Test Solutions - Programming Manual

Signal Generators

SSG Series Signal Generators

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

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 signal generators.
Mini-Circuits offers support over a variety of operating systems, programming environments and third-party applications.

The signal generator 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 from:
https://www.minicircuits.com/softwaredownload/sg.html

For details and specifications of individual models please see:
https://www.minicircuits.com/WebStore/PortableTestEquipment.html?sub_cat=Signal%20Generators

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®

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

1.1 - Control Options

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

1. For Ethernet connected models, using HTTP or Telnet communication over an Ethernet connection (see Ethernet Control over IP Networks), which is largely independent of the operating system
2. Using the provided ActiveX or .Net API objects (DLL files) for USB control from a Windows operating system (see Operating in a Windows Environment via USB)
3. Using interrupt codes for USB control from Unix based operating systems (see Operating in a Linux Environment via USB)

2 - Operating in a Windows Environment via USB

2.1 - The DLL (Dynamic Link Library) Concept

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

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

Mini-Circuits' CD package provides DLL Objects designed to allow your own software application to interface with the functions of the Mini-Circuits signal generators, see ***Error! Reference source not found.***.

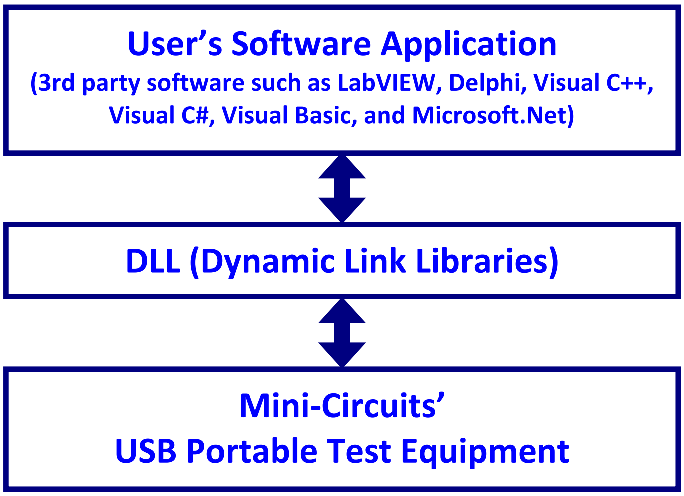

Fig 2.1-a: DLL Interface Concept

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' signal generators 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 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_gen.dll`
   For 64-bit Windows operating systems type (see Fig 2.1-e):
   `\WINDOWS\SysWOW64\Regsvr32 \WINDOWS\SysWOW64\mcl_gen.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

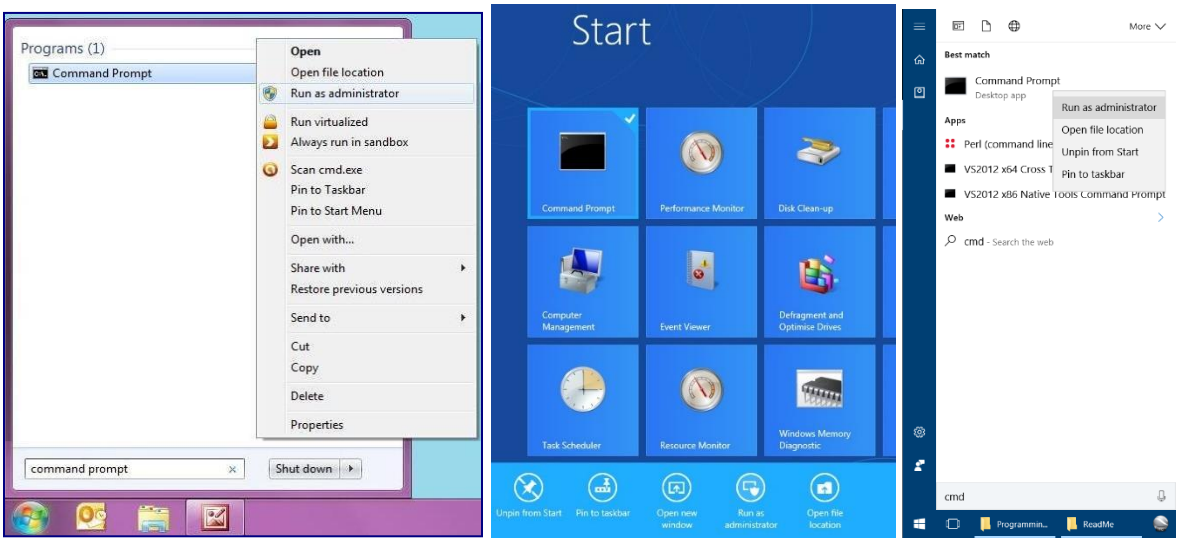

Fig 2.1-c: Opening the Command Prompt in Windows 7 (left), Windows 8 (middle) and Windows 10 (right)

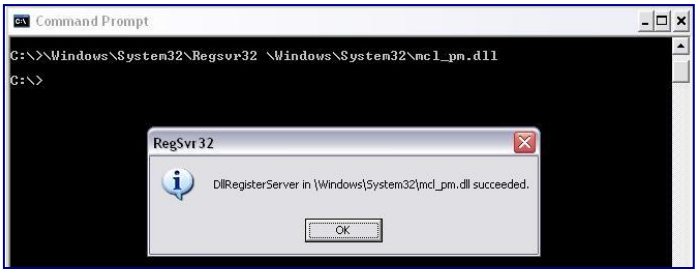

Fig 2.1-d: Registering the DLL in a 32-bit environment

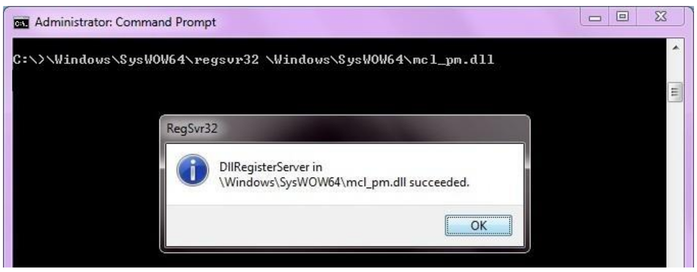

Fig 2.1-e: Registering the DLL in a 64-bit environment

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' signal generators 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 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 control class can be created for each signal generator 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_gen.dll)

Visual Basic

```
Public MyPTE1 As New MCL_Gen.USB_Gen
        ' Initialize new generator object, assign to MyPTE1
Public MyPTE2 As New MCL_Gen.USB_Gen
        ' Initialize new generator object, assign to MyPTE2
```

Visual C++

```
USB_Gen ^MyPTE1 = gcnew USB_Gen;
        // Initialize new generator instance, assign to MyPTE1
USB_Gen ^MyPTE2 = gcnew USB_Gen;
        // Initialize new generator instance, assign to MyPTE2
```

Visual C#

```
public MCL_Gen.USB_Gen MyPTE1 = new MCL_Gen.USB_Gen();
        // Initialize new generator instance, assign to MyPTE1
public MCL_Gen.USB_Gen MyPTE2 = new MCL_Gen.USB_Gen();
        // Initialize new generator instance, assign to MyPTE2
```

Matlab

```
MyPTE1=actxserver('MCL_Gen.USB_Gen')
        % Initialize new generator instance, assign to MyPTE1
MyPTE2=actxserver('MCL_Gen.USB_Gen')
        % Initialize new generator instance, assign to MyPTE2
```

2.2 (b) - Example Declarations using the .NET DLL (mcl_gen64.dll)

Visual Basic

```
Public MyPTE1 As New mcl_gen64.usb_gen
        ' Initialize new generator object, assign to MyPTE1
Public MyPTE2 As New mcl_gen64.usb_gen
        ' Initialize new generator object, assign to MyPTE2
```

Visual C++

```
usb_gen ^MyPTE1 = gcnew usb_gen;
        // Initialize new generator instance, assign to MyPTE1
usb_gen ^MyPTE2 = gcnew usb_gen;
        // Initialize new generator instance, assign to MyPTE2
```

Visual C#

```
public mcl_gen64.usb_gen MyPTE1 = new mcl_gen64.usb_gen();
        // Initialize new generator instance, assign to MyPTE1
public mcl_gen64.usb_gen MyPTE2 = new mcl_gen64.usb_gen();
        // Initialize new generator instance, assign to MyPTE2
```

Matlab

```
MCL_ATT=NET.addAssembly('C:\Windows\SysWOW64\mcl_gen64.dll')
MyPTE1=mcl_gen64.usb_gen      % Initialize new sig gen instance
MyPTE2=mcl_gen64.usb_gen      % Initialize new sig gen 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) - Connection Functions

a) Short Connect (Optional String SN)
b) Short ConnectByAddress (Optional Short Address)
c) Void Disconnect ()
d) Short Read_ModelName (String ModelName)
e) Short Read_SN (String SN)
f) Short Set_Address (Short Address)
g) Short Get_Address ()
h) Short Get_Available_SN_List (String SN_List)
i) Short Get_Available_Address_List (String Add_List)

2.3 (b) - CW (Continuous Wave) Output Functions

a) Short SetPowerON ()
b) Short SetPowerOFF ()
c) Short SetFreqAndPower (Double Fr, Float Pr, Short TriggerOut)
d) Short SetFreq (Double Fr, Short TriggerOut)
e) Short SetPower (Float Pr, Short TriggerOut)
f) Short GetGenStatus(Byte Locked, Short PowerIsOn, Double Fr , Float pr,
_ Short UNLEVELHigh, Short UNLEVELLow)
g) Short ExtRefDetected ()
h) String GetGenRef ()
i) Short GetTriggerIn_Status ()
j) Short Set_Trigger ()
k) Short Clear_Trigger ()
l) Float GetGenMaxFreq ()
m) Float GetGenMinFreq ()
n) Float GetGenStepFreq ()
o) Float GetGenMaxPower ()
p) Float GetGenMinPower ()

2.3 (c) - Status Functions

a) Float GetDeviceTemperature ()
b) Short Check_Connection ()
c) Short GetStatus ()
d) Short GetExtFirmware (Short A0, Short A1, Short A2, String Firmware)
e) Short GetFirmware ()

2.3 (d) - Calibration Functions

a) String GetCALReminderDate ()
b) Short SetCALReminderDate (String RemDate)
c) Short GetCALReminderDateIsRequired ()
d) Short SetCALReminderDateIsRequired (Short DateRequired)
e) Short GetCALReminderOTVal ()
f) Short SetCALReminderOTVal (Short OTVal)
g) Short GetCALReminderOTValIsRequired ()
h) Short SetCALReminderOTValIsRequired (Short OTRequired)
i) Short GetGenOperationTime ()

2.3 (e) - Modulation Functions

a) Short Set_PulseMode (Short T_OFF, Short T_ON, Short Tunit)
b) Short Set_PulseMode_Trigger (Short TriggerType, Short T_ON, Short Tunit)
c) Short Set_ExtPulseMod ()

2.3 (f) - Frequency Sweep Functions

a) Short FSweep_GetDirection ()
b) Short FSweep_GetDwell ()
c) Short FSweep_GetMaxDwell ()
d) Short FSweep_GetMinDwell ()
e) Float FSweep_GetPower ()
f) Double FSweep_GetStartFreq ()
g) Double FSweep_GetStopFreq ()
h) Double FSweep_GetStepSize ()
i) Short FSweep_GetTriggerIn ()
j) Short FSweep_GetTriggerOut ()
k) Short FSweep_SetDirection (Short SweepDirection)
l) Short FSweep_SetDwell (Short dwell_msec)
m) Short Fsweep_SetMode (Short onoff)
n) Float FSweep_SetPower (Float Pr)
o) Short FSweep_SetStartFreq (Double Fr)
p) Short FSweep_SetStopFreq (Double Fr)
q) Short FSweep_SetStepSize (Double Fr)
r) Short FSweep_SetTriggerIn (Short SweepTriggerIn)
s) Short FSweep_SetTriggerOut (Short SweepTriggerOut)

2.3 (g) - Power Sweep Functions

a) Short PSweep_GetDirection ()
b) Short PSweep_GetDwell ()
c) Short PSweep_GetMaxDwell ()
d) Short PSweep_GetMinDwell ()
e) Double PSweep_GetFreq ()
f) Float PSweep_GetStartPower ()
g) Float PSweep_GetStopPower ()
h) Float PSweep_GetStepSize ()
i) Short PSweep_GetTriggerIn ()
j) Short PSweep_GetTriggerOut ()
k) Short PSweep_SetDirection (Short SweepDirection)
l) Short PSweep_SetDwell (Short dwell_msec)
m) Short Psweep_SetMode (Short onoff)
n) Double PSweep_SetFreq (Float Fr)
o) Short PSweep_SetStartPower (Float Pr)
p) Short PSweep_SetStopPower (Float Pr)
q) Short PSweep_SetStepSize (Float Pr)
r) Short PSweep_SetTriggerIn (Short SweepTriggerIn)
s) Short PSweep_SetTriggerOut (Short SweepTriggerOut)

2.3 (h) - Frequency/Power Hop Functions

a) Short Hop_GetDirection ()
b) Short Hop_GetDwell ()
c) Short Hop_GetMaxDwell ()
d) Short Hop_GetMinDwell ()
e) Short Hop_GetNoOfPoints ()
f) Short Hop_GetMaxNoOfPoints ()
g) Short Hop_GetPoint (Short PointNo, Double HopFreq, float HopPower)
h) Short Hop_GetTriggerIn ()
i) Short Hop_GetTriggerOut ()
j) Short Hop_SetDirection (Short HopDirection)
k) Short Hop_SetDwell (Short dwell_msec)
l) Short Hop_SetMode (Short onoff)
m) Short Hop_SetNoOfPoints (Short HopNoOfPoints)
n) Short Hop_SetPoint (Short PointNo, Double HopFreq, Float HopPower)
o) Short Hop_SetTriggerIn (Short HopTriggerIn)
p) Short Hop_SetTriggerOut (Short HopTriggerOut)

2.3 (i) - Ethernet Configuration Functions

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

2.3 (j) - SCPI Communication Functions

a) Short SCPI_Query (String QuerySTR, ByRef String RetSTR)
b) Short SCPI_Command (String CommandSTR)

2.4 - Connection Functions

These common functions apply to all models in the Mini-Circuits SSG signal generator series unless otherwise stated; providing the means to identify, connect and disconnect the generator.

2.4 (a) - Connect to Signal Generator

Declaration

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

Description

This function is called to initialize the USB connection to a signal generator. If multiple generators are connected via USB 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 generator is no longer needed. The generator should be disconnected on completion of the program using the Disconnect function.

Parameters

Data Type	Variable	Description
String	SN	Optional. A string containing the serial number of the signal generator. Can be omitted if only one generator is connected but must be included otherwise.

Return Values

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

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 Signal Generator by Address
Read Serial Number of Signal Generator
Disconnect from Signal Generator

Mini-Circuits®

2.4 (b) - Connect to Signal Generator by Address

Declaration

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

Description

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

Parameters

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

Return Values

Data Type	Value	Description
Short	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 Signal Generator
Get Address of Signal generator
Disconnect from Signal Generator

2.4 (c) - Disconnect from Signal Generator

Declaration

