation Value
Store Last Attenuation Value

2.4 (I) - Store Last Attenuation Value

Declaration

```
int InitiateStoreLastAtt()
```

Description

Saves the current attenuation value to permanent memory so that it can be recalled when the attenuator is next powered back on. Only applies when the attenuator is configured to power-up in "Last Attenuation" mode.

Requirements

Firmware version C3 or later

Return Values

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

Example

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

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode

2.4 (m) - Send SCPI Command

Declaration

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

Description

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

Parameters

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

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Send_SCPI("MN?", RetStr)
                ' Send SCPI command to return the model name
```

Visual C++

```
Status = MyPTE1->Send_SCPI("MN?", RetStr);
                // Send SCPI command to return the model name
```

Visual C#

```
Status = MyPTE1.Send_SCPI("MN?", RetStr);
                // Send SCPI command to return the model name
```

Matlab

```
Status = MyPTE1.Send_SCPI("MN?", RetStr)
                % Send SCPI command to return the model name
```

See Also

Summary of SCPI Commands / Queries

2.4 (n) - Get USB Connection Status

Declaration

```
int GetUSBConnectionStatus()
```

Description

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

Parameters

Data Type	Variable	Description
None		

Return Values

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

Examples

Visual Basic

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

Visual C++

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

Visual C#

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

Matlab

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

See Also

Get Firmware Version

2.4 (o) - Get Status (Antiquated)

Declaration

```
int GetStatus()
```

Description

This function is antiquated; please use Get USB Connection Status instead. GetStatus checks whether the USB connection to the attenuator is active.

Parameters

Data Type	Variable	Description
None		

Return Values

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

Example

Visual Basic
```
Status = MyPTE1.GetStatus()
```
Visual C++
```
Status = MyPTE1->GetStatus();
```
Visual C#
```
Status = MyPTE1.GetStatus();
```
Matlab
```
Status = MyPTE1.GetStatus()
```

See Also

Get USB Connection Status

2.4 (p) - Get Firmware

Declaration

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

Description

This function returns the internal firmware version of the attenuator 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)
}
```

2.5 - DLL - ZVVA / RUDAT / RCDAT (Single Channel) Attenuation Functions

These functions apply to RUDAT & RCDAT models only

2.5 (a) - Set Attenuation

Declaration - ActiveX

```
int SetAttenuation(ByRef Float TotalAtt)
```

Declaration - .NET

```
int SetAttenuation(Float TotalAtt)
```

Description

This function sets the RF attenuation level. The allowed attenuation range and precision is defined in the individual model datasheets.

Applies To

ZVVA, RUDAT and RCDAT models

Parameters

Data Type	Variable	Description
Float	TotalAtt	Required. Numeric value indicating the attenuation to set.

Return Values

Data Type	Value	Description
int	0	Command failed or invalid attenuation set
	1	Command completed successfully
	2	Requested attenuation was higher than the allowed range, the attenuation was set to the device's maximum allowed value

Examples

Visual Basic

```
Status = MyPTE1.SetAttenuation(TotalAtt)
```

Visual C++

```
Status = MyPTE1->SetAttenuation(TotalAtt);
```

Visual C#

```
Status = MyPTE1.SetAttenuation(TotalAtt);
```

Matlab

```
Status = MyPTE1.SetAttenuation(TotalAtt)
```

See Also

Read Attenuation

2.5 (b) - Read Attenuation

Declaration

```
int Read_Att(ByRef Float CAtt1)
```

Description

This function indicates the current attenuation setting.

Applies To

ZVVA, RUDAT and RCDAT models

Parameters

Data Type	Variable	Description
Float	CAtt1	Required. User defined variable which will be updated with the current attenuation setting.

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Read_Att(Att)
```

Visual C++

```
Status = MyPTE1->Read_Att(Att);
```

Visual C#

```
Status = MyPTE1.Read_Att(Att);
```

Matlab

```
Status = MyPTE1.Read_Att(Att)
```

See Also

Set Attenuation

2.5 (c) - Set Start-Up Attenuation Value

Declaration

```
int Set_StartUpAtt(single AttVal)
```

Description

Sets the attenuation value to be loaded when the attenuator is first powered up in "Fixed Attenuation" start-up mode.

Applies To

ZVVA, RUDAT and RCDAT models with firmware A6 or later

Parameters

Data Type	Variable	Description
single	AttVal	The initial attenuation level to be loaded on start-up

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Set_StartUpAtt(25.75)
```

Visual C++

```
Status = MyPTE1->Set_StartUpAtt(25.75);
```

Visual C#

```
Status = MyPTE1.Set_StartUpAtt(25.75);
```

Matlab

```
Status = MyPTE1.Set_StartUpAtt(25.75)
```

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Get Start-Up Attenuation Value

2.5 (d) - Get Start-Up Attenuation Value

Declaration

```
single Get_StartUpAtt()
```

Description

Gets the attenuation value to be loaded when the attenuator is first powered up in "Fixed Attenuation" start-up mode.

Applies To

ZVVA, RUDAT and RCDAT models with firmware A6 or later

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
single	AttVal	The initial attenuation level to be loaded on start-up

Examples

Visual Basic
```
Att = MyPTE1.Get_StartUpAtt()
```
Visual C++
```
Att = MyPTE1->Get_StartUpAtt();
```
Visual C#
```
Att = MyPTE1.Get_StartUpAtt();
```
Matlab
```
Att = MyPTE1.Get_StartUpAtt()
```

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Set Start-Up Attenuation Value
Store Last Attenuation Value

2.6 - DLL - RC4DAT (Multi-Channel) Attenuation Functions

These functions apply to RC4DAT models only

2.6 (a) - Set Attenuation - Single Channel

Declaration

```
int SetChannelAtt(int Channel, float Att)
```

Description

Sets the attenuation for a single channel within the multi-channel attenuator.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)
float	Att	The attenuation value (dB) to set

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.SetChannelAtt(2, 15.75)
```

Visual C++

```
Status = MyPTE1->SetChannelAtt(2, 15.75);
```

Visual C#

```
Status = MyPTE1.SetChannelAtt(2, 15.75);
```

Matlab

```
Status = MyPTE1.SetChannelAtt(2, 15.75)
```

See Also

Set Attenuation - All Channels
Get Attenuation Value - Single Channel
Get Attenuation Value - All Channels

2.6 (b) - Set Attenuation - All Channels

Declaration

```
int SetChannelsAtt(float Att, int CH1, int CH2, int CH3, int CH4)
```

Description

Sets up to 4 channels of the multi-channel attenuator to the same attenuation value.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
float	Att	The attenuation value (dB) to set
int	CH1	1 to set CH1, 0 to leave channel 1 unchanged
int	CH2	1 to set CH2, 0 to leave channel 3 unchanged
int	CH3	1 to set CH3, 0 to leave channel 3 unchanged
int	CH4	1 to set CH4, 0 to leave channel 4 unchanged

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.SetChannelsAtt(15.75, 1, 1, 0, 1)
' Set channels 1, 2 and 4 to 15.75 dB
```

Visual C++

```
Status = MyPTE1->SetChannelAtt(15.75, 1, 1, 0, 1);
// Set channels 1, 2 and 4 to 15.75 dB
```

Visual C#

```
Status = MyPTE1.SetChannelAtt(15.75, 1, 1, 0, 1);
// Set channels 1, 2 and 4 to 15.75 dB
```

Matlab

```
Status = MyPTE1.SetChannelAtt(15.75, 1, 1, 0, 1)
% Set channels 1, 2 and 4 to 15.75 dB
```

See Also

Set Attenuation - Single Channel
Get Attenuation Value - Single Channel
Get Attenuation Value - All Channels

2.6 (c) - Get Attenuation Value - Single Channel

Declaration

```
float ReadChannelAtt(int Channel)
```

Description

Returns the value for a single channel within the multi-channel attenuator.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)

Return Values

Data Type	Value	Description
float	AttVal	The attenuation level (dB) for the specified channel

Examples

Visual Basic

```
Att = MyPTE1.ReadChannelAtt(3)
```

Visual C++

```
Att = MyPTE1->ReadChannelAtt(3);
```

Visual C#

```
Att = MyPTE1.ReadChannelAtt(3);
```

Matlab

```
Att = MyPTE1.ReadChannelAtt(3)
```

See Also

Set Attenuation - Single Channel
Set Attenuation - All Channels
Get Attenuation Value - All Channels

2.6 (d) - Get Attenuation Value - All Channels

Declaration

```
int Read4ChannelsAtt(ByRef float C1Att, ByRef float C2Att,
                                    ByRef float C3Att, ByRef float C4Att)
```

Description

Returns the attenuation values for all channels within the multi-channel attenuator.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
float	C1Att	Reference to a variable which will be updated with the attenuation value for channel 1
float	C2Att	Reference to a variable which will be updated with the attenuation value for channel 2
float	C3Att	Reference to a variable which will be updated with the attenuation value for channel 3
float	C4Att	Reference to a variable which will be updated with the attenuation value for channel 4

Return Values

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

Examples

Visual Basic

```
status = MyPTE1.Read4ChannelsAtt(Att1, Att2, Att3, Att4)
```

Visual C++

```
status = MyPTE1->Read4ChannelsAtt(Att1, Att2, Att3, Att4);
```

Visual C#

```
status = MyPTE1.Read4ChannelsAtt(ref(Att1), ref(Att2), ref(Att3), ref(Att4));
```

Matlab

```
[status,Att1,Att2,Att3,Att4]=MyPTE1.Read4ChannelsAtt(Att1, Att2, Att3, Att4)
```

See Also

Set Attenuation - Single Channel
Set Attenuation - All Channels
Get Attenuation Value - Single Channel

2.6 (e) - Set Channel Start-Up Attenuation Value

Declaration

```
int Set_ChannelStartUpAtt(int Channel, int StartUpAtt)
```

Description

Sets the start up attenuation value for a single channel within the multi-channel attenuator (the attenuation value to be loaded when DC power is applied).

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)
int	StartUpAtt	The initial attenuation value (dB) to be loaded on start-up

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Set_ChannelStartUpAtt(3, 25.75)
```

Visual C++

```
Status = MyPTE1->Set_ChannelStartUpAtt(3, 25.75);
```

Visual C#

```
Status = MyPTE1.Set_ChannelStartUpAtt(3, 25.75);
```

Matlab

```
Status = MyPTE1.Set_ChannelStartUpAtt(3, 25.75)
```

See Also

Get Channel Start-Up Attenuation Value

2.6 (f) - Get Channel Start-Up Attenuation Value

Declaration

```
int Get_ChannelStartUpAtt(int Channel)
```

Description

Returns the start up attenuation value for a single channel within the multi-channel attenuator (the attenuation value to be loaded when DC power is applied).

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)

Return Values

Data Type	Value	Description
int	AttVal	The initial attenuation value (dB) to be loaded on start-up

Examples

Visual Basic

```
Att = MyPTE1.Get_ChannelStartUpAtt(3)
```

Visual C++

```
Att = MyPTE1->Get_ChannelStartUpAtt(3);
```

Visual C#

```
Att = MyPTE1.Get_ChannelStartUpAtt(3);
```

Matlab

```
Att = MyPTE1.Get_ChannelStartUpAtt(3)
```

See Also

Set Channel Start-Up Attenuation Value

2.7 - DLL - Ethernet Configuration Functions

These functions apply to RCDAT & RC4DAT models only

2.7 (a) - Get Ethernet Configuration

Declaration

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

Requirements

RC Series programmable attenuator with RJ45 network interface.

Description

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

Parameters

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

int	Gateway4	Required. Integer variable which will be updated with the last (lowest order) octet of the network gateway.

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4, M1, M2, M3, M4,
                                    _ GW1, GW2, GW3, GW4) > 0 Then

        MsgBox ("IP address: " & IP1 & "." & IP2 & "." & IP3 & "." & IP4)
        MsgBox ("Subnet Mask: " & M1 & "." & M2 & "." & M3 & "." & M4)
        MsgBox ("Gateway: " & GW1 & "." & GW2 & "." & GW3 & "." & GW4)

End If
```

Visual C++

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

}
```

Visual C#

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

}
```

Matlab

```
[status, IP1, IP2, IP3, IP4, M1, M2, M3, M4, GW1, GW2, GW3, GW4] =
MyPTE1.GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4, M1, M2, M3, M4, GW1,
GW2, GW3, GW4)
If status > 0 then
{
        MsgBox ("IP address: ", IP1, ".", IP2, ".", IP3, ".", IP4)
        MsgBox ("Subnet Mask: ", M1, "." & M2, "." & M3, ".", M4)
        MsgBox ("Gateway: ", GW1, ".", GW2, ".", GW3, ".", GW4)

}
```

See Also

Get MAC Address
Get TCP/IP Port

2.7 (b) - Get IP Address

Declaration

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

Description

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

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

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

Return Values

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

Example

Visual Basic

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

Visual C++

```
if (MyPTE1->GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4) > 0)
{
        MessageBox::Show("IP address: " + IP1 + "." + IP2 + "." + IP3 + "."
                                                                    _ + IP4);
}
```

Visual C#

```
if (MyPTE1.GetEthernet_CurrentConfig(IP1, IP2, IP3, IP4) > 0)
{
        MessageBox.Show("IP address: " + IP1 + "." + IP2 + "." + IP3 + "."
                                                                    _ + IP4);
}
```

Matlab

```
[status, IP1, IP2, IP3, IP4] = MyPTE1.GetEthernet_CurrentConfig(IP1, IP2,
IP3, IP4)
If status > 0 then
{
        MsgBox ("IP address: ", IP1, ".", IP2, ".", IP3, ".", IP4)
}
```

See Also

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

2.7 (c) - Get MAC Address

Declaration

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

Description

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

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

Data Type	Variable	Description
int	MAC1	Passed by reference to be updated with the decimal value of the first section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC1=11
int	MAC2	Passed by reference to be updated with the decimal value of the second section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC2=47
int	MAC3	Passed by reference to be updated with the decimal value of the third section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC3=165
int	MAC4	Passed by reference to be updated with the decimal value of the fourth section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC4=103
int	MAC5	Passed by reference to be updated with the decimal value of the fifth section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC5=137
int	MAC6	Passed by reference to be updated with the decimal value of the sixth section of the MAC address. For example: MAC address =11:47:165:103:137:171 MAC6=171

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_MACAddess(M1, M2, M3, M4, M5, M6) > 0 Then
        MsgBox ("MAC address: " & M1 & ":" & M2 & ":" & M3 & ":" & M4 & ":"
                                                        _ & M5 & ":" & M6)
End If
```

Visual C++

```
if (MyPTE1->GetEthernet_MACAddess(M1, M2, M3, M4, M5, M6) > 0)
{
        MessageBox::Show("MAC address: " + M1 + "." + M2 + "." + M3 + "."
                                                _ + M4 + "." + M5 + "." + M6);
}
```

Visual C#

```
if (MyPTE1.GetEthernet_MACAddess(M1, M2, M3, M4, M5, M6) > 0)
{
        MessageBox.Show("MAC address: " + M1 + "." + M2 + "." + M3 + "."
                                                _ + M4 + "." + M5 + "." + M6);
        }
```

Matlab

```
[status, M1, M2, M3, M4, M5, M6] = MyPTE1.GetEthernet_MACAddess(M1, M2, M3,
M4, M5, M6)
If status > 0 then
{
        MsgBox ("MAC address: ", M1, ".", M2, ".", M3, ".", M4, ".", M5, ".",
M6)
}
```

See Also

Get Ethernet Configuration

2.7 (d) - Get Network Gateway

Declaration

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

Description

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

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

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

Return Values

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

Example

Visual Basic

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

Visual C++

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

Visual C#

```
if (MyPTE1.GetEthernet_NetworkGateway(IP1, IP2, IP3, IP4) > 0)
{
        MessageBox.Show("Gateway: " + IP1 + "." + IP2 + "." + IP3 + "."
                                                                    _ + IP4);
}
```

Matlab

```
[status, IP1, IP2, IP3, IP4] = MyPTE1.GetEthernet_NetworkGateway(IP1, IP2,
IP3, IP4)
If status > 0 then
{
        MsgBox ("Gateway: ", IP1, ".", IP2, ".", IP3, ".", IP4)
}
```

See Also

Get Ethernet Configuration
Save Network Gateway

2.7 (e) - Get Subnet Mask

Declaration

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

Description

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

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

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

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_SubNetMask(b1, b2, b3, b4) > 0 Then
        MsgBox ("Subnet mask: " & b1 & "." & b2 & "." & b3 & "." & b4)
End If
```

Visual C++

```
if (MyPTE1->GetEthernet_SubNetMask(b1, b2, b3, b4) > 0)
{
        MessageBox::Show("Subnet mask: " + b1 + "." + b2 + "." + b3 + "."
                                                                    _ + b4);
}
```

Visual C#

```
if (MyPTE1.GetEthernet_SubNetMask(b1, b2, b3, b4) > 0)
{
        MessageBox.Show("Subnet mask: " + b1 + "." + b2 + "." + b3 + "."
                                                                    _ + b4);
}
```

Matlab

```
[status, b1, b2, b3, b4] = MyPTE1.GetEthernet_SubNetMask(b1, b2, b3, b4)
If status > 0 then
{
        MsgBox ("Subnet mask: ", b1, ".", b2, ".", b3, ".", b4)
}
```

See Also

Get Ethernet Configuration
Save Subnet Mask

2.7 (f) - Get TCP/IP Port

Declaration

```
int GetEthernet_TCPIPPort(ByRef int port)
```

Description

Returns the TCP/IP port used by the attenuator for HTTP communication (default is port 80). Port 23 is reserved for Telnet communication and cannot be set as the HTTP port.

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

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

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_TCPIPPort(port) > 0 Then
        MsgBox ("Port: " & port)
End If
```

Visual C++

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

Visual C#

```
if (MyPTE1.GetEthernet_TCPIPPort(port) > 0)
{
        MessageBox.Show("Port: " + port);
}
```

Matlab

```
[status, port] = MyPTE1.GetEthernet_TCPIPPort(port)
If status > 0 then
{
        MsgBox ("Port: ", port)
}
```

See Also

Save TCP/IP Port
Get Telnet Port

2.7 (g) - Get Telnet Port

Declaration

```
int GetEthernet_TelnetPort(ByRef int port)
```

Description

Returns the port used by the attenuator for Telnet communication (default is port 23).

Requirements

RC Series programmable attenuator with firmware C7 or above.

Parameters

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

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_TelnetPort(port) > 0 Then
        MsgBox ("Port: " & port)
End If
```

Visual C++

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

Visual C#

```
if (MyPTE1.GetEthernet_TelnetPort(port) > 0)
{
        MessageBox.Show("Port: " + port);
}
```

Matlab

```
[status, port] = MyPTE1.TelnetPort_SubNetMask(port)
If status > 0 then
{
        MsgBox ("Port: ", port)
}
```

See Also

Save Telnet Port
Get TCP/IP Port

2.7 (h) - Get DHCP Status

Declaration

```
int GetEthernet_UseDHCP()
```

Description

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

Parameters

Data Type	Variable	Description
None		

Return Values

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

Example

Visual Basic

```
DHCPstatus = MyPTE1.GetEthernet_UseDHCP()
```

Visual C++

```
DHCPstatus = MyPTE1->GetEthernet_UseDHCP();
```

Visual C#

```
DHCPstatus = MyPTE1.GetEthernet_UseDHCP();
```

Matlab

```
[DHCPstatus] = MyPTE1.GetEthernet_UseDHCP
```

See Also

Get Ethernet Configuration
Use DHCP

2.7 (i) - Get Password Status

Declaration

```
int GetEthernet_UsePWD()
```

Description

This function indicates whether the attenuator is currently configured to require a password for HTTP/Telnet communication.

Parameters

Data Type	Variable	Description
None		

Return Values

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

Example

Visual Basic

```
PWDstatus = MyPTE1.GetEthernet_UsePWD()
```

Visual C++

```
PWDstatus = MyPTE1->GetEthernet_UsePWD();
```

Visual C#

```
PWDstatus = MyPTE1.GetEthernet_UsePWD();
```

Matlab

```
[PWDstatus] = MyPTE1.GetEthernet_UsePWD
```

See Also

Get Password
Use Password
Set Password

2.7 (j) - Get Password

Declaration

```
int GetEthernet_PWD(ByRef String Pwd)
```

Description

Returns the password used by the attenuator for HTTP/Telnet communication. The password will be returned even if the device is not currently configured to require a password.

Requirements

RC Series programmable attenuator with RJ45 network interface.

Parameters

Data Type	Variable	Description
String	Pwd	Passed by reference, to be updated with the password

Return Values

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

Example

Visual Basic

```
If MyPTE1.GetEthernet_PWD(pwd) > 0 Then
        MsgBox ("Password: " & pwd)
End If
```

Visual C++

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

Visual C#

```
if (MyPTE1.GetEthernet_PWD(pwd) > 0) {
        MessageBox.Show("Password: " + pwd);
}
```

Matlab

```
[status, pwd] = MyPTE1.GetEthernet_PWD(pwd)
If status > 0 then
{
        MsgBox ("Password: ", pwd)
}
```

See Also

Get Password Status
Use Password
Set Password

2.7 (k) - Save IP Address

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0)
```

Visual C++

```
status = MyPTE1->SaveEthernet_IPAddress(192, 168, 1, 0);
```

Visual C#

```
status = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_IPAddress(192, 168, 1, 0)
```

See Also

Get Ethernet Configuration
Get IP Address

2.7 (I) - Save Network Gateway

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0)
```

Visual C++

```
status = MyPTE1->SaveEthernet_NetworkGateway(192, 168, 1, 0);
```

Visual C#

```
status = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_NetworkGateway(192, 168, 1, 0)
```

See Also

Get Ethernet Configuration
Get Network Gateway

2.7 (m) - Save Subnet Mask

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0)
```

Visual C++

```
status = MyPTE1->SaveEthernet_SubnetMask(255, 255, 255, 0);
```

Visual C#

```
status = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_SubnetMask(255, 255, 255, 0)
```

See Also

Get Ethernet Configuration
Get Subnet Mask

2.7 (n) - Save TCP/IP Port

Declaration

```
int SaveEthernet_TCPIPPort(int port)
```

Description

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

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

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_TCPIPPort(70)
```

Visual C++

```
status = MyPTE1->SaveEthernet_TCPIPPort(70);
```

Visual C#

```
status = MyPTE1.SaveEthernet_TCPIPPort(70);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_TCPIPPort(70)
```

See Also

Get TCP/IP Port
Save Telnet Port

2.7 (o) - Save Telnet Port

Declaration

```
int SaveEthernet_TelnetPort(int port)
```

Description

This function sets the port used by the attenuator for Telnet communication. The default is port 23

Applies To

RC series programmable attenuators with firmware C7 or later.

Parameters

Data Type	Variable	Description
int	port	Required. Numeric value of the Telnet port.

Return Values

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

Example

Visual Basic
```
status = MyPTE1.SaveEthernet_TelnetPort(22)
```
Visual C++
```
status = MyPTE1->SaveEthernet_TelnetPort(22);
```
Visual C#
```
status = MyPTE1.SaveEthernet_TelnetPort(22);
```
Matlab
```
[status] = MyPTE1.SaveEthernet_TelnetPort(22)
```

See Also

Save TCP/IP Port
Get Telnet Port

2.7 (p) - Use DHCP

Declaration

```
int SaveEthernet_UseDHCP(int UseDHCP)
```

Description

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

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_UseDHCP(1)
```

Visual C++

```
status = MyPTE1->SaveEthernet_UseDHCP(1);
```

Visual C#

```
status = MyPTE1.SaveEthernet_UseDHCP(1);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_UseDHCP(1)
```

See Also

Get DHCP Status

2.7 (q) - Use Password

Declaration

```
int SaveEthernet_UsePWD(int UsePwd)
```

Description

This function enables or disables the password requirement for HTTP/Telnet communication with the attenuator.

Parameters

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

Return Values

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

Example

Visual Basic

```
status = MyPTE1.SaveEthernet_UsePWD(1)
```

Visual C++

```
status = MyPTE1->SaveEthernet_UsePWD(1);
```

Visual C#

```
status = MyPTE1.SaveEthernet_UsePWD(1);
```

Matlab

```
[status] = MyPTE1.SaveEthernet_UsePWD(1)
```

See Also

Get Password Status
Get Password
Set Password

2.7 (r) - Set Password

Declaration

```
int SaveEthernet_PWD(String Pwd)
```

Description

This function sets the password used by the attenuator for HTTP/Telnet communication. The password will not affect attenuator operation unless Use Password is also enabled.

Parameters

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

Return Values

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

Example

Visual Basic
```
status = MyPTE1.SaveEthernet_PWD("123")
```
Visual C++
```
status = MyPTE1->SaveEthernet_PWD("123");
```
Visual C#
```
status = MyPTE1.SaveEthernet_PWD("123");
```
Matlab
```
[status] = MyPTE1.SaveEthernet_PWD("123")
```

See Also

Get Password Status
Get Password
Use Password

2.8 - DLL - Attenuation Hopping Functions

These functions require firmware version B1 or later.

Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.

An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

2.8 (a) - Hop Mode - Set Number of Points

Declaration

```
int Hop_SetNoOfPoints(int NoOfPoints)
```

Description

Sets the number of points to be used in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	NoOfPoints	The number of points to set in the hop sequence

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetNoOfPoints(3)
```

Visual C++

```
Status = MyPTE1->Hop_SetNoOfPoints(3);
```

Visual C#

```
Status = MyPTE1.Hop_SetNoOfPoints(3);
```

Matlab

```
Status = MyPTE1.Hop_SetNoOfPoints(3)
```

See Also

Hop Mode - Get Number of Points
Hop Mode - Get Maximum Number of Points

2.8 (b) - Hop Mode - Get Number of Points

Declaration

```
int Hop_GetNoOfPoints()
```

Description

Returns the number of points to be used in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	1 - 100	The number of points in the hop sequence

Examples

Visual Basic

```
Points = MyPTE1.Hop_GetNoOfPoints()
```

Visual C++

```
Points = MyPTE1->Hop_GetNoOfPoints();
```

Visual C#

```
Points = MyPTE1.Hop_GetNoOfPoints();
```

Matlab

```
Points = MyPTE1.Hop_GetNoOfPoints()
```

See Also

Hop Mode - Set Number of Points
Hop Mode - Get Maximum Number of Points

2.8 (c) - Hop Mode - Get Maximum Number of Points

Declaration

```
int Hop_GetMaxNoOfPoints()
```

Description

Returns the maximum number of points that can be used in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	MaxPoints	The maximum number of hop points

Examples

Visual Basic

```
MaxPoints = MyPTE1.Hop_GetMaxNoOfPoints()
```

Visual C++

```
MaxPoints = MyPTE1->Hop_GetMaxNoOfPoints();
```

Visual C#

```
MaxPoints = MyPTE1.Hop_GetMaxNoOfPoints();
```

Matlab

```
MaxPoints = MyPTE1.Hop_GetMaxNoOfPoints()
```

See Also

Hop Mode - Set Number of Points
Hop Mode - Get Number of Points

2.8 (d) - Hop Mode - Set Sequence Direction

Declaration

```
int Hop_SetDirection(int Direction)
```

Description

Sets the direction in which the attenuator will progress through the list of attenuation hops.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	Direction	Numeric value indicating the direction: 0 = Forward - The list of attenuation hops will be loaded from index 1 to index n 1 = Backwards - The list of attenuation hops will be loaded from index n to index 1 2 = Bi-directionally - The list of attenuation hops will be loaded in the forward and then reverse directions

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetDirection(1)
```

Visual C++

```
Status = MyPTE1->Hop_SetDirection(1);
```

Visual C#

```
Status = MyPTE1.Hop_SetDirection(1);
```

Matlab

```
Status = MyPTE1.Hop_SetDirection(1)
```

See Also

Hop Mode - Get Sequence Direction

2.8 (e) - Hop Mode - Get Sequence Direction

Declaration

```
int Hop_GetDirection()
```

Description

Returns the direction in which the attenuator will progress through the list of attenuation hops.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	0	Forward - The list of attenuation hops will be loaded from index 1 to index n
	1	Backwards - The list of attenuation hops will be loaded from index n to index 1
	2	Bi-directionally - The list of attenuation hops will be loaded in the forward and then reverse directions

Examples

Visual Basic

```
Points = MyPTE1.Hop_GetDirection()
```

Visual C++

```
Points = MyPTE1->Hop_GetDirection();
```

Visual C#

```
Points = MyPTE1.Hop_GetDirection();
```

Matlab

```
Points = MyPTE1.Hop_GetDirection()
```

See Also

Hop Mode - Set Sequence Direction

2.8 (f) - Hop Mode - Get Maximum Dwell Time

Declaration

```
int Hop_GetMaxDwell()
```

Description

Returns the maximum dwell time that can be used for any point in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	MaxDwell	Maximum hop dwell time

Examples

Visual Basic

```
MaxDwell = MyPTE1.Hop_GetMaxDwell()
```

Visual C++

```
MaxDwell = MyPTE1->Hop_GetMaxDwell();
```

Visual C#

```
MaxDwell = MyPTE1.Hop_GetMaxDwell();
```

Matlab

```
MaxDwell = MyPTE1.Hop_GetMaxDwell()
```

See Also

Hop Mode - Get Minimum Dwell Time

2.8 (g) - Hop Mode - Get Minimum Dwell Time

Declaration

```
int Hop_GetMinDwell()
```

Description

Returns the minimum dwell time that can be used for any point in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	MinDwell	Minimum hop dwell time

Examples

Visual Basic

```
MinDwell = MyPTE1.Hop_GetMinDwell()
```

Visual C++

```
MinDwell = MyPTE1->Hop_GetMinDwell();
```

Visual C#

```
MinDwell = MyPTE1.Hop_GetMinDwell();
```

Matlab

```
MinDwell = MyPTE1.Hop_GetMinDwell()
```

See Also

Hop Mode - Get Maximum Dwell Time

2.8 (h) - Hop Mode - Single Channel - Set Hop

Declaration

```
int Hop_SetPoint(int Point, single Att, int Dwell, int Dw_Units)
```

Description

Sets the attenuation level and dwell time for a specific point with in the hop sequence of a single channel (ZVVA / RUDAT / RCDAT) attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
int	Point	The index number of the point within the sequence (from 1 to the number of points set)
single	Att	The attenuation level to set for this point
int	Dwell	The dwell time for this point
int	Dw_Units	Numeric code indicating the dwell time units: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetPoint(1, 10.75, 600, 115)
```