```
Void Disconnect()
```

Description

This function is called to close the USB connection to the signal generator. It is strongly recommended that this function be used prior to ending the program. Failure to do so may result in a connection failure with the generator. Should this occur, terminate the program, unplug and shut down the signal generator before reconnecting and restarting.

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 Signal Generator
Connect to Signal Generator by Address

2.4 (d) - Read Model Name of Signal Generator

Declaration

```
Short Read_ModelName(String ModelName)
```

Description

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

Parameters

Data Type	Variable	Description
String	ModelName	Required. A string variable that will be updated with the Mini-Circuits model name for the signal generator.

Return Values

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

Examples

Visual Basic

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

Visual C++

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

Visual C#

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

Matlab

```
[status, ModelName]= MyPTE1.Read_ModelName(ModelName)
if status > 0
        h = msgbox('The connected generator is ', ModelName)
                % Display a message stating the model name
end
```

See Also

Read Serial Number of Signal Generator

2.4 (e) - Read Serial Number of Signal Generator

Declaration

```
Short Read_SN(String SN)
```

Description

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

Parameters

Data Type	Variable	Description
String	SN	Required. A string variable that will be updated with the Mini-Circuits serial number for the signal generator.

Return Values

Data Type	Value	Description
Short	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
        h = msgbox('The connected generator is ', SN)
                % Display a message stating the serial number
end
```

See Also

Connect to Signal Generator
Get List of Connected Serial Numbers

Mini-Circuits

2.4 (f) - Set Address of Signal Generator

Declaration

```
Short Set_Address(Short Address)
```

Description

This function allows the internal address of the connected signal generator to be changed from the factory default of 255. This allows the user to connect by a short address rather than serial number in future.

Parameters

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

Return Values

Data Type	Value	Description
Short	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

Connect to Signal Generator by Address
Get Address of Signal generator
Get List of Available Addresses

2.4 (g) - Get Address of Signal generator

Declaration

```
Short Get_Address()
```

Description

This function returns the address of the connected signal generator.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	1-255	Address of the signal generator

Examples

Visual Basic
```
addr = MyPTE1.Get_Address()
```
Visual C++
```
addr = MyPTE1->Get_Address();
```
Visual C#
```
addr = MyPTE1.Get_Address();
```
Matlab
```
addr = MyPTE1.Get_Address
```

See Also

Connect to Signal Generator by Address
Set Address of Signal Generator
Get List of Available Addresses

2.4 (h) - Get List of Connected Serial Numbers

Declaration

```
Short Get_Available_SN_List(String SN_List)
```

Description

This function takes a user defined variable and updates it with a list of serial numbers for all signal generators currently connected via USB.

Parameters

Data Type	Variable	Description
String	SN_List	Required. String variable which the function will update with a list of all available serial numbers, separated by a single space character, for example "11110001 11110002 11110003".

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	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
        % split the List into array of SN's
end
```

See Also

Get List of Available Addresses

2.4 (i) - Get List of Available Addresses

Declaration

```
Short Get_Available_Address_List(String Add_List)
```

Description

This function takes a user defined variable and updates it with a list of addresses of all signal generators currently connected via USB.

Parameters

Data Type	Variable	Description
String	Add_List	Required. String variable which the function will update with a list of addresses separated by a single space character, for example, "5 101 254 255"

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	Non zero	The number of signal generators 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
        % split the List into array of Addresses
end
```

See Also

Connect to Signal Generator by Address
Set Address of Signal Generator
Get Address of Signal generator

2.5 - CW (Continuous Wave) Output Functions

These common functions apply to all models in the Mini-Circuits SSG signal generator series unless otherwise stated; providing the basic means to configure the RF output.

2.5 (a) - Turn On RF Output

Declaration

```
Short Set_Power_ON()
```

Description

This function enables the RF output from the signal generator.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	Non zero	Command completed successfully

Examples

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

See Also

Turn Off RF Output
Set Output Frequency and Power
Set Output Frequency
Set Output Power

2.5 (b) - Turn Off RF Output

Declaration

```
Short Set_Power_OFF()
```

Description

This function disables the RF output from the signal generator.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Command failed
Short	Non zero	Command completed successfully

Examples

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

See Also

Turn On RF Output
Set Output Frequency and Power
Set Output Frequency
Set Output Power

2.5 (c) - Set Output Frequency and Power

Declaration

```
Short SetFreqAndPower(Double Fr, Float Pr, Short TriggerOut)
```

Description

This function sets the RF output frequency and power level of the signal generator and enables or disables the "trigger out" function.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The frequency in MHz.
Float	Pr	Required. The power in dBm.
Short	TriggerOut	Required. An integer variable to determine whether the "trigger out" function should be enabled. 1 enables trigger out, 0 disables it.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetFreqAndPower(Freq, Power, 0)
```
Visual C++
```
status = MyPTE1->SetFreqAndPower(Freq, Power, 0);
```
Visual C#
```
status = MyPTE1.SetFreqAndPower(Freq, Power, (short)0);
```
Matlab
```
status = MyPTE1.SetFreqAndPower(Freq, Power, 0)
```

See Also

Turn On RF Output
Turn Off RF Output
Set Output Frequency
Set Output Power
Get Generator Output Status

2.5 (d) - Set Output Frequency

Declaration

```
Short SetFreq(Double Fr, Short TriggerOut)
```

Description

This function sets the RF output frequency of the signal generator and enables or disables the "trigger out" function. The output power of the signal generator will not be changed.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The frequency in MHz.
Short	TriggerOut	Required. An integer variable to determine whether the "trigger out" function should be enabled. 1 enables trigger out, 0 disables it.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetFreq (Freq, 0)
```
Visual C++
```
status = MyPTE1->SetFreq (Freq, 0);
```
Visual C#
```
status = MyPTE1.SetFreq (Freq, (short)0);
```
Matlab
```
status = MyPTE1.SetFreq (Freq, 0)
```

See Also

Turn On RF Output
Turn Off RF Output
Set Output Frequency and Power
Set Output Power
Get Generator Output Status

2.5 (e) - Set Output Power

Declaration

```
Short SetPower(Float Pr, Short TriggerOut)
```

Description

This function sets the RF output power of the signal generator and enables or disables the "trigger out" function. The output frequency of the signal generator will not be changed.

Parameters

Data Type	Variable	Description
Float	Pr	Required. The power in dBm.
Short	TriggerOut	Required. An integer variable to determine whether the "trigger out" function should be enabled. 1 enables trigger out, 0 disables it.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetPower (Power, 0)
```
Visual C++
```
status = MyPTE1->SetPower (Power, 0);
```
Visual C#
```
status = MyPTE1.SetPower (Power, (short)0);
```
Matlab
```
status = MyPTE1.SetPower (Power, 0)
```

See Also

Turn On RF Output
Turn Off RF Output
Set Output Frequency and Power
Set Output Frequency
Get Generator Output Status

2.5 (f) - Get Generator Output Status

Declaration

```
Short GetGenStatus(Byte Locked, Short PowerIsOn, Double Fr, Float Pr,
                          Short UnLevelHigh, Short UnLevelLow)
```

Description

This function returns the current status of the signal generator RF output in a series of user defined variables. The following parameters are checked:

- Generator lock status (locked/unlocked)
- RF output status (on/off)
- Current output frequency
- Current output power
- Current output power relative to user requested level

Parameters

Data Type	Variable	Description
Byte	Locked	Required. User defined variable which will be set to 1 if the frequency is locked or 0 otherwise.
Short	PowerIsOn	Required. User defined variable which will be set to 1 if the RF output power is enabled or 0 otherwise.
Double	Fr	Required. User defined variable which will be updated with the generator output frequency in MHz.
Float	Pr	Required. User defined variable which will be updated with the generator output power in dBm.
Short	UnLevelHigh	Required. User defined variable that will be set to 1 if the user requested a higher power level than the generator can achieve. The variable is set to 0 if the output power is at the correct level. See model datasheets for output power specifications.
Short	UnLevelLow	Required. User defined variable that will be set to 1 if the user requested a lower power level than the generator can achieve. The variable is set to 0 if the output power is at the correct level. See model datasheets for output power specifications.

Return Values

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

Examples

Visual Basic

```
status = MyPTE1.GetGenStatus(lock, PowerIsOn, freq, power, UNLEVELHigh,
                                                                _ UNLEVELLow)
```

Visual C++

```
status = MyPTE1->GetGenStatus(lock, PowerIsOn, freq, power, UNLEVELHigh,
                                                               _ UNLEVELLow);
```

Visual C#

```
status = MyPTE1.GetGenStatus(ref(lock), ref(PowerIsOn), ref(freq),
                              _ ref(power), ref(UNLEVELHigh), ref(UNLEVELLow));
```

Matlab

```
[status, lock, PowerIsOn, freq, power, UNLEVELHigh, UNLEVELLow] =
  _ MyPTE1.GetGenStatus(lock, PowerIsOn, freq, power, UNLEVELHigh, UNLEVELLow)
```

See Also

Turn On RF Output
Turn Off RF Output
Set Output Frequency and Power
Set Output Frequency
Set Output Power

2.5 (g) - Check External Reference

Declaration

```
Short ExtRefDetected()
```

Description

This function checks whether an external 10MHz reference is connected to the generator. An external reference will be used automatically if it is present, otherwise the internal reference is used. The generator will assume that any signal on the Ref In port is a valid 10MHz source.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	External reference is not connected
	1	External reference is connected

Examples

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

See Also

Get Reference Source

2.5 (h) - Get Reference Source

Declaration

```
String GetGenRef()
```

Description

This function reports the whether the generator is currently operating with an external or internal reference source.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
String	INT	Generator is using the internal reference
	EXT	Generator is using an external reference source

Examples

Visual Basic
```
Ref = MyPTE1.GetGenRef
```
Visual C++
```
Ref = MyPTE1->GetGenRef();
```
Visual C#
```
Ref = MyPTE1.GetGenRef();
```
Matlab
```
Ref = MyPTE1.GetGenRef
```

See Also

Check External Reference

2.5 (i) - Get Trigger In Status

Declaration

```
Short GetTriggerIn_Status()
```

Description

This function indicates whether the generator's trigger input is at logic level low or high.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Trigger input is at logic low
	1	Trigger input is at logic high

Examples

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

See Also

Set Trigger As Input
Set Trigger As Output
Set Trigger Out

2.5 (j) - Set Trigger Out

Declaration

```
Short SetTrigger()
```

Description

This function sets the generator’s trigger output to logic high.

Parameters

Data Type	Variable	Description
None		

Return Values

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

Examples

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

See Also

Get Trigger In Status
Set Trigger As Input
Set Trigger As Output

2.5 (k) - Clear Trigger

Declaration

```
Short Clear_Trigger()
```

Description

This function clears the generator's trigger output (resets to logic low).

Parameters

Data Type	Variable	Description
None		

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Clear_Trigger()
```

Visual C++

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

Visual C#

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

Matlab

```
status = MyPTE1.Clear_Trigger()
```

See Also

Set Trigger Out

2.5 (l) - Get Generator Maximum Frequency Spec

Declaration

```
Float GetGenMaxFreq()
```

Description

This function reports the maximum output frequency in MHz that the generator is capable of providing.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Frequency	Maximum output frequency in MHz

Examples

Visual Basic
```
Freq = MyPTE1.GetGenMaxFreq
```
Visual C++
```
Freq = MyPTE1->GetGenMaxFreq();
```
Visual C#
```
Freq = MyPTE1.GetGenMaxFreq();
```
Matlab
```
Freq = MyPTE1.GetGenMaxFreq
```

See Also

Get Generator Minimum Frequency Spec
Get Generator Step Size Spec
Get Generator Maximum Power Spec
Get Generator Minimum Power Spec

2.5 (m) - Get Generator Minimum Frequency Spec

Declaration

```
Float GetGenMinFreq()
```

Description

This function reports the minimum output frequency in MHz that the generator is capable of providing.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Frequency	Minimum output frequency in MHz

Examples

Visual Basic

```
Freq = MyPTE1.GetGenMinFreq
```

Visual C++

```
Freq = MyPTE1->GetGenMinFreq();
```

Visual C#

```
Freq = MyPTE1.GetGenMinFreq();
```

Matlab

```
Freq = MyPTE1.GetGenMinFreq
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Step Size Spec
Get Generator Maximum Power Spec
Get Generator Minimum Power Spec

2.5 (n) - Get Generator Step Size Spec

Declaration

```
Float GetGenStepFreq()
```

Description

This function reports the generator's minimum step size in KHz.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Frequency	Generator step size in KHz

Examples

Visual Basic
```
Freq = MyPTE1.GetGenStepFreq
```
Visual C++
```
Freq = MyPTE1->GetGenStepFreq();
```
Visual C#
```
Freq = MyPTE1.GetGenStepFreq();
```
Matlab
```
Freq = MyPTE1.GetGenStepFreq
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Get Generator Maximum Power Spec
Get Generator Minimum Power Spec

2.5 (o) - Get Generator Maximum Power Spec

Declaration

```
Float GetGenMaxPower()
```

Description

This function reports the maximum output power specification in dBm for the active generator. The maximum output power achievable by the generator is guaranteed to be at least as high as this specified level across the full operating frequency and will be even higher in some frequency bands.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Power	Maximum output power in dBm

Example

Visual Basic
```
Power = MyPTE1.GetGenMaxPower
```
Visual C++
```
Power = MyPTE1->GetGenMaxPower();
```
Visual C#
```
Power = MyPTE1.GetGenMaxPower();
```
Matlab
```
Power = MyPTE1.GetGenMaxPower
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Get Generator Step Size Spec
Get Generator Minimum Power Spec

2.5 (p) - Get Generator Minimum Power Spec

Declaration

```
Float GetGenMinPower()
```

Description

This function reports the minimum output power specification in dBm for the active generator. The minimum output power achievable by the generator is guaranteed to be at least as low as this specified level across the full operating frequency and will be even lower in some frequency bands.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Power	Minimum output power in dBm

Example

Visual Basic
```
Power = MyPTE1.GetGenMinPower
```
Visual C++
```
Power = MyPTE1->GetGenMinPower();
```
Visual C#
```
Power = MyPTE1.GetGenMinPower();
```
Matlab
```
Power = MyPTE1.GetGenMinPower
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Get Generator Step Size Spec
Get Generator Maximum Power Spec

2.6 - Status Functions

These basic reporting functions apply to all models in the Mini-Circuits SSG signal generator series unless otherwise stated.

2.6 (a) - Get Temperature of Signal Generator

Declaration

```
Float GetDeviceTemperature()
```

Description

This function returns the internal temperature of the signal generator.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	Temperature	The device internal temperature in degrees Celsius

Example

Visual Basic

```
MsgBox ("Temperature is " & MyPTE1.GetDeviceTemperature(2))
        ' Display a message box with the device temperature
```

Visual C++

```
MessageBox::Show("Temperature is " + MyPTE1->GetDeviceTemperature(2));
        // Display a message box with the device temperature
```

Visual C#

```
MessageBox.Show("Temperature is " + MyPTE1.GetDeviceTemperature(2));
        // Display a message box with the device temperature
```

Matlab

```
[temp, status] = MyPTE1.GetDeviceTemperature(2)
h = msgbox('Temperature is ', temp)
        % Display a message box with the device temperature
```

2.6 (b) - Check Connection

Declaration

```
Short Check_Connection()
```

Description

This function checks whether the USB connection to the signal generator is active.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	No connection
Short	1	USB connection to signal generator is active

Example

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

See Also

Connect to Signal Generator
Connect to Signal Generator by Address
Disconnect from Signal Generator

2.6 (c) - Get Status (Antiquated)

Declaration

```
Short GetStatus()
```

Description

This function is antiquated, please use Check Connection instead. GetStatus checks whether the USB connection to the signal generator is active.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	No connection
Short	1	USB connection to signal generator is active

Example

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

See Also

Check Connection

2.6 (d) - Get Firmware

Declaration

```
Short GetExtFirmware(Short A0, Short A1, Short A2, String Firmware)
```

Description

This function returns the internal firmware version of the signal generator along with three reserved variables for factory use.

Parameters

Data Type	Variable	Description
Short	A0	Required. User defined variable for factory use only.
Short	A1	Required. User defined variable for factory use only.
Short	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
Short	0	Command failed
Short	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
        h = msgbox('Firmware version is ', Firmware)
end
```

2.6 (e) - Get Firmware Version (Antiquated)

Declaration

```
Short GetFirmware()
```

Description

This function is antiquated, GetExtFirmware should be used instead. The function returns a numeric value corresponding to the internal firmware version of the generator.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short		Version number of the internal signal generator firmware

Example

Visual Basic
```
FW = MyPTE1.GetFirmware()
```
Visual C++
```
FW = MyPTE1->GetFirmware();
```
Visual C#
```
FW = MyPTE1.GetFirmware();
```
Matlab
```
FW = MyPTE1.GetFirmware()
```

See Also

Get Firmware

2.7 - Calibration Functions

These functions provide access to calibration information for the signal generators.

2.7 (a) - Get Calibration Reminder Date

Declaration

```
String GetCalReminderDate()
```

Description

Returns the date on which a calibration reminder should be displayed.

Return Values

Data Type	Value	Description
String	RemDate	A string representing the calibration reminder date, in the format "yymmdd", eg: "140625" for 25th June 2014.

Examples

Visual Basic
```
RemDate = MyPTE1.GetCalReminderDate
```
Visual C++
```
RemDate = MyPTE1->GetCalReminderDate();
```
Visual C#
```
RemDate = MyPTE1.GetCalReminderDate();
```
Matlab
```
RemDate = MyPTE1.GetCalReminderDate
```

See Also

Set Calibration Reminder Date
Get Calibration Reminder Date Status
Set Calibration Reminder Date Status

2.7 (b) - Set Calibration Reminder Date

Declaration

```
Short SetCalReminderDate(String RemDate)
```

Description

Sets the date on which a calibration reminder should be displayed.

Parameters

Data Type	Value	Description
String	RemDate	A string representing the calibration reminder date, in the format "yymmdd", eg: "140625" for 25th June 2014.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetCalReminderDate("140625")
```
Visual C++
```
status = MyPTE1->SetCalReminderDate("140625");
```
Visual C#
```
status = MyPTE1.SetCalReminderDate("140625");
```
Matlab
```
status = MyPTE1.SetCalReminderDate("140625")
```

See Also

Get Calibration Reminder Date
Get Calibration Reminder Date Status
Set Calibration Reminder Date Status

2.7 (c) - Get Calibration Reminder Date Status

Declaration

```
Short GetCalReminderDateIsRequired()
```

Description

Indicates whether a calibration reminder should be displayed from a given date.

Return Values

Data Type	Value	Description
Short	DateRequired	Numeric value indicating whether a calibration reminder date is required: 0 = Reminder date not required 1 = Reminder date required

Examples

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

See Also

Get Calibration Reminder Date
Set Calibration Reminder Date
Set Calibration Reminder Date Status

2.7 (d) - Set Calibration Reminder Date Status

Declaration

```
Short SetCALReminderDateIsRequired(Short DateRequired)
```

Description

Sets whether a calibration reminder date is required.

Parameters

Data Type	Value	Description
Short	DateRequired	Numeric value indicating whether a calibration reminder date is required: 0 = Reminder date not required 1 = Reminder date required

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetCALReminderDateIsRequired(1)
```
Visual C++
```
status = MyPTE1->SetCALReminderDateIsRequired(1);
```
Visual C#
```
status = MyPTE1.SetCALReminderDateIsRequired(1);
```
Matlab
```
status = MyPTE1.SetCALReminderDateIsRequired(1)
```

See Also

Get Calibration Reminder Date
Set Calibration Reminder Date
Get Calibration Reminder Date Status

2.7 (e) - Get Calibration Reminder Operating Time

Declaration

```
Short GetCALReminderOTVal()
```

Description

Returns the operating time after which a calibration reminder should be displayed. The operating time is measured in hours from the last calibration.

Return Values

Data Type	Value	Description
Short	OpTime	The operating time in hours at which a calibration reminder is due

Examples

Visual Basic
```
OpTime = MyPTE1.GetCALReminderOTVal
```
Visual C++
```
OpTime = MyPTE1->GetCALReminderOTVal();
```
Visual C#
```
OpTime = MyPTE1.GetCALReminderOTVal();
```
Matlab
```
OpTime = MyPTE1.GetCALReminderOTVal
```

See Also

Set Calibration Reminder Operating Time
Get Calibration Reminder Operating Time Status
Set Calibration Reminder Operating Time Status
Get Generator Operating Time

2.7 (f) - Set Calibration Reminder Operating Time

Declaration

```
Short SetCALReminderOTVal(Short OpTime)
```

Description

Sets the operating time after which a calibration reminder should be displayed. The operating time is measured in hours from the last calibration.

Parameters

Data Type	Value	Description
Short	OpTime	The operating time in hours at which a calibration reminder is due

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetCALReminderOTVal(120)
```
Visual C++
```
status = MyPTE1->SetCALReminderOTVal(120);
```
Visual C#
```
status = MyPTE1.SetCALReminderOTVal(120);
```
Matlab
```
status = MyPTE1.SetCALReminderOTVal(120)
```

See Also

Get Calibration Reminder Operating Time
Get Calibration Reminder Operating Time Status
Set Calibration Reminder Operating Time Status
Get Generator Operating Time

2.7 (g) - Get Calibration Reminder Operating Time Status

Declaration

```
Short GetCALReminderOTValIsRequired()
```

Description

Indicates whether a calibration reminder should be displayed after a specified operating time (in hours from last calibration).

Return Values

Data Type	Value	Description
Short	OTRequired	Numeric value indicating whether a calibration reminder is required after a specified operating time: 0 = Operating time reminder not required 1 = Operating time reminder required

Examples

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

See Also

Get Calibration Reminder Operating Time
Set Calibration Reminder Operating Time
Set Calibration Reminder Operating Time Status
Get Generator Operating Time

2.7 (h) - Set Calibration Reminder Operating Time Status

Declaration

```
Short SetCALReminderOTValIsRequired(Short OTRequired)
```

Description

Sets whether a calibration reminder should be displayed after a specified operating time.

Parameters

Data Type	Value	Description
Short	OTRequired	Numeric value indicating whether a calibration reminder is required after a specified operating time: 0 = Operating time reminder not required 1 = Operating time reminder required

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetCALReminderOTValIsRequired(1)
```
Visual C++
```
status = MyPTE1->SetCALReminderOTValIsRequired(1);
```
Visual C#
```
status = MyPTE1.SetCALReminderOTValIsRequired(1);
```
Matlab
```
status = MyPTE1.SetCALReminderOTValIsRequired(1)
```

See Also

Get Calibration Reminder Operating Time
Set Calibration Reminder Operating Time
Get Calibration Reminder Operating Time Status
Get Generator Operating Time

2.7 (i) - Get Generator Operating Time

Declaration

```
Short GetGenOperationTime()
```

Description

Returns the total time in hours that the signal generator has been powered on since the last calibration.

Return Values

Data Type	Value	Description
Short	OpTime	The operating time in hours.

Examples

Visual Basic
```
OpTime = MyPTE1.GetGenOperationTime
```
Visual C++
```
OpTime = MyPTE1->GetGenOperationTime();
```
Visual C#
```
OpTime = MyPTE1.GetGenOperationTime();
```
Matlab
```
OpTime = MyPTE1.GetGenOperationTime
```

See Also

Get Calibration Reminder Operating Time
Set Calibration Reminder Operating Time
Get Calibration Reminder Operating Time Status
Set Calibration Reminder Operating Time Status

2.8 - Modulation Functions

The signal generator can be configured to produce a pulsed output, either in response to an external trigger or continuously using the generator's internal timing systems. The user stores the parameters of the pulse sequence in the generator's memory and can then enable/disable the output as required.

An example programming sequence to configure a pulse sequence using a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set generator to 6GHz and 0.5dBm output
MyPTE1.SetFreqAndPower(6000, 0.5, 0)

' Configure a pulse of 50us duration, 500us off time
MyPTE1.Set_PulseMode(50, 500, 0)

' Enable the output (continuous pulses)
MyPTE1.SetPowerON
```

Visual C++

```
// Set generator to 6GHz and 0.5dBm output
MyPTE1->SetFreqAndPower(6000, 0.5, 0);

// Configure a pulse of 50us duration, 500us off time
MyPTE1->Set_PulseMode(50, 500, 0);

// Enable the output (continuous pulses)
MyPTE1->SetPowerON;
```

Visual C#

```
// Set generator to 6GHz and 0.5dBm output
MyPTE1.SetFreqAndPower(6000, 0.5, 0);

// Configure a pulse of 50us duration, 500us off time
MyPTE1.Set_PulseMode(50, 500, 0);

// Enable the output (continuous pulses)
MyPTE1.SetPowerON;
```

Matlab

```
% Set generator to 6GHz and 0.5dBm output
MyPTE1.SetFreqAndPower(6000, 0.5, 0)

% Configure a pulse of 50us duration, 500us off time
MyPTE1.Set_PulseMode(50, 500, 0)

% Enable the output (continuous pulses)
MyPTE1.SetPowerON
```

Full details of the commands for configuring a pulsed output are covered in the following sections.

2.8 (a) - Set Pulse Mode

Declaration

```
Short Set_PulseMode(Short T_OFF, Short T_ON, Short Tunit)
```

Description

This function creates a pulsed output with a user specified pulse duration and time period. The output during the pulse "on" period is a CW signal with a frequency and power level which should be set by the user in advance (see Set Output Frequency and Power). The pulse period will repeat indefinitely until any other function is called by the user's program.