Visual C++

```
Status = MyPTE1->Hop_SetPoint(1, 10.75, 600, 115);
```

Visual C#

```
Status = MyPTE1.Hop_SetPoint(1, 10.75, 600, 115);
```

Matlab

```
Status = MyPTE1.Hop_SetPoint(1, 10.75, 600, 115)
```

See Also

Hop Mode - Single Channel - Get Hop

2.8 (i) - Hop Mode - Single Channel - Get Hop

Declaration

```
int Hop_GetPoint(int Point, ByRef single Att, ByRef int Dwell,
                                                ByRef int Dw_Units)
```

Description

Gets the attenuation level and dwell time for a specific point with in the hop sequence of a single channel (ZVVA / RUDAT / RCDAT) attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
int	Point	The index number of the point within the sequence (from 1 to the number of points set)
single	Att	Variable passed by reference to be updated with the attenuation level set for this point
int	Dwell	Variable passed by reference to be updated with the dwell time set for this point
int	Dw_Units	Variable passed by reference to be updated with a numeric code indicating the dwell time units for this point: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_GetPoint(1, Att, Dwell, Dw_Units)
```

Visual C++

```
Status = MyPTE1->Hop_GetPoint(1, Att, Dwell, Dw_Units);
```

Visual C#

```
Status = MyPTE1.Hop_GetPoint(1, Att, Dwell, Dw_Units);
```

Matlab

```
[Status, Att, Dwell, Dw_Units] = MyPTE1.Hop_GetPoint(1, Att, Dwell, Dw_Units)
```

See Also

Hop Mode - Single Channel - Set Hop

2.8 (j) - Hop Mode - Multi-Channel - Set Active Channels

Declaration

```
int Hop_SetActiveChannels(int CH1_YesNo, int CH2_YesNo,
                                         int CH3_YesNo, int CH4_YesNo)
```

Description

Sets which of the 4 channels of a multi-channel attenuator are to be included in the attenuation hop sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	CH1_YesNo	1 to include CH1 in the hop, 0 to leave unchanged
int	CH2_YesNo	1 to include CH2 in the hop, 0 to leave unchanged
int	CH3_YesNo	1 to include CH3 in the hop, 0 to leave unchanged
int	CH4_YesNo	1 to include CH4 in the hop, 0 to leave unchanged

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetActiveChannels(1, 1, 0, 0)
' Configure a hop for channels 1 and 2 only
```

Visual C++

```
Status = MyPTE1->Hop_SetActiveChannels(1, 1, 0, 0);
// Configure a hop for channels 1 and 2 only
```

Visual C#

```
Status = MyPTE1.Hop_SetActiveChannels(1, 1, 0, 0);
// Configure a hop for channels 1 and 2 only
```

Matlab

```
Status = MyPTE1.Hop_SetActiveChannels(1, 1, 0, 0)
% Configure a hop for channels 1 and 2 only
```

See Also

Hop Mode - Multi-Channel - Get Active Channels

2.8 (k) - Hop Mode - Multi-Channel - Get Active Channels

Declaration

```
int Hop_GetActiveChannels(ByRef int CH1_YesNo, ByRef int CH2_YesNo,
                          ByRef int CH3_YesNo, ByRef int CH4_YesNo)
```

Description

Checks which of the 4 channels of a multi-channel attenuator are to be included in the attenuation hop sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	CH1_YesNo	Reference to a variable which will be updated with the status of CH1 (1 if it included in the hop, 0 otherwise)
int	CH2_YesNo	Reference to a variable which will be updated with the status of CH2 (1 if it included in the hop, 0 otherwise)
int	CH3_YesNo	Reference to a variable which will be updated with the status of CH3 (1 if it included in the hop, 0 otherwise)
int	CH4_YesNo	Reference to a variable which will be updated with the status of CH4 (1 if it included in the hop, 0 otherwise)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_GetActiveChannels(CH1, CH2, CH3, CH4)
' Check which channels are to be included in the hop
```

Visual C++

```
Status = MyPTE1->Hop_GetActiveChannels(CH1, CH2, CH3, CH4);
// Check which channels are to be included in the hop
```

Visual C#

```
Status=MyPTE1.Hop_GetActiveChannels(ref(CH1),ref(CH2),ref(CH3),ref(CH4));
// Check which channels are to be included in the hop
```

Matlab

```
[Status,CH1,CH2,CH3,CH4]=MyPTE1.Hop_GetActiveChannels(CH1, CH2, CH3, CH4)
% Check which channels are to be included in the hop
```

See Also

Hop Mode - Multi-Channel - Set Active Channels

2.8 (l) - Hop Mode - Multi-Channel Hop - Set Hop Point for All Channels

Declaration

```
int Hop_SetPoint4Channels(int PointNo, float HopAtt1, float HopAtt2,
                          float HopAtt3, float HopAtt4, int Dwell, int DwellUnits)
```

Description

Sets the attenuation values to be loaded for each channel and the dwell time for a specific point within the hop sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	PointNo	The ndex number of the hop within the hop sequence
float	HopAtt1	Attenuation to set for CH1 at this point in the hop sequence
float	HopAtt2	Attenuation to set for CH2 at this point in the hop sequence
float	HopAtt3	Attenuation to set for CH3 at this point in the hop sequence
float	HopAtt4	Attenuation to set for CH4 at this point in the hop sequence
int	Dwell	The dwell time for this point in the hop sequence
int	DwellUnits	Numeric code indicating the dwell time units: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetPoint4Channels(2, 20.25, 10.75, 0, 0, 20, 109)
' Set the attenuation values with 20 ms dwell time for hop point 2
```

Visual C++

```
Status = MyPTE1->Hop_SetPoint4Channels(2, 20.25, 10.75, 0, 0, 20, 109);
// Set the attenuation values with 20 ms dwell time for hop point 2
```

Visual C#

```
Status = MyPTE1.Hop_SetPoint4Channels(2, 20.25, 10.75, 0, 0, 20, 109);
// Set the attenuation values with 20 ms dwell time for hop point 2
```

Matlab

```
Status = MyPTE1.Hop_SetPoint4Channels(2, 20.25, 10.75, 0, 0, 20, 109)
% Set the attenuation values with 20 ms dwell time for hop point 2
```

See Also

Hop Mode - Multi-Channel - Get Hop Point for All Channels

2.8 (m) - Hop Mode - Multi-Channel - Get Hop Point for All Channels

Declaration

```
int Hop_GetPoint4Channels(int PointNo, ByRef float HopAtt1,
        ByRef float HopAtt2, ByRef float HopAtt3, ByRef float HopAtt4,
                                    ByRef int Dwell, ByRef int DwellUnits)
```

Description

Returns the attenuation values to be loaded for each channel and the dwell time for a specific point within the hop sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	PointNo	The index number of the hop within the hop sequence
float	HopAtt1	Reference to a variable which will be updated with the attenuation for CH1 at this point in the hop sequence
float	HopAtt2	Reference to a variable which will be updated with the attenuation for CH2 at this point in the hop sequence
float	HopAtt3	Reference to a variable which will be updated with the attenuation for CH3 at this point in the hop sequence
float	HopAtt4	Reference to a variable which will be updated with the attenuation for CH4 at this point in the hop sequence
int	Dwell	Reference to a variable which will be updated with the dwell time for this point in the hop sequence
int	DwellUnits	Reference to a variable which will be updated with a numeric code indicating the dwell time units: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_GetPoint4Channels(2, CH1, CH2, CH3, CH4, DWT, DWTU)
' Get the details for point 2 in the hop sequence
```

Visual C++

```
Status = MyPTE1->Hop_GetPoint4Channels(2, CH1, CH2, CH3, CH4, DWT, DWTU);
// Get the details for point 2 in the hop sequence
```

Visual C#

```
Status
=MyPTE1.Hop_GetPoint4Channels(ref(CH1),ref(CH2),ref(CH3),ref(CH4),
                                                    ref(DWT),ref(DWTU));
// Get the details for point 2 in the hop sequence
```

Matlab

```
[Status, CH1, CH2, CH3, CH4, DWT, DWTU]
        = MyPTE1.Hop_GetPoint4Channels(2, CH1, CH2, CH3, CH4, DWT, DWTU)
        % Get the details for point 2 in the hop sequence
```

See Also

Hop Mode - Multi-Channel Hop - Set Hop Point for All Channels

2.8 (n) - Hop Mode - Turn On / Off

Declaration

```
int Hop_SetMode(int On_Off)
```

Description

Enables or disables the hop sequence according to the previously configured parameters.

Notes:

- Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.
- An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	On_Off	Numeric value to enable/disable the hop sequence: 0 = Disable the hop sequence 1 = Enable the hop sequence

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetMode(1)
```

Visual C++

```
Status = MyPTE1->Hop_SetMode(1);
```

Visual C#

```
Status = MyPTE1.Hop_SetMode(1);
```

Matlab

```
Status = MyPTE1.Hop_SetMode(1)
```

2.9 - DLL - Attenuation Sweeping / Fading Functions

These functions require firmware version B1 or later.

Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.

An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

2.9 (a) - Sweep Mode - Set Sweep Direction

Declaration

```
int Sweep_SetDirection(int Direction)
```

Description

Sets the direction in which the attenuator will sweep between the start and stop attenuation values.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	Direction	Numeric value indicating the direction: 0 = Forward - Sweep from start to stop attenuation 1 = Backwards - Sweep from stop to start attenuation 2 = Bi-directionally - Sweep in the forward and then reverse directions

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetDirection(1)
```

Visual C++

```
Status = MyPTE1->Sweep_SetDirection(1);
```

Visual C#

```
Status = MyPTE1.Sweep_SetDirection(1);
```

Matlab

```
Status = MyPTE1.Sweep_SetDirection(1)
```

See Also

Sweep Mode - Get Sweep Direction

2.9 (b) - Sweep Mode - Get Sweep Direction

Declaration

```
int Sweep_GetDirection()
```

Description

Returns the direction in which the attenuator will sweep between the start and stop attenuation values.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	0	Forward - Sweep from start to stop attenuation
	1	Backwards - Sweep from stop to start attenuation
	2	Bi-directionally - Sweep in the forward and then reverse directions

Examples

Visual Basic

```
Points = MyPTE1.Sweep_GetDirection()
```

Visual C++

```
Points = MyPTE1->Sweep_GetDirection();
```

Visual C#

```
Points = MyPTE1.Sweep_GetDirection();
```

Matlab

```
Points = MyPTE1.Sweep_GetDirection()
```

See Also

Sweep Mode - Set Sweep Direction

2.9 (c) - Sweep Mode - Set Dwell Time

Declaration

```
int Sweep_SetDwell(int Dwell, int Dwell_Units)
```

Description

Sets the dwell time to be used for each attenuation step within the sweep.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	Dwell	The dwell time
int	Dwell_Units	Numeric code indicating the dwell time units: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetDwell(650, 117)
```

Visual C++

```
Status = MyPTE1->Sweep_SetDwell(650, 117);
```

Visual C#

```
Status = MyPTE1.Sweep_SetDwell(650, 117);
```

Matlab

```
Status = MyPTE1.Sweep_SetDwell(650, 117)
```

See Also

Sweep Mode - Get Dwell Time
Sweep Mode - Get Maximum Dwell Time
Sweep Mode - Get Minimum Dwell Time

2.9 (d) - Sweep Mode - Get Dwell Time

Declaration

```
int Sweep_GetDwell(ByRef int Dwell, ByRef int Dwell_Units)
```

Description

Returns the dwell time to be used for each attenuation step within the sweep.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	Dwell	Variable passed by reference, to be updated with the dwell time
int	Dwell_Units	Variable passed by reference, to be updated with a numeric value indicating the dwell time units: 117 = Dwell time in microseconds (µs) 109 = Dwell time in milliseconds (ms) 115 = Dwell time in seconds (s)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_GetDwell(Dwell, Dw_Units)
```

Visual C++

```
Status = MyPTE1->Sweep_GetDwell(Dwell, Dw_Units);
```

Visual C#

```
Status = MyPTE1.Sweep_GetDwell(Dwell, Dw_Units);
```

Matlab

```
[Status, Dwell, Dw_Units] = MyPTE1.Sweep_GetDwell(Dwell, Dw_Units)
```

See Also

Sweep Mode - Set Dwell Time
Sweep Mode - Get Maximum Dwell Time
Sweep Mode - Get Minimum Dwell Time

2.9 (e) - Sweep Mode - Get Maximum Dwell Time

Declaration

```
int Sweep_GetMaxDwell()
```

Description

Returns the maximum dwell time that can be used for each attenuation step within the sweep.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	MaxDwell	Maximum sweep dwell time

Examples

Visual Basic

```
MaxDwell = MyPTE1.Sweep_GetMaxDwell()
```

Visual C++

```
MaxDwell = MyPTE1->Sweep_GetMaxDwell();
```

Visual C#

```
MaxDwell = MyPTE1.Sweep_GetMaxDwell();
```

Matlab

```
MaxDwell = MyPTE1.Sweep_GetMaxDwell()
```

See Also

Sweep Mode - Set Dwell Time
Sweep Mode - Get Dwell Time
Sweep Mode - Get Minimum Dwell Time

2.9 (f) - Sweep Mode - Get Minimum Dwell Time

Declaration

```
int Sweep_GetMinDwell()
```

Description

Returns the minimum dwell time that can be used for each attenuation step within the sweep.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
int	MinDwell	Minimum sweep dwell time

Examples

Visual Basic

```
MinDwell = MyPTE1.Sweep_GetMinDwell()
```

Visual C++

```
MinDwell = MyPTE1->Sweep_GetMinDwell();
```

Visual C#

```
MinDwell = MyPTE1.Sweep_GetMinDwell();
```

Matlab

```
MinDwell = MyPTE1.Sweep_GetMinDwell()
```

See Also

Sweep Mode - Set Dwell Time
Sweep Mode - Get Dwell Time
Sweep Mode - Get Maximum Dwell Time

2.9 (g) - Sweep Mode - Single Channel - Set Start Attenuation

Declaration

```
int Sweep_SetStartAtt(single Att)
```

Description

Sets the first attenuation level to be loaded during the sweep for a single channel attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
single	Att	The initial attenuation value to be loaded during the sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetStartAtt(0)
```

Visual C++

```
Status = MyPTE1->Sweep_SetStartAtt(0);
```

Visual C#

```
Status = MyPTE1.Sweep_SetStartAtt(0);
```

Matlab

```
Status = MyPTE1.Sweep_SetStartAtt(0)
```

See Also

Sweep Mode - Single Channel - Get Start Attenuation

2.9 (h) - Sweep Mode - Single Channel - Get Start Attenuation

Declaration

```
single Sweep_GetStartAtt()
```

Description

Returns the first attenuation level to be loaded during the sweep for a single channel attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
single	Att	The initial attenuation to be loaded during the sweep

Examples

Visual Basic

```
Att = MyPTE1.Sweep_GetStartAtt()
```

Visual C++

```
Att = MyPTE1->Sweep_GetStartAtt();
```

Visual C#

```
Att = MyPTE1.Sweep_GetStartAtt();
```

Matlab

```
Att = MyPTE1.Sweep_GetStartAtt()
```

See Also

Sweep Mode - Single Channel - Set Start Attenuation

2.9 (i) - Sweep Mode - Single Channel - Set Stop Attenuation

Declaration

```
int Sweep_SetStopAtt(single Att)
```

Description

Sets the final attenuation level to be loaded during the sweep for a single channel attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
single	Att	The final attenuation value to be loaded during the sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetStopAtt(25.75)
```

Visual C++

```
Status = MyPTE1->Sweep_SetStopAtt(25.75);
```

Visual C#

```
Status = MyPTE1.Sweep_SetStopAtt(25.75);
```

Matlab

```
Status = MyPTE1.Sweep_SetStopAtt(25.75)
```

See Also

Sweep Mode - Single Channel - Get Stop Attenuation

2.9 (j) - Sweep Mode - Single Channel - Get Stop Attenuation

Declaration

```
single Sweep_GetStopAtt()
```

Description

Returns the final attenuation level to be loaded during the sweep for a single channel attenuator.

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
single	Att	The final attenuation to be loaded during the sweep

Examples

Visual Basic

```
Att = MyPTE1.Sweep_GetStopAtt()
```

Visual C++

```
Att = MyPTE1->Sweep_GetStopAtt();
```

Visual C#

```
Att = MyPTE1.Sweep_GetStopAtt();
```

Matlab

```
Att = MyPTE1.Sweep_GetStopAtt()
```

See Also

Sweep Mode - Single Channel - Set Stop Attenuation

2.9 (k) - Sweep Mode - Single Channel - Set Step Size

Declaration

```
int Sweep_SetStepSize(single Att)
```

Description

Sets the attenuation step size that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
single	Att	The attenuation step size

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetStepSize(0.75)
```

Visual C++

```
Status = MyPTE1->Sweep_SetStepSize(0.75);
```

Visual C#

```
Status = MyPTE1.Sweep_SetStepSize(0.75)
```

Matlab

```
Status = MyPTE1.Sweep_SetStepSize(0.75)
```

See Also

Sweep Mode - Single Channel - Get Step Size

2.9 (I) - Sweep Mode - Single Channel - Get Step Size

Declaration

```
single Sweep_GetStepSize()
```

Description

Returns the attenuation step size that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

ZVVA, RUDAT and RCDAT models with firmware B1 or later

Parameters

Data Type	Variable	Description
none		

Return Values

Data Type	Value	Description
single	Att	The attenuation step size

Examples

Visual Basic

```
Att = MyPTE1.Sweep_GetStepSize()
```

Visual C++

```
Att = MyPTE1->Sweep_GetStepSize();
```

Visual C#

```
Att = MyPTE1.Sweep_GetStepSize();
```

Matlab

```
Att = MyPTE1.Sweep_GetStepSize()
```

See Also

Sweep Mode - Single Channel - Set Step Size

2.9 (m) - Sweep Mode - Multi-Channel - Set Active Channels

Declaration

```
int Sweep_SetActiveChannels(int CH1_YesNo, int CH2_YesNo,
                                          int CH3_YesNo, int CH4_YesNo)
```

Description

Sets which of the 4 channels of a multi-channel attenuator are to be included in an attenuation sweep sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	CH1_YesNo	1 to include CH1 in the sweep, 0 to leave unchanged
int	CH2_YesNo	1 to include CH2 in the sweep, 0 to leave unchanged
int	CH3_YesNo	1 to include CH3 in the sweep, 0 to leave unchanged
int	CH4_YesNo	1 to include CH4 in the sweep, 0 to leave unchanged

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetActiveChannels(1, 1, 0, 0)
' Configure a sweep for channels 1 and 2 only
```

Visual C++

```
Status = MyPTE1->Sweep_SetActiveChannels(1, 1, 0, 0);
// Configure a sweep for channels 1 and 2 only
```

Visual C#

```
Status = MyPTE1.Sweep_SetActiveChannels(1, 1, 0, 0);
// Configure a sweep for channels 1 and 2 only
```

Matlab

```
Status = MyPTE1.Sweep_SetActiveChannels(1, 1, 0, 0)
% Configure a sweep for channels 1 and 2 only
```

See Also

Sweep Mode - Multi-Channel - Get Active Channels

2.9 (n) - Sweep Mode - Multi-Channel - Get Active Channels

Declaration

```
int Sweep_GetActiveChannels(ByRef int CH1_YesNo, ByRef int CH2_YesNo,
                            ByRef int CH3_YesNo, ByRef int CH4_YesNo)
```

Description

Checks which of the 4 channels of a multi-channel attenuator are to be included in an attenuation sweep sequence.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	CH1_YesNo	Reference to a variable which will be updated with the status of CH1 (1 if it included in the sweep, 0 otherwise)
int	CH2_YesNo	Reference to a variable which will be updated with the status of CH2 (1 if it included in the sweep, 0 otherwise)
int	CH3_YesNo	Reference to a variable which will be updated with the status of CH3 (1 if it included in the sweep, 0 otherwise)
int	CH4_YesNo	Reference to a variable which will be updated with the status of CH4 (1 if it included in the sweep, 0 otherwise)

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_GetActiveChannels(CH1, CH2, CH3, CH4)
' Check which channels are to be included in the sweep
```

Visual C++

```
Status = MyPTE1->Sweep_GetActiveChannels(CH1, CH2, CH3, CH4);
// Check which channels are to be included in the sweep
```

Visual C#

```
Status=MyPTE1.Sweep_GetActiveChannels(ref(CH1),ref(CH2),ref(CH3),ref(CH4));
// Check which channels are to be included in the sweep
```

Matlab

```
[Status,CH1,CH2,CH3,CH4]=MyPTE1.Sweep_GetActiveChannels(CH1, CH2, CH3, CH4)
% Check which channels are to be included in the sweep
```

See Also

Sweep Mode - Multi-Channel - Set Active Channels

2.9 (o) - Sweep Mode - Multi-Channel - Set Channel Start Attenuation

Declaration

```
int Sweep_SetChannelStartAtt(int Channel, float Att)
```

Description

Sets the initial attenuation value for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)
float	Att	The starting attenuation value (dB) for the above channel during a multo-channel sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetChannelStartAtt(2, 15.75)
```

Visual C++

```
Status = MyPTE1->Sweep_SetChannelStartAtt(2, 15.75);
```

Visual C#

```
Status = MyPTE1.Sweep_SetChannelStartAtt(2, 15.75);
```

Matlab

```
Status = MyPTE1.Sweep_SetChannelStartAtt(2, 15.75)
```

See Also

Sweep Mode - Multi-Channel - Get Channel Start Attenuation

2.9 (p) - Sweep Mode - Multi-Channel - Get Channel Start Attenuation

Declaration

```
float Sweep_GetChannelStartAtt(int Channel)
```

Description

Gets the initial attenuation value for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)

Return Values

Data Type	Value	Description
float	Att	The starting attenuation value (dB) for the above channel during a multo-channel sweep

Examples

Visual Basic

```
Status = MyPTE1.Sweep_GetChannelStartAtt(2)
```

Visual C++

```
Status = MyPTE1->Sweep_GetChannelStartAtt(2);
```

Visual C#

```
Status = MyPTE1.Sweep_GetChannelStartAtt(2);
```

Matlab

```
Status = MyPTE1.Sweep_GetChannelStartAtt(2)
```

See Also

Sweep Mode - Multi-Channel - Set Channel Start Attenuation

2.9 (q) - Sweep Mode - Multi-Channel - Set Channel Stop Attenuation

Declaration

```
int Sweep_SetChannelStopAtt(int Channel, float Att)
```

Description

Sets the final attenuation value for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)
float	Att	The final attenuation value (dB) for the above channel during a multo-channel sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetChannelStopAtt(2, 90)
```

Visual C++

```
Status = MyPTE1->Sweep_SetChannelStopAtt(2, 90);
```

Visual C#

```
Status = MyPTE1.Sweep_SetChannelStopAtt(2, 90);
```

Matlab

```
Status = MyPTE1.Sweep_SetChannelStopAtt(2, 90)
```

See Also

Sweep Mode - Multi-Channel - Set Channel Stop Attenuation

2.9 (r) - Sweep Mode - Multi-Channel - Get Channel Stop Attenuation

Declaration

```
float Sweep_GetChannelStopAtt(int Channel)
```

Description

Gets the final attenuation value for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)

Return Values

Data Type	Value	Description
float	Att	The final attenuation value (dB) for the above channel during a multo-channel sweep

Examples

Visual Basic

```
Status = MyPTE1.Sweep_GetChannelStopAtt(2)
```

Visual C++

```
Status = MyPTE1->Sweep_GetChannelStopAtt(2);
```

Visual C#

```
Status = MyPTE1.Sweep_GetChannelStopAtt(2);
```

Matlab

```
Status = MyPTE1.Sweep_GetChannelStopAtt(2)
```

See Also

Sweep Mode - Multi-Channel - Set Channel Stop Attenuation

2.9 (s) - Sweep Mode - Multi-Channel - Set Channel Step Size

Declaration

```
int Sweep_SetChannelStepSize(int Channel, float Att)
```

Description

Sets the step size for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)
float	Att	The attenuation size (dB) for the above channel during a multo-channel sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetChannelStepSize(2, 0.5)
```

Visual C++

```
Status = MyPTE1->Sweep_SetChannelStepSize(2, 0.5);
```

Visual C#

```
Status = MyPTE1.Sweep_SetChannelStepSize(2, 0.5);
```

Matlab

```
Status = MyPTE1.Sweep_SetChannelStepSize(2, 0.5)
```

See Also

Sweep Mode - Multi-Channel - Get Channel Step Size

2.9 (t) - Sweep Mode - Multi-Channel - Get Channel Step Size

Declaration

```
float Sweep_GetChannelStepSize(int Channel)
```

Description

Gets the step size for a single channel within a multi-channel attenuator sweep.

Applies To

RC4DAT Series

Parameters

Data Type	Variable	Description
int	Channel	The channel number (1 to 4)

Return Values

Data Type	Value	Description
float	Att	The attenuation step size (dB) for the above channel during a multo-channel sweep

Examples

Visual Basic

```
Status = MyPTE1.Sweep_GetChannelStepSize(2)
```

Visual C++

```
Status = MyPTE1->Sweep_GetChannelStepSize(2);
```

Visual C#

```
Status = MyPTE1.Sweep_GetChannelStepSize(2);
```

Matlab

```
Status = MyPTE1.Sweep_GetChannelStepSize(2)
```

See Also

Sweep Mode - Multi-Channel - Set Channel Step Size

2.9 (u) - Sweep Mode - Turn On / Off

Declaration

```
int Sweep_SetMode(int On_Off)
```

Description

Enables or disables the attenuation sweep according to the previously configured parameters.

Notes:

- Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.
- An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

Requirements

Firmware version B1 or later.

Parameters

Data Type	Variable	Description
int	On_Off	Numeric value to enable/disable the hop sequence: 0 = Disable the hop sequence 1 = Enable the hop sequence

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Sweep_SetMode(1)
```

Visual C++

```
Status = MyPTE1->Sweep_SetMode(1);
```

Visual C#

```
Status = MyPTE1.Sweep_SetMode(1);
```

Matlab

```
Status = MyPTE1.Sweep_SetMode(1)
```

3 - Operating in a Linux Environment via USB

When connected by USB, the computer will recognize the programmable attenuator as a Human Interface Device (HID). In this mode of operation the following USB interrupt codes can be used. Alternatively, the RU series can be operated over a serial RS232 connection and the RC series can be operated over an Ethernet TCP/IP Network (please see Serial Control Using RS232 Communication and Ethernet Control over IP Networks for details).

To open a connection to Mini-Circuits programmable attenuators, the Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID: 0x20CE
- Programmable Attenuator Product ID: 0x23

Communication with the attenuator 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 attenuator.

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 - Interrupts - General Commands

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

	Description	Command Code (Byte 0)
a	Get Device Model Name	40
b	Get Device Serial Number	41
c	Send SCPI Command	1
d	Get Firmware	99
e	Set Attenuation	19
f	Read Attenuation	18

3.1 (a) - Get Device Model Name

Description

Returns the Mini-Circuits part number of the programmable attenuator.

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 RUDAT-6000-30 (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	82	ASCII character code for R
2	85	ASCII character code for U
3	68	ASCII character code for D
4	68	ASCII character code for A
5	84	ASCII character code for T
6	45	ASCII character code for -
7	54	ASCII character code for 6
8	48	ASCII character code for 0
9	48	ASCII character code for 0
10	48	ASCII character code for 0
11	45	ASCII character code for -
12	51	ASCII character code for 3
13	48	ASCII character code for 0
14	0	Zero value byte to indicate end of string

See Also

Get Device Serial Number
SCPI: Get Model Name

3.1 (b) - Get Device Serial Number

Description

Returns the serial number of the programmable attenuator.

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 programmable attenuator has serial number 11309220111 (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	49	ASCII character code for 1
12	0	Zero value byte to indicate end of string

See Also

Get Device Model Name
SCPI: Get Serial Number

3.1 (c) - Send SCPI Command

Description

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

Transmit Array

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

Returned Array

Byte	Data	Description
0	1	Interrupt code for Send SCPI Command
1 to (n-1)	SCPI Return String	The SCPI return string, one character per byte, represented as ASCII character codes
n	0	Zero value byte to indicate the end of the SCPI return string
(n+1) to 63	Not significant	“Don’t care” bytes, can be any value

Example

The SCPI command to request the model name is `:MN?` (see Get Model Name)

The ASCII character codes representing the 4 characters in this command should be sent in bytes 1 to 4 of the transmit array as follows (see the Programming Examples & Troubleshooting Guide for conversions between decimal, binary and ASCII characters):

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

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

Byte	Data	Description
0	1	Interrupt code for Send SCPI Command
1	82	ASCII character code for R
2	85	ASCII character code for U
3	68	ASCII character code for D
4	68	ASCII character code for A
5	84	ASCII character code for T
6	45	ASCII character code for -
7	54	ASCII character code for 6
8	48	ASCII character code for 0
9	48	ASCII character code for 0
10	48	ASCII character code for 0
11	45	ASCII character code for -
12	51	ASCII character code for 3
13	48	ASCII character code for 0
14	0	Zero value byte to indicate end of string

See Also

Summary of SCPI Commands / Queries

3.1 (d) - Get Firmware

Description

This function returns the internal firmware version of the programmable attenuator.

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 below returned array indicates that the system has firmware version "C3" (see the Programming Examples & Troubleshooting Guide for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	99	Interrupt code for Get Firmware
1	49	Not significant
2	77	Not significant
3	78	Not significant
4	63	Not significant
5	67	ASCII character code for C
6	51	ASCII character code for 3

See Also

SCPI: Get Firmware

3.1 (e) - Set Attenuation

Description

This function sets the RF attenuation level. The allowed attenuation range and precision is defined in the individual model datasheets.

Transmit Array

Byte	Data	Description
0	19	Interrupt code for Set Attenuation
1	Att_Byte0	First byte of the attenuation (dB) to set: Att_Byte0 = INTEGER(Attenuation)
2	Att_Byte1	Second byte of the attenuation (dB) to set: Att_Byte1 = (Attenuation - Att_Byte0) * 4
3	Channel_No	The attenuator channel to set (for single channel models the channel number is 1)
4 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	19	Interrupt code for Set Attenuation
1 - 63	Not significant	"Don't care" bytes, could be any value

Example

To set RUDAT-6000-90 to 43.75 dB, the transmit array is:

Byte	Data	Description
0	19	Interrupt code for Set Attenuation
1	43	Att_Byte0 = INTEGER(43.75) = 43
2	3	Att_Byte1 = (43.75 - 43) * 4 = 3
3	1	Channel 1 for single channel models

See Also

Read Attenuation

3.1 (f) - Read Attenuation

Description

This function returns the current attenuation setting.