Parameters

Data Type	Variable	Description
Short	T_OFF	Required. The off period between pulses.
Short	T_ON	Required. The pulse "on" duration.
Short	Tunit	Required. The units for the T_OFF and T_ON time periods; 0 for microseconds or 1 for milliseconds.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_PulseMode(10, 2, 1)
        ' Set 2ms pulses of 1000MHz, 10dBm CW, separated by 10ms
```
Visual C++
```
status = MyPTE1->SetFreqAndPower(1000, 10, 0);
status = MyPTE1->Set_PulseMode(10, 2, 1);
        // Set 2ms pulses of 1000MHz, 10dBm CW, separated by 10ms
```
Visual C#
```
status = MyPTE1.SetFreqAndPower(1000, 10, 0);
status = MyPTE1.Set_PulseMode(10, 2, 1);
        // Set 2ms pulses of 1000MHz, 10dBm CW, separated by 10ms
```
Matlab
```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_PulseMode(10, 2, 1)
        % Set 2ms pulses of 1000MHz, 10dBm CW, separated by 10ms
```

See Also

Set Output Frequency and Power
Set Triggered Pulse Mode
Set External Pulse Modulation

2.8 (b) - Set Triggered Pulse Mode

Declaration

```
Short Set_PulseMode_Trigger(Short TriggerType, Short T_ON,
                                                     Short Tunit)
```

Description

This function creates a pulsed output with a user specified pulse duration that will start when an external trigger is received at the "Trigger In" input. The output during the pulse "on" period is a CW signal with a frequency and power level which should be set by the user in advance (see Set Output Frequency and Power). The pulsed output will be enabled until any other function is called by the user's program.

Parameters

Data Type	Variable	Description
Short	TriggerType	Required. The trigger input sequence that will trigger the pulsed output; 0 for Trigger In = on then off, or 1 for Trigger In = off then on.
Short	T_ON	Required. The pulse "on" duration.
Short	Tunit	Required. The units for the T_ON time period; 0 for microseconds or 1 for milliseconds.

Return Values

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

Examples

Visual Basic

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_PulseMode_Trigger(0, 2, 1)
        ' Set 2ms pulses of 1000MHz, 10dBm CW
        ' Start the pulse when a "on, off" is received at Trigger In
```

Visual C++

```
status = MyPTE1->SetFreqAndPower(1000, 10, 0);
status = MyPTE1->Set_PulseMode_Trigger(0, 2, 1);
        // Set 2ms pulses of 1000MHz, 10dBm CW
        // Start the pulse when a "on, off" is received at Trigger In
```

Visual C#

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0);
status = MyPTE1.Set_PulseMode_Trigger(0, 2, 1);
        // Set 2ms pulses of 1000MHz, 10dBm CW
        // Start the pulse when a "on, off" is received at Trigger In
```

Matlab

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_PulseMode_Trigger(0, 2, 1)
        % Set 2ms pulses of 1000MHz, 10dBm CW
        % Start the pulse when a "on, off" is received at Trigger In
```

See Also

Set Output Frequency and Power
Set Pulse Mode
Set External Pulse Modulation

Mini-Circuits

2.8 (c) - Set External Pulse Modulation

Declaration

```
Short Set_ExtPulseMod()
```

Description

This function enables a pulsed output in response to the generator's Trigger In port. The generator's CW output will be enabled with the specified frequency and power while the Trigger In port is held high. The output will be disabled while the Trigger In port is held low. The CW frequency and power for the "on" period should be set in advance (see Set Output Frequency and Power). The external pulse modulation mode will be disabled when any other command is sent to the generator.

Parameters

Data Type	Variable	Description
None		

Return Values

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

Examples

Visual Basic

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_ExtPulseMod()
    ' Enable externally modulated pulses of 1000MHz, 10dBm
```

Visual C++

```
status = MyPTE1->SetFreqAndPower(1000, 10, 0);
status = MyPTE1->Set_ExtPulseMod();
    // Enable externally modulated pulses of 1000MHz, 10dBm
```

Visual C#

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0);
status = MyPTE1.Set_ExtPulseMod();
    // Enable externally modulated pulses of 1000MHz, 10dBm
```

Matlab

```
status = MyPTE1.SetFreqAndPower(1000, 10, 0)
status = MyPTE1.Set_ExtPulseMod()
    % Enable externally modulated pulses of 1000MHz, 10dBm
```

See Also

Set Output Frequency and Power
Set Pulse Mode
Set Triggered Pulse Mode

2.9 - Frequency Sweep Functions

These functions define the frequency sweep capabilities of the generators. The signal generator can be configured to produce an automatic, swept frequency output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

An example programming sequence to configure a sweep on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.FSweep_SetStartFreq(1000)
MyPTE1.FSweep_SetStopFreq(6000)
MyPTE1.FSweep_SetStepSize(100)

' Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.FSweep_SetDwell(10)
MyPTE1.FSweep_SetPower(10)

' Start the sweep
MyPTE1.FSweep_SetMode(1)
```

Visual C++

```
// Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1->FSweep_SetStartFreq(1000);
MyPTE1->FSweep_SetStopFreq(6000);
MyPTE1->FSweep_SetStepSize(100);

// Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1->FSweep_SetDwell(10);
MyPTE1->FSweep_SetPower(10);

// Start the sweep
MyPTE1->FSweep_SetMode(1);
```

Visual C#

```
// Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.FSweep_SetStartFreq(1000);
MyPTE1.FSweep_SetStopFreq(6000);
MyPTE1.FSweep_SetStepSize(100);

// Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.FSweep_SetDwell(10);
MyPTE1.FSweep_SetPower(10);

// Start the sweep
MyPTE1.FSweep_SetMode(1);
```

Matlab

```
% Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.FSweep_SetStartFreq(1000)
MyPTE1.FSweep_SetStopFreq(6000)
MyPTE1.FSweep_SetStepSize(100)

% Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.FSweep_SetDwell(10)
MyPTE1.FSweep_SetPower(10)

% Start the sweep
MyPTE1.FSweep_SetMode(1)
```

Full details of the commands for configuring a frequency sweep are covered in the following sections.

2.9 (a) - Frequency Sweep – Get Direction

Declaration

```
Short FSweep_GetDirection()
```

Description

This function returns the current frequency sweep direction. The possible settings are:
0 – Increasing from start to stop frequency
1 – Decreasing from stop to start frequency
2 – Increasing from start to stop, before decreasing from stop to start frequency

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Increasing frequency sweep from start to stop frequency
Short	1	Decreasing frequency sweep from stop to start frequency
Short	2	Increasing then decreasing frequency sweep (from start to stop to start frequency)

Examples

Visual Basic
```
Sweep = MyPTE1.FSweep_GetDirection
```
Visual C++
```
Sweep = MyPTE1->FSweep_GetDirection();
```
Visual C#
```
Sweep = MyPTE1.FSweep_GetDirection();
```
Matlab
```
Sweep = MyPTE1.FSweep_GetDirection
```

See Also

Frequency Sweep – Set Direction

2.9 (b) - Frequency Sweep – Get Dwell Time

Declaration

```
Short FSweep_GetDwell()
```

Description

This function returns the current dwell time setting in milliseconds; this is the length of time that the generator will pause at each frequency point.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.FSweep_GetDwell
```
Visual C++
```
Dwell = MyPTE1->FSweep_GetDwell();
```
Visual C#
```
Dwell = MyPTE1.FSweep_GetDwell();
```
Matlab
```
Dwell = MyPTE1.FSweep_GetDwell
```

See Also

Frequency Sweep – Get Maximum Dwell Time
Frequency Sweep – Get Minimum Dwell Time
Frequency Sweep – Set Dwell Time

2.9 (c) - Frequency Sweep – Get Maximum Dwell Time

Declaration

```
Short FSweep_GetMaxDwell()
```

Description

This function returns the maximum allowed dwell time in milliseconds. Dwell time is the length of time that the generator will pause at each frequency point.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Maximum dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.FSweep_GetMaxDwell
```
Visual C++
```
Dwell = MyPTE1->FSweep_GetMaxDwell();
```
Visual C#
```
Dwell = MyPTE1.FSweep_GetMaxDwell();
```
Matlab
```
Dwell = MyPTE1.FSweep_GetMaxDwell
```

See Also

Frequency Sweep – Get Dwell Time
Frequency Sweep – Get Minimum Dwell Time
Frequency Sweep – Set Dwell Time

2.9 (d) - Frequency Sweep – Get Minimum Dwell Time

Declaration

```
Short FSweep_GetMinDwell()
```

Description

This function returns the minimum allowed dwell time in milliseconds. Dwell time is the length of time that the generator will pause at each frequency point.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Minimum dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.FSweep_GetMinDwell
```
Visual C++
```
Dwell = MyPTE1->FSweep_GetMinDwell();
```
Visual C#
```
Dwell = MyPTE1.FSweep_GetMinDwell();
```
Matlab
```
Dwell = MyPTE1.FSweep_GetMinDwell
```

See Also

Frequency Sweep – Get Dwell Time
Frequency Sweep – Get Maximum Dwell Time
Frequency Sweep – Set Dwell Time

2.9 (e) - Frequency Sweep – Get Power

Declaration

```
Float FSweep_GetPower()
```

Description

This function returns the current output power setting of the frequency sweep in dBm.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	-999	Command failed
Float	Power	Output power in dBm

Examples

Visual Basic
```
Power = MyPTE1.FSweep_GetPower
```
Visual C++
```
Power = MyPTE1->FSweep_GetPower();
```
Visual C#
```
Power = MyPTE1.FSweep_GetPower();
```
Matlab
```
Power = MyPTE1.FSweep_GetPower
```

See Also

Get Generator Maximum Power Spec
Get Generator Minimum Power Spec
Frequency Sweep – Set Power

2.9 (f) - Frequency Sweep – Get Start Frequency

Declaration

```
Double FSweep_GetStartFreq()
```

Description

This function returns the start frequency in MHz of the current frequency sweep.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Freq	Start frequency in MHz

Examples

Visual Basic
```
Freq = MyPTE1.FSweep_GetStartFreq
```
Visual C++
```
Freq = MyPTE1->FSweep_GetStartFreq();
```
Visual C#
```
Freq = MyPTE1.FSweep_GetStartFreq();
```
Matlab
```
Freq = MyPTE1.FSweep_GetStartFreq
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Frequency Sweep – Get Stop Frequency
Frequency Sweep – Set Start Frequency
Frequency Sweep – Set Stop Frequency

2.9 (g) - Frequency Sweep – Get Stop Frequency

Declaration

```
Double FSweep_GetStopFreq()
```

Description

This function returns the stop frequency in MHz of the current frequency sweep.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Freq	Stop frequency in MHz

Examples

Visual Basic
```
Freq = MyPTE1.FSweep_GetStopFreq
```
Visual C++
```
Freq = MyPTE1->FSweep_GetStopFreq();
```
Visual C#
```
Freq = MyPTE1.FSweep_GetStopFreq();
```
Matlab
```
Freq = MyPTE1.FSweep_GetStopFreq
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Frequency Sweep – Get Start Frequency
Frequency Sweep – Set Start Frequency
Frequency Sweep – Set Stop Frequency

2.9 (h) - Frequency Sweep – Get Step Size

Declaration

```
Double FSweep_GetStepSize()
```

Description

This function returns the step size in MHz of the current frequency sweep.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Freq	Step size in MHz

Examples

Visual Basic
```
Freq = MyPTE1.FSweep_GetStepSize
```
Visual C++
```
Freq = MyPTE1->FSweep_GetStepSize();
```
Visual C#
```
Freq = MyPTE1.FSweep_GetStepSize();
```
Matlab
```
Freq = MyPTE1.FSweep_GetStepSize
```

See Also

Get Generator Step Size Spec
Frequency Sweep – Get Start Frequency
Frequency Sweep – Get Stop Frequency
Frequency Sweep – Set Start Frequency
Frequency Sweep – Set Stop Frequency
Frequency Sweep – Set Step Size

2.9 (i) - Frequency Sweep – Get Trigger In Mode

Declaration

```
Short FSweep_GetTriggerIn()
```

Description

This function returns the Trigger Input mode for the frequency sweep, this dictates how the generator will respond to an external trigger:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each frequency point
2 – Wait for external trigger (Trigger In = logic 1) before starting each frequency

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Ignore Trigger In
Short	1	Wait for Trigger In for each frequency point
Short	2	Wait for Trigger In before starting each sweep

Examples

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

See Also

Frequency Sweep – Get Trigger Out Mode
Frequency Sweep – Set Trigger In Mode
Frequency/Power Hop – Set Trigger Out Mode

2.9 (j) - Frequency Sweep – Get Trigger Out Mode

Declaration

```
Short FSweep_GetTriggerOut()
```

Description

This function returns Trigger Output mode for the frequency sweep, this dictates how the Trigger Out port will be used during the frequency sweep:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each frequency point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each frequency sweep is initiated

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Trigger Out disabled
Short	1	Trigger Out set at each frequency point
Short	2	Trigger Out set as each sweep is initialized

Examples

Visual Basic

```
Status = MyPTE1.FSweep_GetTriggerOut
```

Visual C++

```
Status = MyPTE1->FSweep_GetTriggerOut();
```

Visual C#

```
Status = MyPTE1.FSweep_GetTriggerOut();
```

Matlab

```
Status = MyPTE1.FSweep_GetTriggerOut
```

See Also

Frequency Sweep – Get Trigger In Mode
Frequency Sweep – Set Trigger In Mode
Frequency/Power Hop – Set Trigger Out Mode

2.9 (k) - Frequency Sweep – Set Direction

Declaration

```
Short FSweep_SetDirection(Short SweepDirection)
```

Description

This function sets the direction of the frequency sweep. The 3 options are:
0 – Increasing from start to stop frequency
1 – Decreasing from stop to start frequency
2 – Increasing from start to stop, before decreasing from stop to start frequency

Parameters

Data Type	Variable	Description
Short	Sweep _Direction	Required. Numeric value corresponding to the sweep direction mode: 0 - Increasing frequency sweep from start to stop frequency 1 - Decreasing frequency sweep from stop to start frequency 2 - Increasing then decreasing frequency sweep (from start to stop to start frequency)

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetDirection(0)
```
Visual C++
```
Status = MyPTE1->FSweep_SetDirection(0);
```
Visual C#
```
Status = MyPTE1.FSweep_SetDirection(0);
```
Matlab
```
Status = MyPTE1.FSweep_SetDirection(0)
```

See Also

Frequency Sweep – Get Direction

2.9 (l) - Frequency Sweep – Set Dwell Time

Declaration

```
Short FSweep_SetDwell(Short dwell_msec)
```

Description

This function sets the dwell time in milliseconds; this is the length of time that the generator will pause at each frequency point.

Parameters

Data Type	Variable	Description
Short	dwell_msec	Required. The dwell time in milliseconds.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetDwell(15)
```
Visual C++
```
Status = MyPTE1->FSweep_SetDwell(15);
```
Visual C#
```
Status = MyPTE1.FSweep_SetDwell(15);
```
Matlab
```
Status = MyPTE1.FSweep_SetDwell(15)
```

See Also

Frequency Sweep – Get Dwell Time
Frequency Sweep – Get Maximum Dwell Time
Frequency Sweep – Get Minimum Dwell Time

2.9 (m) - Frequency Sweep – Start/Stop Sweep

Declaration

```
Short FSweep_SetMode(Short onoff)
```

Description

This function starts or stops the frequency sweep using the previously defined parameters.

Note: The frequency sweep will stop automatically if any other command is sent.

Parameters

Data Type	Variable	Description
Short	onoff	Required. Integer value to enable/disable the sweep: 1 – Start frequency sweep 0 – Stop frequency sweep

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.FSweep_SetMode(1)     ' Start
Status = MyPTE1.FSweep_SetMode(0)     ' Stop
```

Visual C++

```
Status = MyPTE1->FSweep_SetMode(1);   // Start
Status = MyPTE1->FSweep_SetMode(0);   // Stop
```

Visual C#

```
Status = MyPTE1.FSweep_SetMode(1);    // Start
Status = MyPTE1.FSweep_SetMode(0);    // Stop
```

Matlab

```
Status = MyPTE1.FSweep_SetMode(1)     % Start
Status = MyPTE1.FSweep_SetMode(0)     % Stop
```

See Also

Frequency/Power Hop – Start/Stop Hop Sequence
Power Sweep – Start/Stop Sweep

2.9 (n) - Frequency Sweep – Set Power

Declaration

```
Short FSweep_SetPower(Float Pr)
```

Description

This function sets the output power level in dBm to be used for the frequency sweep in.

Parameters

Data Type	Variable	Description
Float	Pr	Required. The fixed power level in dBm to be used for the frequency sweep.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetPower(-10.5)
```
Visual C++
```
Status = MyPTE1->FSweep_SetPower(-10.5);
```
Visual C#
```
Status = MyPTE1.FSweep_SetPower(-10.5);
```
Matlab
```
Status = MyPTE1.FSweep_SetPower(-10.5)
```

See Also

Frequency Sweep – Get Power

2.9 (o) - Frequency Sweep – Set Start Frequency

Declaration

```
Short FSweep_SetStartFreq(Double Fr)
```

Description

This function sets the start frequency in MHz for the sweep.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The start frequency in MHz.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetStartFreq(250)
```
Visual C++
```
Status = MyPTE1->FSweep_SetStartFreq(250);
```
Visual C#
```
Status = MyPTE1.FSweep_SetStartFreq(250);
```
Matlab
```
Status = MyPTE1.FSweep_SetStartFreq(250)
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Frequency Sweep – Get Start Frequency
Frequency Sweep – Get Stop Frequency
Frequency Sweep – Set Stop Frequency

2.9 (p) - Frequency Sweep – Set Stop Frequency

Declaration

```
Short FSweep_SetStopFreq(Double Fr)
```

Description

This function sets the stop frequency in MHz for the sweep.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The stop frequency in MHz.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetStopFreq(5500)
```
Visual C++
```
Status = MyPTE1->FSweep_SetStopFreq(5500);
```
Visual C#
```
Status = MyPTE1.FSweep_SetStopFreq(5500);
```
Matlab
```
Status = MyPTE1.FSweep_SetStopFreq(5500)
```

See Also

Get Generator Maximum Frequency Spec
Get Generator Minimum Frequency Spec
Frequency Sweep – Get Start Frequency
Frequency Sweep – Get Stop Frequency
Frequency Sweep – Set Start Frequency

2.9 (q) - Frequency Sweep – Set Step Size

Declaration

```
Short FSweep_SetStepSize(Double Fr)
```

Description

This function sets the step size in MHz to be used in the frequency sweep.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The step size in MHz.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetStepSize(0.1)
```
Visual C++
```
Status = MyPTE1->FSweep_SetStepSize(0.1);
```
Visual C#
```
Status = MyPTE1.FSweep_SetStepSize(0.1);
```
Matlab
```
Status = MyPTE1.FSweep_SetStepSize(0.1)
```

See Also

Get Generator Step Size Spec
Frequency Sweep – Get Start Frequency
Frequency Sweep – Get Stop Frequency
Frequency Sweep – Get Step Size
Frequency Sweep – Set Start Frequency
Frequency Sweep – Set Stop Frequency

2.9 (r) - Frequency Sweep – Set Trigger In Mode

Declaration

```
Short FSweep_SetTriggerIn(Short SweepTriggerIn)
```

Description

This function specifies how the frequency sweep should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each frequency point
2 – Wait for external trigger (Trigger In = logic 1) before starting each frequency sweep

Parameters

Data Type	Variable	Description
Short	Sweep _TriggerIn	Required. Integer value to specify the Trigger In mode: 0 - Ignore Trigger In 1 - Wait for Trigger In before each frequency point 2 - Wait for Trigger In before commencing sweep

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetTriggerIn(1)
```
Visual C++
```
Status = MyPTE1->FSweep_SetTriggerIn(1);
```
Visual C#
```
Status = MyPTE1.FSweep_SetTriggerIn(1);
```
Matlab
```
Status = MyPTE1.FSweep_SetTriggerIn(1)
```

See Also

Frequency Sweep – Get Trigger In Mode
Frequency Sweep – Get Trigger Out Mode
Frequency Sweep – Set Trigger Out Mode

2.9 (s) - Frequency Sweep – Set Trigger Out Mode

Declaration

```
Short FSweep_SetTriggerOut(Short SweepTriggerOut)
```

Description

This function specifies how the Trigger Out port will be used during the frequency sweep. The modes are:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each frequency point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each frequency sweep is initiated

Parameters

Data Type	Variable	Description
Short	Sweep _TriggerOut	Required. Integer value to specify the Trigger Out mode: 0 - Trigger Out disabled 1 - Set Trigger Out at each frequency point 2 - Set Trigger Out on commencing the sweep

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.FSweep_SetTriggerOut(1)
```
Visual C++
```
Status = MyPTE1->FSweep_SetTriggerOut(1);
```
Visual C#
```
Status = MyPTE1.FSweep_SetTriggerOut(1);
```
Matlab
```
Status = MyPTE1.FSweep_SetTriggerOut(1)
```

See Also

Frequency Sweep – Get Trigger In Mode
Frequency Sweep – Get Trigger Out Mode
Frequency Sweep – Set Trigger In Mode

2.10 - Power Sweep Functions

These functions define the power sweep capabilities of the generators. The signal generator can be configured to produce an automatic, swept power output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

An example programming sequence to configure a sweep on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.PSweep_SetStartPower(-20)
MyPTE1.PSweep_SetStopPower(20)
MyPTE1.PSweep_SetStepSize(0.5)

' Set fixed 1000MHz output and 10ms dwell time for the sweep
MyPTE1.PSweep_SetDwell(10)
MyPTE1.PSweep_SetFreq(1000)

' Start the sweep
MyPTE1.PSweep_SetMode(1)
```

Visual C++

```
// Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1->PSweep_SetStartPower(-20);
MyPTE1->PSweep_SetStopPower(20);
MyPTE1->PSweep_SetStepSize(0.5);

// Set fixed 1000MHz output and 10ms dwell time for the sweep
MyPTE1->PSweep_SetDwell(10);
MyPTE1->PSweep_SetFreq(1000);

// Start the sweep
MyPTE1->PSweep_SetMode(1);
```

Visual C#

```
// Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.PSweep_SetStartPower(-20);
MyPTE1.PSweep_SetStopPower(20);
MyPTE1.PSweep_SetStepSize(0.5);

// Set fixed 1000MHz output and 10ms dwell time for the sweep
MyPTE1.PSweep_SetDwell(10);
MyPTE1.PSweep_SetFreq(1000);

// Start the sweep
MyPTE1.PSweep_SetMode(1);
```

Matlab

```
% Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.PSweep_SetStartPower(-20)
MyPTE1.PSweep_SetStopPower(20)
MyPTE1.PSweep_SetStepSize(0.5)

% Set fixed 1000MHz output and 10ms dwell time for the sweep
MyPTE1.PSweep_SetDwell(10)
MyPTE1.PSweep_SetFreq(1000)

% Start the sweep
MyPTE1.PSweep_SetMode(1)
```

Full details of the commands for configuring a frequency sweep are covered in the following sections.

2.10 (a) - Power Sweep – Get Direction

Declaration

```
Short PSweep_GetDirection()
```

Description

This function returns the current power sweep direction. The possible settings are:
0 – Increasing from start to stop power
1 – Decreasing from stop to start power
2 – Increasing from start to stop, before decreasing from stop to start power

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Ascending power sweep
Short	1	Descending power sweep
Short	2	Ascending then descending power sweep

Examples

Visual Basic
```
Sweep = MyPTE1.PSweep_GetDirection
```
Visual C++
```
Sweep = MyPTE1->PSweep_GetDirection();
```
Visual C#
```
Sweep = MyPTE1.PSweep_GetDirection();
```
Matlab
```
Sweep = MyPTE1.PSweep_GetDirection
```

See Also

Power Sweep – Set Direction

2.10 (b) - Power Sweep – Get Dwell Time

Declaration

```
Short PSweep_GetDwell()
```

Description

This function returns the current dwell time setting in milliseconds; this is the length of time that the generator will pause at each power setting.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.PSweep_GetDwell
```
Visual C++
```
Dwell = MyPTE1->PSweep_GetDwell();
```
Visual C#
```
Dwell = MyPTE1.PSweep_GetDwell();
```
Matlab
```
Dwell = MyPTE1.PSweep_GetDwell
```

See Also

Power Sweep – Get Maximum Dwell Time
Power Sweep – Get Minimum Dwell Time
Power Sweep – Set Dwell Time

2.10 (c) - Power Sweep – Get Maximum Dwell Time

Declaration

```
Short PSweep_GetMaxDwell()
```

Description

This function returns the maximum allowed dwell time in milliseconds. Dwell time is the length of time that the generator will pause at each power setting.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Maximum dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.PSweep_GetMaxDwell
```
Visual C++
```
Dwell = MyPTE1->PSweep_GetMaxDwell();
```
Visual C#
```
Dwell = MyPTE1.PSweep_GetMaxDwell();
```
Matlab
```
Dwell = MyPTE1.PSweep_GetMaxDwell
```

See Also

Power Sweep – Get Dwell Time
Power Sweep – Get Minimum Dwell Time
Power Sweep – Set Dwell Time

2.10 (d) - Power Sweep – Get Minimum Dwell Time

Declaration

```
Short PSweep_GetMinDwell()
```

Description

This function returns the minimum allowed dwell time in milliseconds. Dwell time is the length of time that the generator will pause at each power setting.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Time	Minimum dwell time in milliseconds

Examples

Visual Basic
```
Dwell = MyPTE1.PSweep_GetMinDwell
```
Visual C++
```
Dwell = MyPTE1->PSweep_GetMinDwell();
```
Visual C#
```
Dwell = MyPTE1.PSweep_GetMinDwell();
```
Matlab
```
Dwell = MyPTE1.PSweep_GetMinDwell
```

See Also

Power Sweep – Get Dwell Time
Power Sweep – Get Maximum Dwell Time
Power Sweep – Set Dwell Time

2.10 (e) - Power Sweep – Get Frequency

Declaration

```
Float PSweep_GetFreq()
```

Description

This function returns the current frequency setting of the power sweep in MHz.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Float	[0	Command failed
Float	Frequency	Frequency in MHz

Examples

Visual Basic
```
Freq = MyPTE1.PSweep_GetFreq
```
Visual C++
```
Freq = MyPTE1->PSweep_GetFreq();
```
Visual C#
```
Freq = MyPTE1.PSweep_GetFreq();
```
Matlab
```
Freq = MyPTE1.PSweep_GetFreq
```

See Also

Power Sweep – Set Frequency

2.10 (f) - Power Sweep – Get Start Power

Declaration

```
Double PSweep_GetStartPower()
```

Description

This function returns the start power of the current power sweep in dBm.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Power	Start power in dBm

Examples

Visual Basic
```
Power = MyPTE1.PSweep_GetStartPower
```
Visual C++
```
Power = MyPTE1->PSweep_GetStartPower();
```
Visual C#
```
Power = MyPTE1.PSweep_GetStartPower();
```
Matlab
```
Power = MyPTE1.PSweep_GetStartPower
```

See Also

Power Sweep – Get Stop Power
Power Sweep – Get Step Size
Power Sweep – Set Start Power
Power Sweep – Set Stop Power
Power Sweep – Set Step Size

2.10 (g) - Power Sweep – Get Stop Power

Declaration

```
Double PSweep_GetStopPower()
```

Description

This function returns the stop power of the current power sweep in dBm.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Power	Stop power in dBm

Examples

Visual Basic
```
Power = MyPTE1.PSweep_GetStopPower
```
Visual C++
```
Power = MyPTE1->PSweep_GetStopPower();
```
Visual C#
```
Power = MyPTE1.PSweep_GetStopPower();
```
Matlab
```
Power = MyPTE1.PSweep_GetStopPower
```

See Also

Power Sweep – Get Start Power
Power Sweep – Get Step Size
Power Sweep – Set Start Power
Power Sweep – Set Stop Power
Power Sweep – Set Step Size

2.10 (h) - Power Sweep – Get Step Size

Declaration

```
Double PSweep_GetStepSize()
```

Description

This function returns the step size in dBm of the current power sweep.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Double	[0	Command failed
Double	Power	Step size in dBm

Examples

Visual Basic
```
Power = MyPTE1.PSweep_GetStepSize
```
Visual C++
```
Power = MyPTE1->PSweep_GetStepSize();
```
Visual C#
```
Power = MyPTE1.PSweep_GetStepSize();
```
Matlab
```
Power = MyPTE1.PSweep_GetStepSize
```

See Also

Power Sweep – Get Start Power
Power Sweep – Get Stop Power
Power Sweep – Set Start Power
Power Sweep – Set Stop Power
Power Sweep – Set Step Size

2.10 (i) - Power Sweep – Get Trigger In Mode

Declaration