Transmit Array

Byte	Data	Description
0	18	Interrupt code for Read Attenuation
1 - 63	Not significant	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	18	Interrupt code for Read Attenuation
1	CH1_Att_Byte0	First byte of the attenuation (dB) for channel 1 (note: single channel models only have channel 1)
2	CH1_Att_Byte1	Second byte of the attenuation (dB) for channel 1(note: single channel models only have channel 1) Attenuation = CH1_Att_Byte0 + (CH1_Att_Byte1 / 4)
3	CH2_Att_Byte0	First byte of the attenuation (dB) for channel 2 (note: not relevant to single channel models)
4	CH2_Att_Byte1	Second byte of the attenuation (dB) for channel 2 (note: not relevant to single channel models) Attenuation = CH2_Att_Byte0 + (CH2_Att_Byte1 / 4)
5	CH3_Att_Byte0	First byte of the attenuation (dB) for channel 3 (note: not relevant to single channel models)
6	CH3_Att_Byte1	Second byte of the attenuation (dB) for channel 3 (note: not relevant to single channel models) Attenuation = CH3_Att_Byte0 + (CH3_Att_Byte1 / 4)
7	CH4_Att_Byte0	First byte of the attenuation (dB) for channel 4 (note: not relevant to single channel models)
8	CH4_Att_Byte1	Second byte of the attenuation (dB) for channel 4 (note: not relevant to single channel models) Attenuation = CH4_Att_Byte0 + (CH4_Att_Byte1 / 4)
9 - 63	Not significant	"Don't care" bytes, could be any value

Examples

The following return array would indicate an attenuation of 75.75 dB for RCDAT-6000-90 (single channel model):

Byte	Data	Description
0	18	Interrupt code for Read Attenuation
1	75	
2	3	Attenuation = 75 + (3 / 4) = 75.75 dB

The following return array would indicate an attenuations of 75.75dB, 50.25 dB, 0 dBand 5 dB respectively for channels 1 to 4 of RC4DAT Series (4 channel model):

Byte	Data	Description
0	18	Interrupt code for Read Attenuation
1	75	
2	3	CH1 Attenuation = 75 + (3 / 4) = 75.75 dB
3	50	
4	1	CH2 Attenuation = 50 + (1 / 4) = 50.25 dB
5	0	
6	0	CH3 Attenuation = 0 + (0 / 4) = 0 dB
7	5	
8	0	CH4 Attenuation = 5 + (0 / 4) = 5 dB

See Also

Set Attenuation

3.2 - Interrupts - Ethernet Configuration Commands

These commands and queries apply to Mini-Circuits' RC series of programmable attenuators for configuring the Ethernet parameters.

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

3.2 (a) - Set Static IP Address

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Get Static IP Address
Reset Ethernet Configuration

3.2 (b) - Set Static Subnet Mask

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Get Static Subnet Mask
Reset Ethernet Configuration

3.2 (c) - Set Static Network Gateway

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Get Static Network Gateway
Reset Ethernet Configuration

3.2 (d) - Set HTTP Port

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

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

3.2 (e) - Set Telnet Port

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

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

3.2 (f) - Use Password

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Set Password
Get Password Status
Get Password
Reset Ethernet Configuration

3.2 (g) - Set Password

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use Password
Get Password Status
Get Password
Reset Ethernet Configuration

3.2 (h) - Use DHCP

Description

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

Transmit Array

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	207	Interrupt code for Use DHCP
2	DHCP_Mode	0 = DCHP disabled (static IP settings in use) 1 = DHCP enabled (default - dynamic IP in use)
3 - 63	Not significant	Any value

Returned Array

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

Example

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

Byte	Data	Description
0	250	Interrupt code for Set Ethernet Configuration
1	207	Interrupt code for Use DHCP
2	1	Enable DHCP

See Also

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

3.2 (i) - Get Static IP Address

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Set Static IP Address

3.2 (j) - Get Static Subnet Mask

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Set Static Subnet Mask

3.2 (k) - Get Static Network Gateway

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use DHCP
Set Static Network Gateway

3.2 (I) - Get HTTP Port

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Set HTTP Port
Set Telnet Port
Get Telnet Port

3.2 (m) - Get Telnet Port

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Set HTTP Port
Set Telnet Port
Get HTTP Port

3.2 (n) - Get Password Status

Description

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

Transmit Array

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

Returned Array

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

Example

The following returned array indicates that password protection is enabled:

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

See Also

Use Password
Set Password
Get Password

3.2 (o) - Get Password

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use Password
Set Password
Get Password Status

3.2 (p) - Get DHCP Status

Description

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

Transmit Array

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

Returned Array

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

Example

The following returned array indicates that DHCP is enabled:

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

See Also

Use DHCP
Get Dynamic Ethernet Configuration

3.2 (q) - Get Dynamic Ethernet Configuration

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

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

See Also

Use DHCP
Get DHCP Status

3.2 (r) - Get MAC Address

Description

Returns the MAC address of the programmable attenuator.

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Get Dynamic Ethernet Configuration

3.2 (s) - Reset Ethernet Configuration

Description

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

Transmit Array

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

Returned Array

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

4 - Ethernet Control over IP Networks

Mini-Circuits' RCDAT and RC4DAT attenuator series have an RJ45 connector for remote control over Ethernet TCP/IP networks. HTTP (Get/Post commands) and Telnet communication are supported. UDP transmission is also supported for discovering available attenuator devices on the network.

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

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

Notes:

1. The TCP/IP port must be included in every HTTP command to the attenuator unless the default port 80 is used
2. The password must be included in every HTTP command to the attenuator if password security is enabled
3. Port 23 is reserved for Telnet communication
4. The device draws DC power through the USB type B connector; this can be connected to a computer or the AC mains adapter

4.1 - Configuring Ethernet Settings

The IP address, gateway address, DHCP status and all other Ethernet parameters can be configured programmatically via the USB or Ethernet connections. The DLL file contains a series of functions for Ethernet configuration via USB (see DLL - Ethernet Configuration Functions) and the SCPI command-set contains a series of similar functions which can be sent via Ethernet or USB (see SCPI - Ethernet Configuration Commands).

After updating the Ethernet settings when connected by Ethernet, the device will reset and the connection will be temporarily lost. A new connection will subsequently need to be established using the updated Ethernet parameters. If a connection cannot be established (for example if an invalid or already used IP address was selected for the network) then the Ethernet parameters can be reset by connecting via the USB interface.

The standard Mini-Circuits GUI software also provides a simple interface to view and edit these settings via USB or Ethernet. The GUI can be downloaded from:

http://www.minicircuits.com/support/software_download.html

4.2 - Ethernet Communication Methodology

Communication over Ethernet can be accomplished using HTTP (Get/Post commands) or Telnet to send SCPI commands. The HTTP and Telnet protocols are both commonly supported and simple to implement in most programming languages. Any Internet browser can be used as a console/tester for HTTP control by typing the commands/queries directly into the address bar. The SCPI commands that can be sent to the programmable attenuators are defined in the SCPI Command Set section.

4.2 (a) - Setting Attenuation Using HTTP

The basic format of the HTTP command to set the attenuator is:

http://ADDRESS:PORT/PWD;COMMAND

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

Example 1:

http://192.168.100.100:800/PWD=123;SETATT=10.25

Explanation:
- The attenuator has IP address 192.168.100.100 and uses port 800
- Password security is enabled and set to “123”
- The command is to set the attenuation to 10.25dB (see below for the full explanation of all commands/queries)

Example 2:

http://10.10.10.10/SETATT=0

Explanation:
- The attenuator has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled
- The command is to set the attenuation to 0dB (see below for the full explanation of all commands/queries)

4.2 (b) - Querying Attenuator Properties Using HTTP

The basic format of the HTTP command to query the attenuator is:

http://ADDRESS:PORT/PWD;QUERY?

Where
- http:// is required
- ADDRESS = IP address (required)
- PORT = TCP/IP port (can be omitted if port 80 is used)
- PWD = Password (can be omitted if password is security is not enabled)
- QUERY? = Query to send to the attenuator

Example 1:

http://192.168.100.100:800/PWD=123;MN?

Explanation:
- The attenuator has IP address 192.168.100.100 and uses port 800
- Password security is enabled and set to “123”
- The query is to return the model name of the attenuator (see below for the full explanation of all commands/queries)

Example 2:

http://10.10.10.10/ATT?

Explanation:
- The attenuator has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled
- The query is to return the current attenuation setting (see below for the full explanation of all commands/queries)

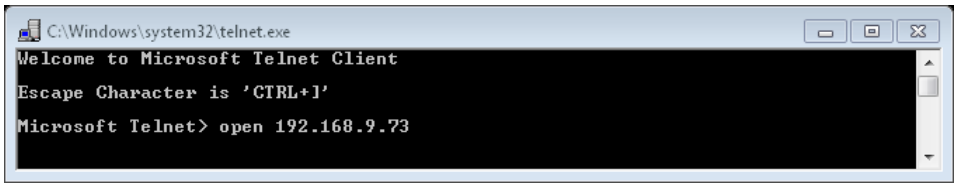
The device will return the result of the query as a string of ASCII characters.

4.2 (c) - Communication Using Telnet

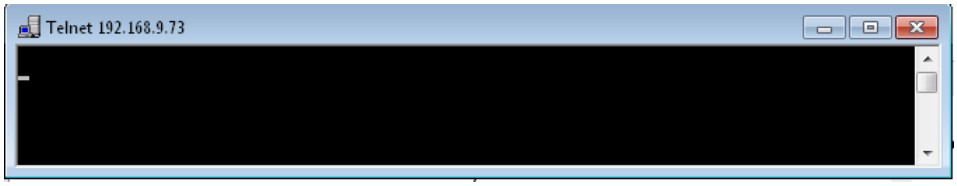
Communication with the device is started by creating a Telnet connection to the attenuator IP address. On successful connection the "line feed" character will be returned. If the attenuator has a password enabled then this must be sent as the first command after connection.

The full list of all commands and queries is detailed in the following sections. A basic example of the Telnet communication structure using the Windows Telnet Client is summarized below:

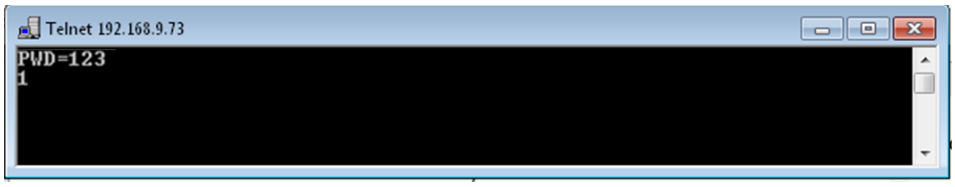
1) Set up Telnet connection to an attenuator with IP address 192.168.9.73



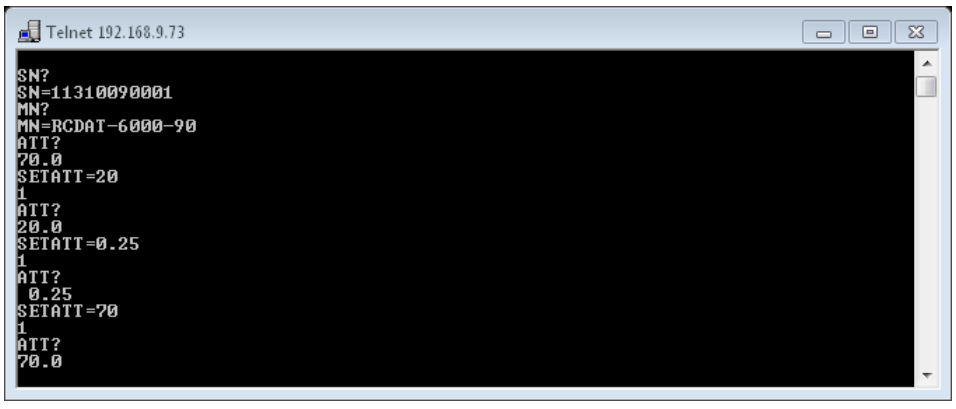
2) The "line feed" character is returned indicating the connection was successful:



3) The password (if enabled) must be sent as the first command; a return value of 1 indicates success:



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



4.3 - Device Discovery Using UDP

In addition to HTTP and Telnet, the RCDAT series of Ethernet controlled attenuators also provide limited support of the UDP protocol for the purpose of "device discovery." This allows a user to request the IP address and configuration of all Mini-Circuits RCDAT attenuators connected on the network; full control of those units is then accomplished using HTTP or Telnet, as detailed previously.

Alternatively, the IP configuration can be identified or changed by connecting the attenuator with the USB interface (see Configuring Ethernet Settings).

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

UDP Ports

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

Transmission

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

Receipt

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

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

Example

Sent Data:

MCLDAT?

Received Data:

Model Name: RCDAT-6000-60
Serial Number: 11302120001
IP Address=192.168.9.101 Port: 80
Subnet Mask=255.255.0.0
Network Gateway=192.168.9.0
Mac Address=D0-73-7F-82-D8-01

Model Name: RCDAT-6000-60
Serial Number: 11302120002
IP Address=192.168.9.102 Port: 80
Subnet Mask=255.255.0.0
Network Gateway=192.168.9.0
Mac Address=D0-73-7F-82-D8-02

Model Name: RCDAT-6000-90
Serial Number: 11302120003
IP Address=192.168.9.103 Port: 80
Subnet Mask=255.255.0.0
Network Gateway=192.168.9.0
Mac Address=D0-73-7F-82-D8-03

5 - SCPI Command Set

5.1 - Using SCPI Commands

SCPI (Standard Commands for Programmable Instruments) is a common method for controlling instrumentation products and a series of proprietary commands / queries are supported by Mini-Circuits' programmable attenuator models.

The SCPI commands are sent as an ASCII text string (up to 63 characters) in the below format:

```
:COMMAND:[value]:[suffix]
```

Where:

`COMMAND`	= the command/query to send
`[value]`	= the value (if applicable) to set
`[suffix]`	= the units (if applicable) that apply to the value

Commands can be sent in upper or lower case and the return value will be an ASCII text string. If an unrecognized command/query is received the sensor will return:

```
-99 Unrecognized Command. Model=[ModelName] SN=[SerialNumber]
```

These commands and queries can be sent using the DLL function Send SCPI Command when the system is connected via the USB interface in a Microsoft Windows environment, or using the USB interrupt commands on a Linux system (see Linux Send SCPI Command). In addition, SCPI commands can also be sent using HTTP get/post commands or Telnet over a TCP/IP network when the system is connected via the Ethernet RJ45 port (see Ethernet Control over IP Networks).

5.2 - Summary of SCPI Commands / Queries

5.2 (a) - SCPI - General Commands

	Description	Command/Query
a	Get Model Name	:MN?
b	Get Serial Number	:SN?
c	Set Start-Up Attenuation Mode	:STARTUPATT:INDICATOR:[Mode]
d	Get Start-Up Attenuation Mode	:STARTUPATT:INDICATOR?
e	Store Last Attenuation Value	:LASTATT:STORE:INITIATE
f	Set USB Address	:SETADD:[Address]
g	Get USB Address	:ADD?
h	Get Firmware Version	:FIRMWARE?

5.2 (b) - SCPI - ZVVA / RUDAT / RCDAT (Single Channel) Attenuation Functions

These functions apply to ZVVA, RUDAT & RCDAT models only

	Description	Command/Query
a	Set Attenuation	:SETATT=[Att]
b	Read Attenuation	:ATT?
c	Set Start-Up Attenuation Value	:STARTUPATT:VALUE:[Att]
d	Get Start-Up Attenuation Value	:STARTUPATT:VALUE?