```
Short PSweep_GetTriggerIn()
```

Description

This function returns the Trigger Input mode for the power sweep, this dictates how the generator will respond to an external trigger:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each power
2 – Wait for external trigger (Trigger In = logic 1) before starting each power sweep

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Ignore Trigger In
Short	1	Wait for Trigger In for each power setting
Short	2	Wait for Trigger In before commencing each sweep

Examples

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

See Also

Power Sweep – Get Trigger Out Mode
Power Sweep – Set Trigger In Mode
Power Sweep – Set Trigger Out Mode

2.10 (j) - Power Sweep – Get Trigger Out Mode

Declaration

```
Short PSweep_GetTriggerOut()
```

Description

This function returns Trigger Output mode for the power sweep, this dictates how the Trigger Out port will be used during the power sweep:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each power is set
2 – Provide a trigger output (Trigger Out = logic 1) as each power sweep is initiated

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Trigger Out disabled
Short	1	Trigger Out set at each power
Short	2	Trigger Out set as each sweep is initiated

Examples

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

See Also

Power Sweep – Get Trigger In Mode
Power Sweep – Set Trigger In Mode
Power Sweep – Set Trigger Out Mode

2.10 (k) - Power Sweep – Set Direction

Declaration

```
Short PSweep_SetDirection(Short SweepDirection)
```

Description

This function sets the direction of the power sweep:
0 – Ascending from start to stop power
1 – Descending from stop to start power
2 – Ascending, then descending power

Parameters

Data Type	Variable	Description
Short	Sweep _Direction	Required. Numeric value corresponding to the sweep direction mode: 0 – Ascending power 1 – Descending power 2 – Ascending, then descending power

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetDirection(0)
```
Visual C++
```
Status = MyPTE1->PSweep_SetDirection(0);
```
Visual C#
```
Status = MyPTE1.PSweep_SetDirection(0);
```
Matlab
```
Status = MyPTE1.PSweep_SetDirection(0)
```

See Also

Power Sweep – Get Direction

2.10 (l) - Power Sweep – Set Dwell Time

Declaration

```
Short PSweep_SetDwell(Short dwell_msec)
```

Description

This function sets the dwell time in milliseconds; this is the length of time that the generator will pause at each power setting.

Parameters

Data Type	Variable	Description
Short	dwell_msec	Required. The dwell time in milliseconds.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetDwell(15)
```
Visual C++
```
Status = MyPTE1->PSweep_SetDwell(15);
```
Visual C#
```
Status = MyPTE1.PSweep_SetDwell(15);
```
Matlab
```
Status = MyPTE1.PSweep_SetDwell(15)
```

See Also

Power Sweep – Get Dwell Time
Power Sweep – Get Maximum Dwell Time
Power Sweep – Get Minimum Dwell Time

2.10 (m) - Power Sweep – Start/Stop Sweep

Declaration

```
Short PSweep_SetMode(Short onoff)
```

Description

This function starts or stops the power sweep using the previously defined parameters.

Note: The power sweep will stop automatically if any other command is sent.

Parameters

Data Type	Variable	Description
Short	onoff	Required. Integer value to enable/disable the sweep: 1 – Start power sweep 0 – Stop power sweep

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetMode(1)      ' Start
Status = MyPTE1.PSweep_SetMode(0)      ' Stop
```
Visual C++
```
Status = MyPTE1->PSweep_SetMode(1);   // Start
Status = MyPTE1->PSweep_SetMode(0);   // Stop
```
Visual C#
```
Status = MyPTE1.PSweep_SetMode(1);    // Start
Status = MyPTE1.PSweep_SetMode(0);    // Stop
```
Matlab
```
Status = MyPTE1.PSweep_SetMode(1)     % Start
Status = MyPTE1.PSweep_SetMode(0)     % Stop
```

See Also

Frequency Sweep – Start/Stop Sweep
Frequency/Power Hop – Start/Stop Hop Sequence

2.10 (n) - Power Sweep – Set Frequency

Declaration

```
Short PSweep_SetFreq(Float Fr)
```

Description

This function sets the output frequency in MHz to be used for the power sweep.

Parameters

Data Type	Variable	Description
Float	Fr	Required. The fixed frequency in MHz to be used for the power sweep.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetFreq(1000)
```
Visual C++
```
Status = MyPTE1->PSweep_SetFreq(1000);
```
Visual C#
```
Status = MyPTE1.PSweep_SetFreq(1000);
```
Matlab
```
Status = MyPTE1.PSweep_SetFreq(1000)
```

See Also

Power Sweep – Get Frequency

2.10 (o) - Power Sweep – Set Start Power

Declaration

```
Short PSweep_SetStartPower(Float Pr)
```

Description

This function sets the start power in dBm for the sweep.

Parameters

Data Type	Variable	Description
Float	Pr	Required. The start power in dBm.

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.PSweep_SetStartPower(-10)
```

Visual C++

```
Status = MyPTE1->PSweep_SetStartPower(-10);
```

Visual C#

```
Status = MyPTE1.PSweep_SetStartPower(-10);
```

Matlab

```
Status = MyPTE1.PSweep_SetStartPower(-10)
```

See Also

Power Sweep – Get Start Power
Power Sweep – Get Stop Power
Power Sweep – Set Step Size
Power Sweep – Set Stop Power
Power Sweep – Set Step Size

2.10 (p) - Power Sweep – Set Stop Power

Declaration

```
Short PSweep_SetStopPower(Float Pr)
```

Description

This function sets the stop power in dBm for the sweep.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The stop power in dBm.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetStopPower(5500)
```
Visual C++
```
Status = MyPTE1->PSweep_SetStopPower(5500);
```
Visual C#
```
Status = MyPTE1.PSweep_SetStopPower(5500);
```
Matlab
```
Status = MyPTE1.PSweep_SetStopPower(5500)
```

See Also

Power Sweep – Get Start Power
Power Sweep – Get Stop Power
Power Sweep – Set Step Size
Power Sweep – Set Start Power
Power Sweep – Set Step Size

2.10 (q) - Power Sweep – Set Step Size

Declaration

```
Short PSweep_SetStepSize(Float Pr)
```

Description

This function sets the step size in dBm to be used in the power sweep.

Parameters

Data Type	Variable	Description
Double	Fr	Required. The step size in dBm.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetStepSize(0.5)
```
Visual C++
```
Status = MyPTE1->PSweep_SetStepSize(0.5);
```
Visual C#
```
Status = MyPTE1.PSweep_SetStepSize(0.5);
```
Matlab
```
Status = MyPTE1.PSweep_SetStepSize(0.5)
```

See Also

Power Sweep – Get Start Power
Power Sweep – Get Stop Power
Power Sweep – Set Start Power
Power Sweep – Set Stop Power
Power Sweep – Set Step Size

2.10 (r) - Power Sweep – Set Trigger In Mode

Declaration

```
Short PSweep_SetTriggerIn(Short SweepTriggerIn)
```

Description

This function specifies how the power sweep should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each power
2 – Wait for external trigger (Trigger In = logic 1) before starting each power sweep

Parameters

Data Type	Variable	Description
Short	Sweep _TriggerIn	Required. Integer value to specify the Trigger In mode: 0 – Ignore Trigger In 1 - Wait for Trigger In before each power 2 - Wait for Trigger In before commencing each sweep

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.PSweep_SetTriggerIn(1)
```
Visual C++
```
Status = MyPTE1->PSweep_SetTriggerIn(1);
```
Visual C#
```
Status = MyPTE1.PSweep_SetTriggerIn(1);
```
Matlab
```
Status = MyPTE1.PSweep_SetTriggerIn(1)
```

See Also

Power Sweep – Get Trigger In Mode
Power Sweep – Get Trigger Out Mode
Power Sweep – Set Trigger Out Mode

2.10 (s) - Power Sweep – Set Trigger Out Mode

Declaration

```
Short PSweep_SetTriggerOut(Short SweepTriggerOut)
```

Description

This function specified how the Trigger Out port will be used during the power sweep. The modes are:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each power is set
2 – Provide a trigger output (Trigger Out = logic 1) as each power sweep is initiated

Parameters

Data Type	Variable	Description
Short	Sweep _TriggerOut	Required. Integer value to specify the Trigger Out mode: 0 – Trigger Out disabled 1 – Set Trigger Out at each power 2 – Set Trigger Out on commencing each sweep

Return Values

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

Examples

Visual Basic

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

Visual C++

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

Visual C#

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

Matlab

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

See Also

Power Sweep – Get Trigger In Mode
Power Sweep – Get Trigger Out Mode
Power Sweep – Set Trigger In Mode

2.11 - Frequency/Power Hop Functions

These functions define the frequency and power hop capabilities of the generators. The signal generator can be configured to automatically hop through a series of user defined frequency and power outputs using the generator's internal timing systems. The user stores the parameters of the hop sequence in the generator's memory and can then enable/disable the output as required.

An example programming sequence to configure a hop sequence on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.Hop_SetNoOfPoints(50)
MyPTE1.Hop_SetDwell(10)

' Set point 1 to 1000MHz, -10dBm
MyPTE1.Hop_SetPoint (1, 1000, -10)

' Set point 2 to 1100MHz, -8dBm
MyPTE1.Hop_SetPoint (2, 1100, -8)

' Set points 3 to 50 in the same way

' Start the hop sequence
MyPTE1.Hop_SetMode(1)
```

Visual C++

```
// Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1->Hop_SetNoOfPoints(50);
MyPTE1->Hop_SetDwell(10);

// Set point 1 to 1000MHz, -10dBm
MyPTE1->Hop_SetPoint (1, 1000, -10);

// Set point 2 to 1100MHz, -8dBm
MyPTE1->Hop_SetPoint (2, 1100, -8);

// Index and set points 3 to 50 in the same way

// Start the hop sequence
MyPTE1->Hop_SetMode(1);
```

Visual C#

```
// Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.Hop_SetNoOfPoints(50);
MyPTE1.Hop_SetDwell(10);

// Set point 1 to 1000MHz, -10dBm
MyPTE1.Hop_SetPoint (1, 1000, -10);

// Set point 2 to 1100MHz, -8dBm
MyPTE1.Hop_SetPoint (2, 1100, -8);

// Index and set points 3 to 50 in the same way

// Start the hop sequence
MyPTE1.Hop_SetMode(1);
```

Matlab

```
% Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.Hop_SetNoOfPoints(50)
MyPTE1.Hop_SetDwell(10)

% Set point 1 to 1000MHz, -10dBm
MyPTE1.Hop_SetPoint (1, 1000, -10)

% Set point 2 to 1100MHz, -8dBm
MyPTE1.Hop_SetPoint (2, 1100, -8)

% Index and set points 3 to 50 in the same way

% Start the hop sequence
MyPTE1.Hop_SetMode(1)
```

Full details of the commands for configuring a frequency/power hop sequence are covered in the following sections.

2.11 (a) - Frequency/Power Hop – Get Direction

Declaration

```
Short Hop_GetDirection()
```

Description

This function returns the direction setting for the current hop sequence:
0 – From first to last in the list
1 – From last to first in the list
2 – From first to last in the list, then back from last to first

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Ascending frequency from lowest to highest
Short	1	Descending frequency from highest to lowest
Short	2	Ascending frequency from lowest to highest, then descending to lowest

Examples

Visual Basic
```
Direction = MyPTE1.Hop_GetDirection
```
Visual C++
```
Direction = MyPTE1->Hop_GetDirection();
```
Visual C#
```
Direction = MyPTE1.Hop_GetDirection();
```
Matlab
```
Direction = MyPTE1.Hop_GetDirection
```

See Also

Frequency/Power Hop – Set Direction

2.11 (b) - Frequency/Power Hop – Get Dwell Time

Declaration

```
Short Hop_GetDwell()
```

Description

This function returns the dwell time setting in milliseconds for the current hop sequence.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Dwell Time	Dwell time setting in milliseconds

Examples

Visual Basic
```
DwellTime = MyPTE1.Hop_GetDwell
```
Visual C++
```
DwellTime = MyPTE1->Hop_GetDwell();
```
Visual C#
```
DwellTime = MyPTE1.Hop_GetDwell();
```
Matlab
```
DwellTime = MyPTE1.Hop_GetDwell
```

See Also

Frequency/Power Hop – Get Maximum Dwell Time
Frequency/Power Hop – Get Minimum Dwell Time
Power Sweep – Set Dwell Time

2.11 (c) - Frequency/Power Hop – Get Maximum Dwell Time

Declaration

```
Short Hop_GetMaxDwell()
```

Description

This function returns the maximum allowed dwell time in milliseconds for any point in a hop sequence.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Dwell Time	Maximum allowed dwell time in milliseconds

Examples

Visual Basic
```
DwellTime = MyPTE1.Hop_GetMaxDwell
```
Visual C++
```
DwellTime = MyPTE1->Hop_GetMaxDwell();
```
Visual C#
```
DwellTime = MyPTE1.Hop_GetMaxDwell();
```
Matlab
```
DwellTime = MyPTE1.Hop_GetMaxDwell
```

See Also

Frequency/Power Hop – Get Dwell Time
Frequency/Power Hop – Get Minimum Dwell Time
Power Sweep – Set Dwell Time

2.11 (d) - Frequency/Power Hop – Get Minimum Dwell Time

Declaration

```
Short Hop_GetMinDwell()
```

Description

This function returns the minimum allowed dwell time in milliseconds for any point in a hop sequence.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Dwell Time	Minimum allowed dwell time in milliseconds

Examples

Visual Basic
```
DwellTime = MyPTE1.Hop_GetMinDwell
```
Visual C++
```
DwellTime = MyPTE1->Hop_GetMinDwell();
```
Visual C#
```
DwellTime = MyPTE1.Hop_GetMinDwell();
```
Matlab
```
DwellTime = MyPTE1.Hop_GetMinDwell
```

See Also

Frequency/Power Hop – Get Dwell Time
Frequency/Power Hop – Get Maximum Dwell Time
Power Sweep – Set Dwell Time

2.11 (e) - Frequency/Power Hop – Get Number of Points

Declaration

```
Short Hop_GetNoOfPoints()
```

Description

This function returns the number of points set for the current hop sequence.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Hops	Number of frequency hop points set

Examples

Visual Basic
```
Hops = MyPTE1.Hop_GetNoOfPoints
```
Visual C++
```
Hops = MyPTE1->Hop_GetNoOfPoints();
```
Visual C#
```
Hops = MyPTE1.Hop_GetNoOfPoints();
```
Matlab
```
Hops = MyPTE1.Hop_GetNoOfPoints
```

See Also

Frequency/Power Hop – Get Maximum Number of Points
Frequency/Power Hop – Get Hop Point
Frequency/Power Hop – Set Number of Points
Frequency/Power Hop – Set Hop Point

2.11 (f) - Frequency/Power Hop – Get Maximum Number of Points

Declaration

```
Short Hop_GetMaxNoOfPoints()
```

Description

This function returns the maximum allowed number of points in a hop sequence.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	Max Hops	Maximum number of frequency hop points

Examples

Visual Basic
```
Hops = MyPTE1.Hop_GetMaxNoOfPoints
```
Visual C++
```
Hops = MyPTE1->Hop_GetMaxNoOfPoints();
```
Visual C#
```
Hops = MyPTE1.Hop_GetMaxNoOfPoints();
```
Matlab
```
Hops = MyPTE1.Hop_GetMaxNoOfPoints
```

See Also

Frequency/Power Hop – Get Number of Points
Frequency/Power Hop – Get Hop Point
Frequency/Power Hop – Set Number of Points
Frequency/Power Hop – Set Hop Point

2.11 (g) - Frequency/Power Hop – Get Hop Point

Declaration

```
Short Hop_GetPoint(Short PointNo, Double HopFreq, Float HopPower)
```

Description

This function returns the frequency and power settings for a specific point in a hop sequence, from 1 to the maximum allowed number of points (device specific, see Frequency/Power Hop – Get Maximum Number of Points).