5.2 (c) - SCPI - RC4DAT (Multi-Channel) Attenuation Functions

These functions apply to RC4DAT models only

	Description	Command/Query
a	Set Attenuation	:CHAN:[Channels]:SETATT:[Att]
b	Read Attenuation	:ATT?
c	Set Channel Start-Up Attenuation Value	:CHAN:[Channels]:STARTUPATT:VALUE:[Att]
d	Get Channel Start-Up Attenuation Value	:CHAN:[Channel]:STARTUPATT:VALUE?

5.2 (d) - SCPI - Attenuation Hopping Commands

These functions require firmware version B1 or later.

	Description	Command/Query
a	Hop - Set Number of Points	:HOP:POINTS:[NoOfPoints]
b	Hop - Get Number of Points	:HOP:POINTS?
c	Hop - Set Active Channels	:HOP:ACTIVECHANNELS:[CH_Value]
d	Hop - Get Active Channels	:HOP:ACTIVECHANNELS?
e	Hop - Set Sequence Direction	:HOP:DIRECTION:[Direction]
f	Hop - Get Sequence Direction	:HOP:DIRECTION?
g	Hop - Set Indexed Point	:HOP:POINT:[PointNo]
h	Hop - Get Indexed Point	:HOP:POINT?
i	Hop - Set Point Dwell Units	:HOP:DWELL_UNIT:[Units]
j	Hop - Set Point Dwell Time	:HOP:DWELL:[Time]
k	Hop - Get Point Dwell Time	:HOP:DWELL?
l	Hop - Set Point Attenuation	:HOP:ATT:[Att]
m	Hop - Set Channel Point Attenuation	:HOP:CHAN:[Channel]:ATT:[Att]
n	Hop - Get Point Attenuation	:HOP:ATT?
o	Hop - Get Channel Point Attenuation	:HOP:CHAN:[Channel]:ATT?
p	Hop - Turn On / Off	:HOP:MODE:[on_off]

5.2 (e) - SCPI - Attenuation Sweeping / Fading Commands

These functions require firmware version B1 or later.

	Description	Command/Query
a	Sweep - Set Direction	SWEEP:DIRECTION:[Direction]
b	Sweep - Get Direction	SWEEP:DIRECTION?
c	Sweep - Set Dwell Units	SWEEP:DWELL_UNIT:[Units]
d	Sweep - Set Dwell Time	SWEEP:DWELL:[Time]
e	Sweep - Get Dwell Time	SWEEP:DWELL?
f	Sweep - Set Active Channels	SWEEP:ACTIVECHANNELS:[CH_Value]
g	Sweep - Get Active Channels	SWEEP:ACTIVECHANNELS?
h	Sweep - Set Start Attenuation	SWEEP:START:[Att]
i	Sweep - Set Channel Start Attenuation	SWEEP:CHAN:[Channel]:START:[Att]
j	Sweep - Get Start Attenuation	SWEEP:START?
k	Sweep - Get Channel Start Attenuation	SWEEP:CHAN:[Channel]:START?
l	Sweep - Set Stop Attenuation	SWEEP:STOP:[Att]
m	Sweep - Set Channel Stop Attenuation	SWEEP:CHAN:[Channel]:STOP:[Att]
n	Sweep - Get Stop Attenuation	SWEEP:STOP?
o	Sweep - Get Channel Stop Attenuation	SWEEP:CHAN:[Channel]:STOP?
p	Sweep - Set Step Size	SWEEP:STEPSIZE:[Att]
q	Sweep - Set Channel Step Size	SWEEP:CHAN:[Channel]:STEPSIZE:[Att]
r	Sweep - Get Step Size	SWEEP:STEPSIZE?
s	Sweep - Get Channel Step Size	SWEEP:CHAN:[Channel]:STEPSIZE?
t	Sweep - Turn On / Off	SWEEP:MODE:[on_off]

5.2 (f) - SCPI - Ethernet Configuration Commands

These functions apply to RCDAT and RC4DAT models with firmware version C8 or later.

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

5.3 - SCPI - General Commands

5.3 (a) - Get Model Name

Description

Returns the full Mini-Circuits part number of the attenuator.

Command Syntax

```
:MN?
```

Return String

```
MN=[model]
```

Variable	Description
[model]	Full model name of the attenuator

Examples

String to Send	String Returned
:MN?	MN=RCDAT-6000-90

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

See Also

Get Serial Number

5.3 (b) - Get Serial Number

Description

Returns the serial number of the attenuator.

Command Syntax

```
:SN?
```

Return String

```
SN=[serial]
```

Variable	Description
[serial]	Serial number of the attenuator

Examples

String to Send	String Returned
:SN?	SN=11401010001

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

See Also

Get Model Name

5.3 (c) - Set Start-Up Attenuation Mode

Description

Sets the start-up mode to be used by the attenuator, this specifies how the initial attenuation value will be chosen when DC power is applied.

Requirements

Firmware version A6 or later

Command Syntax

```
:STARTUPATT:INDICATOR:[Mode]
```

Variable	Value	Description
[Mode]	L	Last Attenuation - The attenuation will be set to the same level as when the device was last powered off
	F	Fixed Attenuation - The attenuation will be set to a user-defined value
	N	Default - The attenuator will assume the factory default state (maximum attenuation)

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:STARTUPATT:INDICATOR:L	1
:STARTUPATT:INDICATOR:F	1
:STARTUPATT:INDICATOR:N	1

HTTP Implementation: `http://10.10.10.10/:STARTUPATT:INDICATOR:F`

See Also

Get Start-Up Attenuation Mode
Set Start-Up Attenuation Value
Get Start-Up Attenuation Value
Set Channel Start-Up Attenuation Value
Get Channel Start-Up Attenuation Value

5.3 (d) - Get Start-Up Attenuation Mode

Description

Returns the start-up mode currently in use by the attenuator; this specifies how the initial attenuation value will be chosen when DC power is applied.

Command Syntax

```
:STARTUPATT:INDICATOR?
```

Requirements

Firmware version A6 or later

Return String

```
[Mode]
```

Variable	Value	Description
[Mode]	L	Last Attenuation - The attenuation will be set to the same level as when the device was last powered off
	F	Fixed Attenuation - The attenuation will be set to a user-defined value
	N	Default - The attenuator will assume the factory default state (maximum attenuation)

Examples

String to Send	String Returned
:STARTUPATT:INDICATOR?	L
:STARTUPATT:INDICATOR?	F
:STARTUPATT:INDICATOR?	N

HTTP Implementation: `http://10.10.10.10/:STARTUPATT:INDICATOR?`

See Also

Set Start-Up Attenuation Mode
Set Start-Up Attenuation Value
Get Start-Up Attenuation Value
Set Channel Start-Up Attenuation Value
Get Channel Start-Up Attenuation Value

5.3 (e) - Store Last Attenuation Value

Description

Saves the final attenuation value to internal memory; necessary before powering off the device when it has been configured to load the last known attenuation on next power up. If this is not the last command before powering off in this mode, the attenuator will re-start with the maximum attenuation value when it is next powered on.

Requirements

Firmware version C3 or later

Command Syntax

```
:LASTATT:STORE:INITIATE
```

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:LASTATT:STORE:INITIATE	1

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

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode

5.3 (f) - Set USB Address

Description

Sets the device address to be used for USB communication (1 to 255).

Command Syntax

:SETADD:[Address]

Variable	Value	Description
[Address]	1-255	The USB address

Return String

[Status]

Variable	Value	Description
[Status]	0	Command failed or invalid address set
	1	Command completed successfully

Examples

String to Send	String Returned
:SETADD:15	1

HTTP Implementation: http://10.10.10.10/:SETADD:15

See Also

Get USB Address

5.3 (g) - Get USB Address

Description

Returns the device address to be used for USB communication.

Command Syntax

```
:ADD?
```

Return String

```
[Address]
```

Variable	Value	Description
[Address]	1-255	The USB address

Examples

String to Send	String Returned
:ADD?	15

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

See Also

Set USB Address

5.3 (h) - Get Firmware Version

Description

Returns the internal firmware version number.

Command Syntax

```
:FIRMWARE?
```

Return String

```
[Firmware]
```

Variable	Description
[Firmware]	The firmware version number

Examples

String to Send	String Returned
:FIRMWARE?	B1

HTTP Implementation: http://10.10.10.10/:FIRMWARE?

5.4 - SCPI - ZVVA / RUDAT / RCDAT (Single Channel) Attenuation Functions

These functions apply to ZVVA, RUDAT & RCDAT models only

5.4 (a) - Set Attenuation

Description

Sets the attenuation for a single channel attenuator.

Applies To

ZVVA, RCDAT and RUDAT models

Command Syntax

```
:SETATT=[Att]
```

Variable	Description
[Att]	The attenuation to set

Return String

```
[Status]
```

Variable	Value	Description
[Status]	0	Command failed or invalid attenuation set
	1	Command completed successfully
	2	Requested attenuation was higher than the allowed range, the attenuation was set to the device's maximum allowed value

Examples

String to Send	String Returned
:SETATT=130	2
:SETATT=12.75	1

HTTP Implementation: http://10.10.10.10/:SETATT=12.75

See Also

Read Attenuation

5.4 (b) - Read Attenuation

Description

Returns the current attenuation for a single channel attenuator.

Applies To

ZVVA, RCDAT and RUDAT models

Command Syntax

```
:ATT?
```

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The attenuation in dB

Examples

String to Send	String Returned
:ATT?	15.0 25.25 10.0 57.75

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

See Also

Set Attenuation

5.4 (c) - Set Start-Up Attenuation Value

Description

Sets the attenuation value to be loaded when a single channel is first powered up. Only applies when the attenuator's start-up mode is set to "Fixed Attenuation".

Applies To

ZVVA, RUDAT and RCDAT models with firmware version A6 or later

Command Syntax

```
:STARTUPATT:VALUE:[Att]
```

Variable	Description
[Att]	The initial attenuation level to be loaded on start-up

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:STARTUPATT:VALUE:12.75	1

HTTP Implementation: http://10.10.10.10/:STARTUPATT:VALUE:12.75

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Get Start-Up Attenuation Value

5.4 (d) - Get Start-Up Attenuation Value

Description

Gets the attenuation value to be loaded when a single channel is first powered up. Only applies when the attenuator's start-up mode is set to "Fixed Attenuation".

Applies To

ZVVA, RUDAT and RCDAT models with firmware version A6 or later

Command Syntax

```
:STARTUPATT:VALUE?
```

Return String

```
[Att]
```

Variable	Description
[Att]	The initial attenuation level to be loaded on start-up

Examples

String to Send	String Returned
:STARTUPATT:VALUE?	12.75

HTTP Implementation: http://10.10.10.10/:STARTUPATT:VALUE?

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Set Start-Up Attenuation Value

5.5 - SCPI - RC4DAT (Multi-Channel) Attenuation Functions

These functions apply to RC4DAT models only

5.5 (a) - Set Attenuation

Description

Sets the attenuation for a multi-channel programmable attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:CHAN:[Channels]:SETATT:[Att]
```

Variable	Description
[Channels]	The channel to set. Multiple channels can be sent by listing each channel number separated by a colon
[Att]	The attenuation to set for all channels listed above

Return String

```
[Status]
```

Variable	Value	Description
[Status]	0	Command failed or invalid attenuation set
	1	Command completed successfully

Examples

String to Send	String Returned
:CHAN:2:SETATT:15.75	1
:CHAN:1:3:4:SETATT:10	1

HTTP Implementation: http://10.10.10.10/:CHAN:2:SETATT:15.75

See Also

Read Attenuation

5.5 (b) - Read Attenuation

Description

Returns the current attenuation for all channels of a multi-channel attenuator.

Command Syntax

```
:ATT?
```

Return String

```
[CH1_Att] [CH2_Att] [CH3_Att] [CH4_Att]
```

Variable	Description
[CH1_Att]	Channel 1 attenuation (dB)
[CH2_Att]	Channel 2 attenuation (dB)
[CH3_Att]	Channel 3 attenuation (dB)
[CH4_Att]	Channel 4 attenuation (dB)

Examples

String to Send	String Returned
:ATT?	15.0 25.25 10.0 57.75

HTTP Implementation: http://10.10.10.10/:ATT?

See Also

Set Attenuation

5.5 (c) - Set Channel Start-Up Attenuation Value

Description

Sets the start up attenuation value for a single channel or channels within the multi-channel attenuator (the attenuation value to be loaded when DC power is applied). Only applies when the attenuator's start-up mode is set to "Fixed Attenuation".

Applies To

RC4DAT Series

Command Syntax

```
:CHAN:[Channels]:STARTUPATT:VALUE:[Att]
```

Variable	Description
[Channels]	The channel to set. Multiple channels can be sent by listing each channel number separated by a colon
[Att]	The initial attenuation level to be loaded on start-up

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:CHAN:1:STARTUPATT:VALUE:12.75	1
:CHAN:1:2:STARTUPATT:VALUE:12.75	1

HTTP Implementation: http://10.10.10.10/:CHAN:1:STARTUPATT:VALUE:12.75

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Get Channel Start-Up Attenuation Value

5.5 (d) - Get Channel Start-Up Attenuation Value

Description

Returns the start up attenuation value for a single channel within the multi-channel attenuator (the attenuation value to be loaded when DC power is applied). Only applies when the attenuator's start-up mode is set to "Fixed Attenuation".

Applies To

RC4DAT Series

Command Syntax

```
:CHAN:[Channel]:STARTUPATT:VALUE?
```

Variable	Description
[Channel]	The channel to query (1 to 4)

Return String

```
[Att]
```

Variable	Description
[Att]	The initial attenuation level to be loaded on start-up

Examples

String to Send	String Returned
:CHAN:2:STARTUPATT:VALUE?	12.75

HTTP Implementation: http://10.10.10.10/:CHAN:2:STARTUPATT:VALUE?

See Also

Set Start-Up Attenuation Mode
Get Start-Up Attenuation Mode
Set Channel Start-Up Attenuation Value

5.6 - SCPI - Attenuation Hopping Commands

These functions require firmware version B1 or later.

Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.

An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

5.6 (a) - Hop Mode - Set Number of Points

Description

Sets the number of points to be used in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:POINTS:[NoOfPoints]
```

Variable	Value	Description
[NoOfPoints]	1-100	The number of points to set in the hop sequence

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:POINTS:10	1

HTTP Implementation: http://10.10.10.10/:HOP:POINTS:10

See Also

Hop Mode - Get Number of Points
Hop Mode - Set Sequence Direction
Hop Mode - Get Sequence Direction

5.6 (b) - Hop Mode - Get Number of Points

Description

Returns the number of points to be used in the attenuation hop sequence.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:POINTS?
```

Return String

```
[NoOfPoints]
```

Variable	Value	Description
[NoOfPoints]	1-100	The number of points in the hop sequence

Examples

String to Send	String Returned
:HOP:POINTS?	10

HTTP Implementation: `http://10.10.10.10/:HOP:POINTS?`

See Also

Hop Mode - Set Number of Points
Hop Mode - Set Sequence Direction
Hop Mode - Get Sequence Direction

5.6 (c) - Hop Mode - Set Active Channels

Description

Sets which channels are to be included in the hop sequence for a multi-channel attenuator. This function does not apply to single channel models (ZVVA, RUDAT and RCDAT Series).

Applies To

RC4DAT Series

Command Syntax

```
:HOP:ACTIVECHANNELS:[CH_Value]
```

Variable	Description
[CH_Value]	Integer value indicating the combination of channels to be included in the HOP. Each channel is represented by an integer: Channel 1 = 1 Channel 2 = 2 Channel 3 = 4 Channel 4 = 8 CH_Value is the sum of the above integer values for the channels to be included in the hop sequence. For example, to include channels 1, 2 and 4 in the sequence: CH_Value = 1 + 2 + 8 = 11

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:ACTIVECHANNELS:11	1

HTTP Implementation: http://10.10.10.10/:HOP:ACTIVECHANNELS:11

See Also

Hop Mode - Get Active Channels

5.6 (d) - Hop Mode - Get Active Channels

Description

Returns which channels are to be included in the hop sequence for a multi-channel attenuator. This function does not apply to single channel models (ZVVA, RUDAT and RCDAT Series).

Applies To

RC4DAT Series

Command Syntax

```
:HOP:ACTIVECHANNELS?
```

Return String

```
[CH_Value]
```

Variable	Description
[CH_Value]	Integer value corresponding to a 4-bit binary string, with each bit representing a channel in the multi-channel attenuator: Bit 3 (MSB) = Channel 4 Bit 2 = Channel 3 Bit 1 = Channel 2 Bit 0 (LSB) = Channel 1 A bit value of 1 indicates the channel is included in the hop whereas a bit value of 0 indicates it is not. For example: CH_Value = 11 (decimal) = 1011 (binary) Channel 4 = 1 (included in hop) Channel 3 = 0 (included in hop) Channel 2 = 1 (not included in hop) Channel 1 = 1 (included in hop)

Examples

String to Send	String Returned
:HOP:ACTIVECHANNELS?	11

HTTP Implementation: http://10.10.10.10/:HOP:ACTIVECHANNELS?

See Also

Hop Mode - Set Active Channels

5.6 (e) - Hop Mode - Set Sequence Direction

Description

Sets the direction in which the attenuator will progress through the list of attenuation hops.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:DIRECTION:[Direction]
```

Variable	Value	Description
[Direction]	0	Forward - The list of attenuation hops will be loaded from index 1 to index n
	1	Backwards - The list of attenuation hops will be loaded from index n to index 1
	2	Bi-directionally - The list of attenuation hops will be loaded in the forward and then reverse directions

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:DIRECTION:0	1

HTTP Implementation: http://10.10.10.10/:HOP:DIRECTION:0

See Also

Hop Mode - Set Number of Points
Hop Mode - Get Number of Points
Hop Mode - Get Sequence Direction

5.6 (f) - Hop Mode - Get Sequence Direction

Description

Returns the direction in which the attenuator will progress through the list of attenuation hops.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:DIRECTION?
```

Return String

```
[Direction]
```

Variable	Value	Description
[Direction]	0	Forward - The list of attenuation hops will be loaded from index 1 to index n
	1	Backwards - The list of attenuation hops will be loaded from index n to index 1
	2	Bi-directionally - The list of attenuation hops will be loaded in the forward and then reverse directions

Examples

String to Send	String Returned
:HOP:DIRECTION?	0

HTTP Implementation: http://10.10.10.10/:HOP:DIRECTION?

See Also

Hop Mode - Set Number of Points
Hop Mode - Get Number of Points
Hop Mode - Set Sequence Direction

5.6 (g) - Hop Mode - Set Indexed Point

Description

Defines which point in the hop sequence is currently indexed, this allows the parameters for that point to be configured (attenuation value and dwell time).

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:POINT:[PointNo]
```

Variable	Description
[PointNo]	The index number of the point in the hop sequence (from 1 to the number of points configured)

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:POINT:3	1

HTTP Implementation: http://10.10.10.10/:HOP:POINT:3

See Also

Hop Mode - Get Indexed Point
Hop Mode - Set Point Attenuation
Hop Mode - Set Channel Point Attenuation
Hop Mode - Set Point Dwell Time Units
Hop Mode - Set Point Dwell Time

5.6 (h) - Hop Mode - Get Indexed Point

Description

Returns the number of the indexed attenuation point within the hop sequence.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:POINT?
```

Return String

```
[PointNo]
```

Variable	Description
[PointNo]	The index number of the point in the hop sequence (from 1 to the number of points configured)

Examples

String to Send	String Returned
:HOP:POINT?	3

HTTP Implementation: http://10.10.10.10/:HOP:POINT?

See Also

Hop Mode - Set Indexed Point
Hop Mode - Set Point Attenuation
Hop Mode - Set Channel Point Attenuation
Hop Mode - Set Point Dwell Time Units
Hop Mode - Set Point Dwell Time

5.6 (i) - Hop Mode - Set Point Dwell Time Units

Description

Sets the units to be used for the dwell time of the indexed point in the hop sequence.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:DWELL_UNIT:[Units]
```

Variable		Description
[Units]	U	Dwell time in microseconds (µs)
	M	Dwell time in milliseconds (ms)
	S	Dwell time in seconds (s)

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:DWELL_UNITS:U	1
:HOP:DWELL_UNITS:M	1
:HOP:DWELL_UNITS:S	1

HTTP Implementation: http://10.10.10.10/:HOP:DWELL_UNITS:U

See Also

Hop Mode - Set Point Dwell Time
Hop Mode - Get Point Dwell Time

5.6 (j) - Hop Mode - Set Point Dwell Time

Description

Sets the dwell time of the indexed point in the hop sequence. The dwell time units are defined separately.

Requirements

Firmware version B1 or later.

Command Syntax

:HOP:DWELL:[Time]

Variable	Description
[Time]	The dwell time of the indexed point

Return String

[Status]

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

Examples

String to Send	String Returned
:HOP:DWELL:650	1

HTTP Implementation: http://10.10.10.10/:HOP:DWELL:650

See Also

Hop Mode - Set Point Dwell Time Units
Hop Mode - Get Point Dwell Time

5.6 (k) - Hop Mode - Get Point Dwell Time

Description

Gets the dwell time (including the units) of the indexed point in the hop sequence.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:DWELL?
```

Return String

```
[Dwell] [Units]
```

Variable	Value	Description
[Dwell]		The dwell time of the indexed point
[Units]	uSec	Dwell time in microseconds (µs)
	mSec	Dwell time in milliseconds (ms)
	Sec	Dwell time in seconds (s)

Examples

String to Send	String Returned
:HOP:DWELL?	625 uSec
:HOP:DWELL?	50 mSec
:HOP:DWELL?	2 Sec

HTTP Implementation: http://10.10.10.10/:HOP:DWELL?

See Also

Hop Mode - Set Point Dwell Time Units
Hop Mode - Set Point Dwell Time

5.6 (I) - Hop Mode - Set Point Attenuation

Description

Sets the attenuation of the indexed hop point.

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:HOP:ATT:[Att]
```

Variable	Description
[Att]	The attenuation of the indexed hop point

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:ATT:75.5	1

HTTP Implementation: `http://10.10.10.10/:HOP:ATT:75.5`

See Also

Hop Mode - Set Channel Point Attenuation
Hop Mode - Get Point Attenuation

5.6 (m) - Hop Mode - Set Channel Point Attenuation

Description

Sets the attenuation value of the indexed hop point for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:HOP:CHAN:[Channel]:ATT:[Att]
```

Variable	Description
[Channel]	The channel to set
[Att]	The attenuation of the indexed hop point

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:ATT:75.5	1

HTTP Implementation: http://10.10.10.10/:HOP:ATT:75.5

See Also

Hop Mode - Set Point Attenuation
Hop Mode - Get Channel Point Attenuation

5.6 (n) - Hop Mode - Get Point Attenuation

Description

Returns the attenuation of the indexed hop point.

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:HOP:ATT?
```

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The attenuation of the indexed hop point

Examples

String to Send	String Returned
:HOP:ATT?	75.50

HTTP Implementation: `http://10.10.10.10/:HOP:ATT?`

See Also

Hop Mode - Set Point Attenuation
Hop Mode - Get Channel Point Attenuation

5.6 (o) - Hop Mode - Get Channel Point Attenuation

Description

Sets the attenuation value of the indexed hop point for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:HOP:CHAN:[Channel]:ATT?
```

Variable	Description
[Channel]	The channel to set (1 to 4)

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The attenuation of the indexed hop point

Examples

String to Send	String Returned
:HOP:ATT?	75.50
:HOP:CHAN:1:ATT?	75.50

HTTP Implementation: http://10.10.10.10/:HOP:ATT?

See Also

Hop Mode - Set Channel Point Attenuation
Hop Mode - Get Point Attenuation

5.6 (p) - Hop Mode - Turn On / Off

Description

Enables or disables the hop sequence according to the previously configured parameters.

Notes:

- Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.
- An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

Requirements

Firmware version B1 or later.

Command Syntax

```
:HOP:MODE:[On_Off]
```

Variable	Value	Description
[On_Off]	ON	Enable the hop sequence using the previously configured list of attenuation hops
	OFF	Disable the hop sequence

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:HOP:MODE:ON	1
:HOP:MODE:OFF	1

HTTP Implementation: http://10.10.10.10/:HOP:MODE:ON

5.7 - SCPI - Attenuation Sweeping / Fading Commands

These functions require firmware version B1 or later.

Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.

An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

5.7 (a) - Sweep Mode - Set Sweep Direction

Description

Sets the direction in which the attenuation level will sweep.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:DIRECTION:[Direction]
```

Variable	Value	Description
[Direction]	0	Forward - Sweep from "Start" to "Stop" value
	1	Backwards - Sweep from "Stop" to "Start" value
	2	Bi-directionally - Sweep in the forward and then reverse directions

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:DIRECTION:0	1

HTTP Implementation: `http://10.10.10.10/:SWEEP:DIRECTION:0`

See Also

Sweep Mode - Get Sweep Direction

5.7 (b) - Sweep Mode - Get Sweep Direction

Description

Returns the direction in which the attenuation level will sweep.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:DIRECTION?
```

Return String

```
[Direction]
```

Variable	Value	Description
[Direction]	0	Forward - Sweep from "Start" to "Stop" value
	1	Backwards - Sweep from "Stop" to "Start" value
	2	Bi-directionally - Sweep in the forward and then reverse directions

Examples

String to Send	String Returned
:SWEEP:DIRECTION?	0

HTTP Implementation: http://10.10.10.10/:SWEEP:DIRECTION?

See Also

Sweep Mode - Set Sweep Direction

5.7 (c) - Sweep Mode - Set Dwell Time Units

Description

Sets the units to be used for the sweep dwell time.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:DWELL_UNIT:[Units]
```

Variable		Description
[Units]	U	Dwell time in microseconds (µs)
	M	Dwell time in milliseconds (ms)
	S	Dwell time in seconds (s)

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:DWELL_UNITS:U	1
:SWEEP:DWELL_UNITS:M	1
:SWEEP:DWELL_UNITS:S	1

HTTP Implementation: http://10.10.10.10/:SWEEP:DWELL_UNITS:U

See Also

Sweep Mode - Set Point Dwell Time
Sweep Mode - Get Dwell Time

5.7 (d) - Sweep Mode - Set Dwell Time

Description

Sets the dwell time to be used for the sweep. The dwell time units are defined separately.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:DWELL:[Time]
```

Variable	Description
[Time]	The dwell time of the indexed point

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:DWELL:650	1

HTTP Implementation: http://10.10.10.10/:SWEEP:DWELL:650

See Also

Sweep Mode - Set Dwell Time Units
Sweep Mode - Get Dwell Time

5.7 (e) - Sweep Mode - Get Dwell Time

Description

Gets the dwell time (including the units) of the attenuation sweep.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:DWELL?
```

Return String

```
[Dwell] [Units]
```

Variable	Value	Description
[Dwell]		The dwell time of the sweep
[Units]	uSec	Dwell time in microseconds (µs)
	mSec	Dwell time in milliseconds (ms)
	Sec	Dwell time in seconds (s)

Examples

String to Send	String Returned
:SWEEP:DWELL?	625 uSec
:SWEEP:DWELL?	50 mSec
:SWEEP:DWELL?	2 Sec

HTTP Implementation: http://10.10.10.10/:SWEEP:DWELL?

See Also

Sweep Mode - Set Dwell Time Units
Sweep Mode - Set Dwell Time

5.7 (f) - Sweep Mode - Set Active Channels

Description

Sets which channels are to be included in the sweep for a multi-channel attenuator. This function does not apply to single channel models (ZVVA, RUDAT and RCDAT Series).

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:ACTIVECHANNELS:[CH_Value]
```

Variable	Description
[CH_Value]	Integer value indicating the combination of channels to be included in the sweep. Each channel is represented by an integer: Channel 1 = 1 Channel 2 = 2 Channel 3 = 4 Channel 4 = 8 CH_Value is the sum of the above integer values for the channels to be included in the sweep. For example, to include channels 1, 2 and 4 in the sequence: CH_Value = 1 + 2 + 8 = 11

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:ACTIVECHANNELS:11	1

HTTP Implementation: http://10.10.10.10/:SWEEP:ACTIVECHANNELS:11

See Also

Sweep Mode - Get Active Channels

5.7 (g) - Sweep Mode - Get Active Channels

Description

Returns which channels are to be included in the sweep for a multi-channel attenuator. This function does not apply to single channel models (ZVVA, RUDAT and RCDAT Series).

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:ACTIVECHANNELS?
```

Return String

```
[CH_Value]
```

Variable	Description
[CH_Value]	Integer value corresponding to a 4-bit binary string, with each bit representing a channel in the multi-channel attenuator: Bit 3 (MSB) = Channel 4 Bit 2 = Channel 3 Bit 1 = Channel 2 Bit 0 (LSB) = Channel 1 A bit value of 1 indicates the channel is included in the sweep whereas a bit value of 0 indicates it is not. For example: CH_Value = 11 (decimal) = 1011 (binary) Channel 4 = 1 (included in sweep) Channel 3 = 0 (included in sweep) Channel 2 = 1 (not included in sweep) Channel 1 = 1 (included in sweep)

Examples

String to Send	String Returned
:SWEEP:ACTIVECHANNELS?	11

HTTP Implementation: http://10.10.10.10/:SWEEP:ACTIVECHANNELS?

See Also

Sweep Mode - Set Active Channels

5.7 (h) - Sweep Mode - Set Start Attenuation

Description

Sets the first attenuation level to be loaded during the sweep.

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:SWEEP:START:[Att]
```

Variable	Description
[Att]	The initial attenuation level to set

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:START:0	1

HTTP Implementation: http://10.10.10.10/:SWEEP:START:0

See Also

Sweep Mode - Set Channel Start Attenuation
Sweep Mode - Get Start Attenuation
Sweep Mode - Set Stop Attenuation
Sweep Mode - Set Step Size

5.7 (i) - Sweep Mode - Set Channel Start Attenuation

Description

Sets the first attenuation level to be loaded during the sweep for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:CHAN:[Channel]:START:[Att]
```

Variable	Description
[Channel]	The channel to set
[Att]	The initial attenuation level to set

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:CHAN:1:START:0	1

HTTP Implementation: `http://10.10.10.10/:SWEEP:CHAN:1:START:0`

See Also

Sweep Mode - Set Start Attenuation
Sweep Mode - Get Channel Start Attenuation
Sweep Mode - Set Channel Stop Attenuation
Sweep Mode - Set Channel Step Size

5.7 (j) - Sweep Mode - Get Start Attenuation

Description

Returns the first attenuation level to be loaded during the sweep.