Parameters

Data Type	Variable	Description
Short	PointNo	Required. The point number; from 0 to (n - 1) where n equals the number of hop points in the sequence.
Double	HopFreq	Required. User defined variable which will be overwritten with the frequency in MHz of the specified hop point.
Float	HopPower	Required. User defined variable which will be overwritten with the power in dBm of the specified hop point.

Return Values

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

Examples

Visual Basic
```
status = MyPTE1.Hop_GetPoint(PointNo, HopFreq, HopPower)
```
Visual C++
```
status = MyPTE1->Hop_GetPoint(PointNo, HopFreq, HopPower);
```
Visual C#
```
status = MyPTE1.Hop_GetPoint(PointNo, HopFreq, HopPower);
```
Matlab
```
[PointNo, HopFreq, HopPower] = MyPTE1.Hop_GetPoint(PointNo, HopFreq,
                                                            _ HopPower)
```

See Also

Frequency/Power Hop – Get Maximum Number of Points
Frequency/Power Hop – Set Number of Points
Frequency/Power Hop – Set Hop Point

2.11 (h) - Frequency/Power Hop – Get Trigger In Mode

Declaration

```
Short Hop_GetTriggerIn()
```

Description

This function returns the Trigger Input mode for the hop sequence, this dictates how the generator will respond to an external trigger:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before hopping to the next point
2 – Wait for external trigger (Trigger In = logic 1) before starting each hop sequence

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Ignore Trigger In
Short	1	Wait for Trigger In before hopping to next point
Short	2	Wait for Trigger In before starting hop sequence

Examples

Visual Basic
```
Mode = MyPTE1.Hop_GetTriggerIn
```
Visual C++
```
Mode = MyPTE1->Hop_GetTriggerIn();
```
Visual C#
```
Mode = MyPTE1.Hop_GetTriggerIn();
```
Matlab
```
Mode = MyPTE1.Hop_GetTriggerIn
```

See Also

Frequency/Power Hop – Get Trigger Out Mode
Frequency/Power Hop – Set Trigger In Mode
Frequency/Power Hop – Set Trigger Out Mode

2.11 (i) - Frequency/Power Hop – Get Trigger Out Mode

Declaration

```
Short Hop_GetTriggerOut()
```

Description

This function returns the Trigger Output mode for the hop sequence, this dictates how the Trigger Out port will be used during the frequency sweep:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each hop point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each hop sequence is initiated

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	[0	Command failed
Short	0	Trigger Out disabled
Short	1	Trigger Out set at each point
Short	2	Trigger Out set as the hop is initiated

Examples

Visual Basic
```
Mode = MyPTE1.Hop_GetTriggerOut
```
Visual C++
```
Mode = MyPTE1->Hop_GetTriggerOut();
```
Visual C#
```
Mode = MyPTE1.Hop_GetTriggerOut();
```
Matlab
```
Mode = MyPTE1.Hop_GetTriggerOut
```

See Also

Frequency/Power Hop – Get Trigger In Mode
Frequency/Power Hop – Set Trigger In Mode
Frequency/Power Hop – Set Trigger Out Mode

2.11 (j) - Frequency/Power Hop – Set Direction

Declaration

```
Short Hop_SetDirection(Short HopDirection)
```

Description

This function sets the direction of the hop sequence:
0 – From first to last in the list
1 – From last to first in the list
2 – From first to last in the list, then back from last to first

Parameters

Data Type	Variable	Description
Short	Hop _Direction	Required. Numeric value corresponding to the sweep direction mode: 0 – Ascending from first to last 1 – Descending from last to first 2 – Ascending from first to last, then descending from last to first

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.Hop_SetDirection(0)
```
Visual C++
```
Status = MyPTE1->Hop_SetDirection(0);
```
Visual C#
```
Status = MyPTE1.Hop_SetDirection(0);
```
Matlab
```
Status = MyPTE1.Hop_SetDirection(0)
```

See Also

Frequency/Power Hop – Get Direction

2.11 (k) - Frequency/Power Hop – Set Dwell Time

Declaration

```
Short Hop_SetDwell(Short dwell_msec)
```

Description

This function sets the dwell time in milliseconds; this is the length of time that the generator will pause at each point in the hop sequence.

Parameters

Data Type	Variable	Description
Short	dwell_msec	Required. The dwell time in milliseconds.

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.Hop_SetDwell(15)
```
Visual C++
```
Status = MyPTE1->Hop_SetDwell(15);
```
Visual C#
```
Status = MyPTE1.Hop_SetDwell(15);
```
Matlab
```
Status = MyPTE1.Hop_SetDwell(15)
```

See Also

Frequency/Power Hop – Get Dwell Time
Frequency/Power Hop – Get Maximum Dwell Time
Frequency/Power Hop – Get Minimum Dwell Time

2.11 (l) - Frequency/Power Hop – Start/Stop Hop Sequence

Declaration

```
Short Hop_SetMode(Short onoff)
```

Description

This function starts or stops the hop sequence using the previously defined parameters.

Note: The hop sequence will stop automatically if any other command is sent.

Parameters

Data Type	Variable	Description
Short	onoff	Required. Integer value to enable/disable the hop sequence: 1 – Start 0 – Stop

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetMode(1)        ' Start
Status = MyPTE1.Hop_SetMode(0)        ' Stop
```

Visual C++

```
Status = MyPTE1->Hop_SetMode(1);      // Start
Status = MyPTE1->Hop_SetMode(0);      // Stop
```

Visual C#

```
Status = MyPTE1.Hop_SetMode(1);       // Start
Status = MyPTE1.Hop_SetMode(0);       // Stop
```

Matlab

```
Status = MyPTE1.Hop_SetMode(1)        % Start
Status = MyPTE1.Hop_SetMode(0)        % Stop
```

See Also

Frequency Sweep – Start/Stop Sweep
Power Sweep – Start/Stop Sweep

2.11 (m) - Frequency/Power Hop – Set Number of Points

Declaration

```
Short Hop_SetNoOfPoints(Short NoOfPoints)
```

Description

This function sets the number of points to be used in the hop sequence.

Parameters

Data Type	Variable	Description
Short	NoOfPoints	Required. The number of points to hop

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.Hop_SetNoOfPoints(10)
```

Visual C++

```
Status = MyPTE1->Hop_SetNoOfPoints(10);
```

Visual C#

```
Status = MyPTE1.Hop_SetNoOfPoints(10);
```

Matlab

```
Status = MyPTE1.Hop_SetNoOfPoints(10)
```

See Also

Frequency/Power Hop – Get Maximum Number of Points
Frequency/Power Hop – Get Hop Point
Frequency/Power Hop – Set Number of Points
Frequency/Power Hop – Set Hop Point

2.11 (n) - Frequency/Power Hop – Set Hop Point

Declaration

```
Short Hop_SetPoint(Short PointNo, Double HopFreq, Float HopPower)
```

Description

This function sets the frequency and power for a specific point in the hop sequence.

Parameters

Data Type	Variable	Description
Short	PointNo	Required. The point number; 0 for the first point in the sequence, 1 for the second, up (n - 1) for the final point where n equals the maximum number of points.
Double	HopFreq	Required. The frequency in MHz.
Float	HopPower	Required. The power in dBm.

Return Values

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

Examples

Visual Basic
```
        Status = MyPTE1.Hop_SetPoint(3, 1000, 10)
                ' Set point 3 in the sequence to 1000MHz @ 10dBm
```
Visual C++
```
        Status = MyPTE1->Hop_SetPoint(3, 1000, 10);
                // Set point 3 in the sequence to 1000MHz @ 10dBm
```
Visual C#
```
        Status = MyPTE1.Hop_SetPoint(3, 1000, 10);
                // Set point 3 in the sequence to 1000MHz @ 10dBm
```
Matlab
```
        Status = MyPTE1.Hop_SetPoint(3, 1000, 10)
                % Set point 3 in the sequence to 1000MHz @ 10dBm
```

See Also

Frequency/Power Hop – Get Maximum Number of Points
Frequency/Power Hop – Get Hop Point
Frequency/Power Hop – Set Number of Points
Frequency/Power Hop – Set Hop Point

2.11 (o) - Frequency/Power Hop – Set Trigger In Mode

Declaration

```
Short Hop_SetTriggerIn(Short HopTriggerIn)
```

Description

This function specifies how the hop sequence should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before hopping to the next point
2 – Wait for external trigger (Trigger In = logic 1) before starting each hop sequence

Parameters

Data Type	Variable	Description
Short	HopTriggerIn	Required. Integer value to specify the Trigger In mode: 0 – Ignore Trigger In 1 - Wait for Trigger In before each hop 2 - Wait for Trigger In before starting each hop sequence

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.Hop_SetTriggerIn(1)
```
Visual C++
```
Status = MyPTE1->Hop_SetTriggerIn(1);
```
Visual C#
```
Status = MyPTE1.Hop_SetTriggerIn(1);
```
Matlab
```
Status = MyPTE1.Hop_SetTriggerIn(1)
```

See Also

Frequency/Power Hop – Get Trigger In Mode
Frequency/Power Hop – Get Trigger Out Mode
Frequency/Power Hop – Set Trigger Out Mode

2.11 (p) - Frequency/Power Hop – Set Trigger Out Mode

Declaration

```
Short Hop_SetTriggerOut(Short HopTriggerOut)
```

Description

This function specified how the Trigger Out port will be used during the hop sequence. The modes are:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) on setting each point
2 – Provide a trigger output (Trigger Out = logic 1) on commencing each hop sequence

Parameters

Data Type	Variable	Description
Short	HopTrigger_Out	Required. Integer value to specify the Trigger Out mode: 0 – Trigger Out disabled 1 – Set Trigger Out at each point 2 – Set Trigger Out on starting each hop sequence

Return Values

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

Examples

Visual Basic
```
Status = MyPTE1.Hop_SetTriggerOut(1)
```
Visual C++
```
Status = MyPTE1->Hop_SetTriggerOut(1);
```
Visual C#
```
Status = MyPTE1.Hop_SetTriggerOut(1);
```
Matlab
```
Status = MyPTE1.Hop_SetTriggerOut(1)
```

See Also

Frequency/Power Hop – Get Trigger In Mode
Frequency/Power Hop – Get Trigger Out Mode
Frequency/Power Hop – Set Trigger In Mode

2.12 - Ethernet Configuration Functions

These functions provide a means for identifying or configuring the Ethernet settings such as IP address, TCP/IP port and network gateway. They can only be called while the devices are connected via the USB interface.

2.12 (a) - Get Ethernet Configuration

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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
        h = msgbox ("IP address: ", IP1, ".", IP2, ".", IP3, ".", IP4)
        h = msgbox ("Subnet Mask: ", M1, "." & M2, "." & M3, ".", M4)
        h = msgbox ("Gateway: ", GW1, ".", GW2, ".", GW3, ".", GW4)
end
```

See Also

Get MAC Address
Get TCP/IP Port

2.12 (b) - Get IP Address

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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
        h = msgbox ("IP address: ", IP1, ".", IP2, ".", IP3, ".", IP4)
end
```

See Also

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

2.12 (c) - Get MAC Address

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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
    h = msgbox ("MAC address: ", M1, ".", M2, ".", M3, ".", M4, ".", M5,
    ".", M6)
end
```

See Also

Get Ethernet Configuration

2.12 (d) - Get Network Gateway

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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
                h = msgbox ("Gateway: ", IP1, ".", IP2, ".", IP3, ".", IP4)
        end
```

See Also

Get Ethernet Configuration
Save Network Gateway

2.12 (e) - Get Subnet Mask

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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
                h = msgbox ("Subnet mask: ", b1, ".", b2, ".", b3, ".", b4)
        end
```

See Also

Get Ethernet Configuration
Save Subnet Mask

2.12 (f) - Get TCP/IP Port

Declaration

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

Description

This function returns the TCP/IP port used by the generator 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. Integer variable which will be updated with the TCP/IP port.