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:SWEEP:START?
```

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The initial attenuation level to be set during the sweep

Examples

String to Send	String Returned
:SWEEP:START?	0.0

HTTP Implementation: `http://10.10.10.10/:SWEEP:START?`

See Also

Sweep Mode - Set Start Attenuation
Sweep Mode - Get Channel Start Attenuation
Sweep Mode - Get Stop Attenuation
Sweep Mode - Get Step Size

5.7 (k) - Sweep Mode - Get Channel Start Attenuation

Description

Returns the first attenuation level to be loaded during the sweep for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:CHAN:[Channel]:START?
```

Variable	Description
[Channel]	The channel to query (1 to 4)

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The initial attenuation level to be set during the sweep

Examples

String to Send	String Returned
:SWEEP:CHAN:1:START?	0.0

HTTP Implementation: `http://10.10.10.10/:SWEEP:CHAN:1:START?`

See Also

Sweep Mode - Set Channel Start Attenuation
Sweep Mode - Get Start Attenuation
Sweep Mode - Get Channel Stop Attenuation
Sweep Mode - Get Channel Step Size

5.7 (I) - Sweep Mode - Set Stop Attenuation

Description

Sets the final attenuation level to be loaded during the sweep.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:START:[Att]
```

Variable	Description
[Att]	The final attenuation level to set

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:STOP:65.75	1

HTTP Implementation: http://10.10.10.10/:SWEEP:STOP:65.75

See Also

Sweep Mode - Get Stop Attenuation
Sweep Mode - Set Start Attenuation
Sweep Mode - Set Channel Stop Attenuation
Sweep Mode - Set Step Size

5.7 (m) - Sweep Mode - Set Channel Stop Attenuation

Description

Sets the final attenuation level to be loaded during the sweep for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:CHAN:[Channel]:STOP:[Att]
```

Variable	Description
[Channel]	The channel to set
[Att]	The final attenuation level to set

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:CHAN:1:STOP:90	1

HTTP Implementation: http://10.10.10.10/:SWEEP:CHAN:1:STOP:90

See Also

Sweep Mode - Set Channel Start Attenuation
Sweep Mode - Set Stop Attenuation
Sweep Mode - Get Channel Stop Attenuation
Sweep Mode - Set Channel Step Size

5.7 (n) - Sweep Mode - Get Stop Attenuation

Description

Returns the final attenuation level to be loaded during the sweep.

Requirements

Firmware version B1 or later.

Command Syntax

```
:SWEEP:STOP?
```

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The final attenuation level to be set during the sweep

Examples

String to Send	String Returned
:SWEEP:STOP?	65.75

HTTP Implementation: `http://10.10.10.10/:SWEEP:STOP?`

See Also

Sweep Mode - Set Stop Attenuation
Sweep Mode - Get Start Attenuation
Sweep Mode - Get Channel Stop Attenuation
Sweep Mode - Get Step Size

5.7 (o) - Sweep Mode - Get Channel Stop Attenuation

Description

Returns the final attenuation level to be loaded during the sweep for a specific channel within a multi-channel attenuator.

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:CHAN:[Channel]:STOP?
```

Variable	Description
[Channel]	The channel to query (1 to 4)

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The final attenuation level to be set during the sweep

Examples

String to Send	String Returned
:SWEEP:CHAN:1:STOP?	0.0

HTTP Implementation: `http://10.10.10.10/:SWEEP:STOP?`

See Also

Sweep Mode - Get Channel Start Attenuation
Sweep Mode - Set Channel Stop Attenuation
Sweep Mode - Get Stop Attenuation
Sweep Mode - Get Channel Step Size

5.7 (p) - Sweep Mode - Set Step Size

Description

Sets the attenuation step size that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:SWEEP:STEPSIZE:[Att]
```

Variable	Description
[Att]	The attenuation step size

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:STEPSIZE:0.5	1

HTTP Implementation: `http://10.10.10.10/:SWEEP:STEPSIZE:0.5`

See Also

Sweep Mode - Get Step Size
Sweep Mode - Set Start Attenuation
Sweep Mode - Set Stop Attenuation
Sweep Mode - Set Channel Step Size

5.7 (q) - Sweep Mode - Set Channel Step Size

Description

Sets the attenuation step size for a multi-channel attenuator that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

RC4DAT Series

Command Syntax

```
:SWEEP:CHAN:[Channel]:STEPSIZE:[Att]
```

Variable	Description
[Channel]	The channel to set
[Att]	The attenuation step size

Return String

```
[Status]
```

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

Examples

String to Send	String Returned
:SWEEP:CHAN:1:STEPSIZE:0.5	1

HTTP Implementation: http://10.10.10.10/:SWEEP:CHAN:1:STEPSIZE:0.5

See Also

Sweep Mode - Set Channel Start Attenuation
Sweep Mode - Set Channel Stop Attenuation
Sweep Mode - Set Step Size
Sweep Mode - Get Channel Step Size

5.7 (r) - Sweep Mode - Get Step Size

Description

Returns the attenuation step size that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:SWEEP:STEPSIZE?
```

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The attenuation step size

Examples

String to Send	String Returned
:SWEEP:STEPSIZE?	0.50

HTTP Implementation: http://10.10.10.10/:SWEEP:STEPSIZE?

See Also

Sweep Mode - Set Step Size
Sweep Mode - Get Start Attenuation
Sweep Mode - Get Stop Attenuation
Sweep Mode - Get Channel Step Size

5.7 (s) - Sweep Mode - Get Channel Step Size

Description

Returns the attenuation step size for a multi-channel attenuators that will be used to increment the attenuation from the start to stop levels (or decrement from stop to start if the sweep is running in the reverse direction).

Applies To

ZVVA, RUDAT and RCDAT models with firmware version B1 or later.

Command Syntax

```
:SWEEP:CHAN:[Channel]:STEPSIZE?
```

Variable	Description
[Channel]	The channel to query (1 to 4)

Return String

```
[Attenuation]
```

Variable	Description
[Attenuation]	The attenuation step size

Examples

String to Send	String Returned
:SWEEP:CHAN:1:STEPSIZE?	0.50

HTTP Implementation: http://10.10.10.10/:SWEEP:CHAN:1:STEPSIZE?

See Also

Sweep Mode - Get Channel Start Attenuation
Sweep Mode - Get Channel Stop Attenuation
Sweep Mode - Set Channel Step Size
Sweep Mode - Get Step Size

5.7 (t) - Sweep Mode - Turn On / Off

Description

Enables or disable the attenuation sweep sequence according to the parameters set.

Notes:

- Once an attenuation sequence is programmed and enabled, it is managed by the attenuator's internal microprocessor; this supports very fast sequences with minimum dwell times in the order of 600 µs. It is not possible to query any attenuator parameters whilst the sequence is active so any subsequent command / query to the device will disable the sequence.
- An alternative implementation method is to control the sequence and timing from your program, only sending "set attenuation" commands to the attenuator at the appropriate times. The advantage of this approach is that the program is able to query and keep track of the current attenuation state. The disadvantage is that the communication delays inherent in USB / Ethernet communication dictate a minimum dwell time in the order of milliseconds with this approach, rather than microseconds.

Requirements

Firmware version B1 or later.

Command Syntax

:SWEEP:MODE:[On_Off]

Variable	Value	Description
[On_Off]	ON	Enable the sweep sequence according to the previously configured parameters
	OFF	Disable the sweep sequence

Return String

[Status]

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

Examples

String to Send	String Returned
:SWEEP:MODE:ON	1
:SWEEP:MODE:OFF	1

HTTP Implementation: http://10.10.10.10/:SWEEP:MODE:ON

5.8 - SCPI - Ethernet Configuration Commands

These functions apply to RCDAT or RC4DAT models with firmware C8 or later.

5.8 (a) - Set Static IP Address

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

:ETHERNET:CONFIG:IP:[ip]

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

Return String

[status]

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

Examples

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

HTTP Implementation:

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

See Also

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

5.8 (b) - Get Static IP Address

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[ip]
```

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

Examples

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

HTTP Implementation:

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

See Also

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

5.8 (c) - Set Static Subnet Mask

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

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

See Also

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

5.8 (d) - Get Static Subnet Mask

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[mask]
```

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

Examples

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

HTTP Implementation:

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

See Also

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

5.8 (e) - Set Static Network Gateway

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Variable	Description
[gateway]	IP address of the network gateway

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:NG:192.168.100.1.0
```

See Also

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

5.8 (f) - Get Static Network Gateway

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[gateway]
```

Variable	Description
[gateway]	IP address of the network gateway

Examples

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

HTTP Implementation:

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

See Also

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

5.8 (g) - Set HTTP Port

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

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

See Also

Get HTTP Port
Set Telnet Port
Update Ethernet Settings

5.8 (h) - Get HTTP Port

Description

Gets the IP port to be used for HTTP communication.

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[port]
```

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

Examples

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

HTTP Implementation:

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

See Also

Set HTTP Port
Get Telnet Port

5.8 (i) - Set Telnet Port

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:TELNETPORT:21
```

See Also

Set HTTP Port
Get Telnet Port
Update Ethernet Settings

5.8 (j) - Get Telnet Port

Description

Gets the IP port to be used for Telnet communication.

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[port]
```

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

Examples

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

HTTP Implementation:

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

See Also

Get HTTP Port
Set Telnet Port

5.8 (k) - Set Password Requirement

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:PWDENABLED:1
```

See Also

Get Password Requirement
Set Password
Get Password
Update Ethernet Settings

5.8 (I) - Get Password Requirement

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[enabled]
```

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

Examples

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

HTTP Implementation:

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

See Also

Set Password Requirement
Set Password
Get Password

5.8 (m) - Set Password

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

```
http://10.10.10.10/:ETHERNET:CONFIG:PWD:PASS-123
```

See Also

Set Password Requirement
Get Password Requirement
Get Password
Update Ethernet Settings

5.8 (n) - Get Password

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[pwd]
```

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

Examples

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

HTTP Implementation:

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

See Also

Set Password Requirement
Get Password Requirement
Set Password

5.8 (o) - Set DHCP Status

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

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

Return String

```
[status]
```

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

Examples

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

HTTP Implementation:

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

See Also

Set Static IP Address
Get DHCP Status
Update Ethernet Settings

5.8 (p) - Get DHCP Status

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[enabled]
```

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

Examples

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

HTTP Implementation:

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

See Also

Set Static IP Address
Set DHCP Status
Get Current Ethernet Configuration

5.8 (q) - Get MAC Address

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[mac]
```

Variable	Description
[mac]	MAC address of the attenuator

Examples

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

HTTP Implementation:

http://10.10.10.10/:ETHERNET:CONFIG:MAC?

See Also

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

5.8 (r) - Get Current Ethernet Configuration

Description

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

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

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

Return String

```
[ip];[mask];[gateway]
```

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

Examples

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

HTTP Implementation:

http://10.10.10.10/:ETHERNET:CONFIG:LISTEN?

See Also

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

5.8 (s) - Update Ethernet Settings

Description

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

Note: If a connection cannot be established after the INIT command has been issued it may indicate an invalid configuration was created (for example a static IP address which clashes with another device on the network). The Ethernet settings can always be overwritten by connecting to the system using the USB connection.

Requirements

RCDAT or RC4DAT model with firmware version C8 or later.

Command Syntax

:ETHERNET:CONFIG:INIT

Return String

[status]

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

Examples

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

HTTP Implementation:

http://10.10.10.10/:ETHERNET:CONFIG:INIT

See Also

Get Current Ethernet Configuration

6 - Serial Control Using RS232 Communication

The Mini-Circuits ZVVA and RUDAT programmable attenuator series have a 9-pin D-Sub connector for serial RS232 communication. To create a connection to the programmable attenuator, the following settings should be used:

- Baud = 9600
- Parity = N
- Data_Bits = 8

The 9-pin D-Sub connector of the attenuator should be connected to the computer's RS232 port. The device draws DC power through the USB type B connector; this can be connected to a computer or the AC mains adapter.

Communication with the attenuator is based on sending and receiving ASCII data over the RS232 port. Each command must be followed by a Carriage Return character.

A worked example is included in the Programming Examples & Troubleshooting Guide.

6.1 - Summary of ASCII Commands

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

	Description	Command	Comments
a	Get Device Model Name	M\r\n	
b	Get Device Serial Number	S\r\n	
c	Set Attenuation	B[a]E\r\n	a = attenuation
d	Read Attenuation (Integer)	R\r\n	
e	Read Attenuation (Decimal)	A\r\n	
f	Send SCPI Command	P[c]\r\n	c = SCPI command

6.2 - Description of ASCII Commands

6.2 (a) - Get Device Model Name

This function determines the Mini-Circuits model name of the connected attenuator.

Command

M

Return Value

[mn]

Where:
[mn] = model name of the attenuator

Example

Send the text command "M\r\n".

The response will be of the format “RUDAT-6000-30”.

See Also

Get Device Serial Number

6.2 (b) - Get Device Serial Number

This function returns the serial number of the connected attenuator.

Command

S

Return Value

[sn]

Where:
[sn] = serial number of the attenuator

Example

Send the text command "S\r\n".

The response will be of the format “11301050025”.

See Also

Get Device Model Name

6.2 (c) - Set Attenuation

This function sets the RF attenuation level. The allowed attenuation range and precision is defined in the individual model datasheets.

Command

B[a]E

Where:
[a] = attenuation value to set

Return Value

ACK

Example

To set the RF attenuation to 20.25dB:

- [a] = 20.25

Send the text command “B20.25E\r\n”.

See Also

Read Attenuation (Integer)
Read Attenuation (Decimal)

6.2 (d) - Read Attenuation (Integer)

Returns the current RF attenuation value in terms of the number of attenuation steps set. To calculate the decimal attenuation value:

Attenuation = Returned Value x Minimum Step Size

Command

```
R
```

Return Value

```
[a]
```

Where:

[a] = attenuation divided by minimum step size

Example

Send the text command "R\r\n".

A response of “82” indicates the following attenuation values, depending on the model:

- Models with 0.1 dB step size:
  - Attenuation = 82 x 0.1 = 8.2 dB
- Models with 0.25 dB step size:
  - Attenuation = 82 x 0.25 = 20.5 dB
- Models with 0.5 dB step size:
  - Attenuation = 82 x 0.5 = 41 dB

See Also

Set Attenuation
Read Attenuation (Decimal)

6.2 (e) - Read Attenuation (Decimal)

This function returns the current RF attenuation setting precisely (including decimal places).

Command

A

Return Value

[a]

Where:
[a] = attenuation

Example

Send the text command "A\r\n".

The response would be in the format “20.25” to indicate the attenuation is set at 20.25dB.

See Also

Set Attenuation
Read Attenuation (Integer)

6.2 (f) - Send SCPI Command

This function sends a SCPI command to the programmable attenuator and collects the returned acknowledgement. SCPI (Standard Commands for Programmable Instruments) is a common method for communicating with and controlling instrumentation products and in this case provides access to an enhanced set of functions / operations for the attenuator.

Command

```
P[c]
```

Where:
[c] = SCPI command to send

Return Value

```
[r]
```

Where:
[r] = SCPI response

Example

To query the model name using SCPI:

- [c] = :MN?

Send the text command “P:MN?\r\n”.

The response would be in the format of "MN=RCDAT-6000-90" for model RCDAT-6000-60.

See Also

Summary of SCPI Commands / Queries