Return Values

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

Example

Visual Basic

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

Visual C++

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

Visual C#

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

Matlab

```
[status, port] = MyPTE1.GetEthernet_SubNetMask(port)
if status > 0
        h = msgbox ("Port: ", port)
end
```

See Also

Get Ethernet Configuration
Save TCP/IP Port

2.12 (g) - Get DHCP Status

Declaration

```
Short GetEthernet_UseDHCP()
```

Description

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

Parameters

Data Type	Variable	Description
None		

Return Values

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

Example

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

2.12 (h) - Get Password Status

Declaration

```
Short GetEthernet_UsePWD()
```

Description

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

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
Short	0	Password not required
Short	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

2.12 (i) - Get Password

Declaration

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

Description

This function returns the current password used by the generator for HTTP/Telnet communication. The password will be returned even if the device is not currently configured to require a password.

Parameters

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

Return Values

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

Example

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
        h = msgbox ("Password: ", pwd)
end
```

See Also

Get Password Status
Use Password
Set Password

2.12 (j) - Save IP Address

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

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

2.12 (k) - Save Network Gateway

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

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

2.12 (l) - Save Subnet Mask

Declaration

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

Description

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

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

Parameters

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

Return Values

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

Example

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

2.12 (m) - Save TCP/IP Port

Declaration

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

Description

This function sets the TCP/IP port used by the generator 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
Short	0	Command failed
Short	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

2.12 (n) - Use DHCP

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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

2.12 (o) - Use Password

Declaration

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

Description

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

Parameters

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

Return Values

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

Example

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

2.12 (p) - Set Password

Declaration

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

Description

This function sets the password used by the generator for HTTP/Telnet communication. The password will not affect 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
Short	0	Command failed
Short	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

2.13 - SCPI Communication Functions

SCPI (Standard Commands for Programmable Instruments) is a common method for communicating with and controlling instrumentation products.

2.13 (a) - Send SCPI Query

Declaration

```
Short SCPI_Query(String QuerySTR, ByRef String RetSTR)
```

Description

Sends a SCPI command or query to the generator and returns the response.

Parameters

Data Type	Variable	Description
String	QuerySTR	Required. The SCPI command or query to send to the device.
String	RetSTR	Required. Variable passed by reference, to be updated with the generator'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.SCPI_Query(":REF?", RetSTR)
               ' Check which reference is in use.  For example:
               ' RetSTR equals "EXT:10MHz" if external 10MHz ref in use
```

Visual C++

```
Status = MyPTE1->SCPI_Query(":REF?", RetSTR);
               // Check which reference is in use.  For example:
               // RetSTR equals "EXT:10MHz" if external 10MHz ref in use
```

Visual C#

```
Status = MyPTE1.SCPI_Query(":REF?", RetSTR);
               // Check which reference is in use.  For example:
               // RetSTR equals "EXT:10MHz" if external 10MHz ref in use
```

Matlab

```
[Status, RetSTR] = MyPTE1.SCPI_Query(":REF?", RetSTR)
               % Check which reference is in use.  For example:
               % RetSTR equals "EXT:10MHz" if external 10MHz ref in use
```

See Also

Send SCPI Command
SCPI Functions

2.13 (b) - Send SCPI Command

Declaration

```
Short SCPI_Command(String CommandSTR)
```

Description

Sends a SCPI command to the generator.

Parameters

Data Type	Variable	Description
String	CommandSTR	Required. The SCPI command to send to the device.

Return Values

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

Examples

Visual Basic

```
Status = MyPTE1.SCPI_Command(":PWR:RF:ON")
               ' Enable RF output
```

Visual C++

```
Status = MyPTE1->SCPI_Command(":PWR:RF:ON");
               // Enable RF output
```

Visual C#

```
Status = MyPTE1.SCPI_Command(":PWR:RF:ON");
               // Enable RF output
```

Matlab

```
Status = MyPTE1.SCPI_Command(":PWR:RF:ON")
               % Enable RF output
```

See Also

Send SCPI Query
Send SCPI Command
SCPI Functions

3 - Operating in a Linux Environment via USB

To open a connection to Mini-Circuits Signal Generators (SSG Series), the Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID: 0x20CE
- Signal generator Product ID: 0x12

Communication with the signal generator 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 signal generator.

Following a successful operation, the first byte of the returned array will mirror the code sent in the first byte of the transmit array.

A worked example is included in the Programming Manual - Appendices. The example uses the libhid and libusb libraries to interface with the signal generator as a USB HID (Human Interface Device).

3.1 - Summary of Commands

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

3.1 (a) - Common Functions

#	Description	Command (Byte 0)
a	Get Device Model Name	40
b	Get Device Serial Number	41
c	Set Frequency and Power	103
d	Set Frequency	101
e	Set Power	102
f	Set RF Power On/Off	104
g	Get Generator Output Status	105
h	Get Generator Minimum Frequency	42
i	Get Generator Maximum Frequency	43
j	Get Generator Step Size Spec	44
k	Get Generator Minimum Power Spec	45
l	Get Generator Maximum Power Spec	46
m	Check External Reference	47
n	Get Firmware	99
o	Send SCPI Command	121

3.1 (b) - Modulation Functions

#	Description	Command (Byte 0)
a	Set Pulse Mode	117
b	Set Triggered Pulse Mode	118
c	Set External Pulse Modulation Mode	128

3.1 (c) - Frequency/Power Hop Functions

#	Description	Command (Byte 0)	Command (Byte 1)
a	Get Hop Direction	205	7
b	Get Hop Dwell Time	205	4
c	Get Maximum Hop Dwell Time	205	5
d	Get Minimum Hop Dwell Time	205	6
e	Get Maximum Number of Hop Points	205	10
f	Get Number of Hop Points	205	0
g	Get Specific Hop Setting	205	1
h	Get Hop Trigger-In Mode	205	8
i	Get Hop Trigger-Out Mode	205	9
j	Set Hop Direction	204	5
k	Set Hop Dwell Time	204	4
l	Start/Stop Hop Sequence	204	8
m	Set Number of Hop Points	204	0
n	Set Specific Hop Parameters	204	1
o	Set Hop Trigger-In Mode	204	6
p	Set Hop Trigger-Out Mode	204	7

3.1 (d) - Frequency Sweep Functions

#	Description	Command (Byte 0)	Command (Byte 1)
a	Get Sweep Direction	201	7
b	Get Sweep Dwell Time	201	4
c	Get Maximum Sweep Dwell Time	201	5
d	Get Minimum Sweep Dwell Time	201	6
e	Get Sweep Power	201	3
f	Get Sweep Start Frequency	201	0
g	Get Sweep Stop Frequency	201	1
h	Get Sweep Step Size	201	2
i	Get Sweep Trigger-In Mode	201	8
j	Get Sweep Trigger-Out Mode	201	9
k	Set Sweep Direction	200	5
l	Set Sweep Dwell Time	200	4
m	Start/Stop Sweep Sequence	200	8
n	Set Sweep Power	200	3
o	Set Sweep Start Frequency	200	0
p	Set Sweep Stop Frequency	200	1
q	Set Sweep Step Size	200	2
r	Set Sweep Trigger-In Mode	200	6
s	Set Sweep Trigger-Out Mode	200	7

3.1 (e) - Power Sweep Functions

#	Description	Command (Byte 0)	Command (Byte 1)
a	Get Sweep Direction	203	7
b	Get Sweep Dwell Time	203	4
c	Get Maximum Sweep Dwell Time	203	5
d	Get Minimum Sweep Dwell Time	203	6
e	Get Sweep Frequency	203	3
f	Get Sweep Start Power	203	0
g	Get Sweep Stop Power	203	1
h	Get Sweep Power Step Size	203	2
i	Get Sweep Trigger-In Mode	203	8
j	Get Sweep Trigger-Out Mode	203	9
k	Set Sweep Direction	202	5
l	Set Sweep Dwell Time	202	4
m	Start/Stop Sweep Sequence	202	8
n	Set Sweep Frequency	202	3
o	Set Sweep Start Power	202	0
p	Set Sweep Stop Power	202	1
q	Set Sweep Power Step Size	202	2
r	Set Sweep Trigger-In Mode	202	6
s	Set Sweep Trigger-Out Mode	202	7

3.1 (f) - Ethernet Configuration Functions

#	Description	Command (Byte 0)	Command (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 - Common Commands

These common functions apply to all models in the Mini-Circuits SSG signal generator series unless otherwise stated.

3.2 (a) - Get Device Model Name

Description

Returns the full Mini-Circuits part number of the signal generator.

Transmit Array

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

Returned Array

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

Example

The following array would be returned for Mini-Circuits’ SSG-4000HP signal generator. See Appendix A for conversions between decimal, binary and ASCII characters.

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Char 1	Char 2	Char 3	Char 4	Char 5
Value	40	83	83	71	45	52
ASCII Character	N/A	S	S	G	-	4

Byte	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
Description	Char 6	Char 7	Char 8	Char 9	Char 10	End Marker
Value	48	48	48	72	80	0
ASCII Character	0	0	0	H	P	N/A

See Also

Get Device Serial Number

3.2 (b) - Get Device Serial Number

Description

Returns the serial number of the connected signal generator.

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 current signal generator has serial number 1100040023. See Appendix A for conversions between decimal, binary and ASCII characters.

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Char 1	Char 2	Char 3	Char 4	Char 5
Value	41	49	49	48	48	48
ASCII Character	N/A	1	1	0	0	0

Byte	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
Description	Char 6	Char 7	Char 8	Char 9	Char 10	End Marker
Value	52	48	48	50	51	0
ASCII Character	4	0	0	2	3	N/A

See Also

Get Device Model Name

3.2 (c) - Set Frequency and Power

Description

Sets the RF output frequency and power level of the signal generator and enables or disables the "trigger out" function.

Transmit Array

Byte	Data	Description
0	103	Interrupt code for Set Frequency and Power
1	Freq_1	Frequency (Hz), split over 5 bytes BYTE1 = INT (Freq / 256^4)
2	Freq_2	Frequency (Hz), split over 5 bytes: REMAINDER1 = Freq - BYTE1 * (256^4) BYTE2 = INT (REMAINDER1 / 256^3)
3	Freq_3	Frequency (Hz), split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE2 * (256^3) BYTE3 = INT (REMAINDER2 / 256^2)
4	Freq_4	Frequency (Hz), split over 5 bytes REMAINDER3 = REMAINDER2 - BYTE3 * (256^2) BYTE4 = INT (REMAINDER3 / 256)
5	Freq_5	Frequency (Hz), split over 5 bytes BYTE5 = INT (REMAINDER3 - BYTE4 * 256)
6	Power (+/-)	Power polarity: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
7	Power_1	Power magnitude (dBm), split over 2 bytes: BYTE7 = INT (Magnitude * 100 / 256)
8	Power_2	Power magnitude (dBm), split over 2 bytes: BYTE8 = Magnitude * 100 - (BYTE7 * 256)
9	Trigger_Out	Trigger Out status: 0 = Disable trigger output 1 = Enable trigger output
10 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	103	Interrupt code for Set Frequency and Power
1 - 63	Not significant	"Don't care" bytes, can be any value

Example

The following transmit array would set the generator output to 5501.56MHz with a power level of +4.5dBm and enable the Trigger Out:

Byte	Data	Description	
0	103	Interrupt code for Set Frequency and Power	
1	1	BYTE1	= INT (5,501,560,000 / 256^4) = INT (1.2809) = 1
2	71	REMAINDER1 BYTE2	= 5,501,560,000 - 1 * (256^4) = 1,206,592,704 = INT (1,206,592,704 / 256^3) = 71
3	235	REMAINDER2 BYTE3	= 1,206,592,704 - 71 * (256^3) = 60,196 = INT (60,196 / 256^2) = 235
4	36	REMAINDER3 BYTE4	= 1,206,592,704 - 71 * (256^2) = 9,408 = INT (9,408 / 256) = 36
5	192	BYTE5	= INT (9,408 - (36 * 256)) = 192
6	0	Power level is positive (dBm)	
7	1	BYTE7	= INT ((4.5 * 100) / 256) = INT (1.76) = 1
8	194	BYTE8	= (4.5 *100) - (1 * 256) = 194
9	1	Enable Trigger Out	
10 - 63	Not used	"Don't care" bytes, can be any value	

See Also

Set RF Power On/Off
Get Generator Output Status

3.2 (e) - Set Frequency

Description

Sets the RF output frequency of the signal generator and enables or disables the "trigger out" function.

Transmit Array

Byte	Data	Description
0	101	Interrupt code for Set Frequency
1	Freq_1	Frequency (Hz), split over 5 bytes BYTE1 = INT (Freq / 256^4)
2	Freq_2	Frequency (Hz), split over 5 bytes: REMAINDER1 = Freq - BYTE1 * (256^4) BYTE2 = INT (REMAINDER1 / 256^3)
3	Freq_3	Frequency (Hz), split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE2 * (256^3) BYTE3 = INT (REMAINDER2 / 256^2)
4	Freq_4	Frequency (Hz), split over 5 bytes REMAINDER3 = REMAINDER2 - BYTE3 * (256^2) BYTE4 = INT (REMAINDER3 / 256)
5	Freq_5	Frequency (Hz), split over 5 bytes BYTE5 = INT (REMAINDER3 - BYTE4 * 256)
6	Trigger_Out	Trigger Out status: 0 = Disable trigger output 1 = Enable trigger output
7 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	101	Interrupt code for Set Frequency
1 - 63	Not significant	"Don't care" bytes, can be any value

Example

The following transmit array would set the generator output to 4100.55MHz and enable Trigger Out:

Byte	Data	Description	
0	10	Interrupt code for Set Frequency	
1	0	BYTE1	= INTEGER (4,100,550,000 / 256 ^ 4) = INTEGER (0.9547) = 0
2	244	REMAINDER1 BYTE2	= 4,100,550,000 - 0 * (256 ^ 4) = 4,100,550,000 = INTEGER (4,100,550,000 / 256 ^ 3) = 244
3	105	REMAINDER2 BYTE3	= 4,100,550,000 - 244 * (256 ^ 3) = 6,909,296 = INTEGER (6,909,296 / 256 ^ 2) = 105
4	109	REMAINDER3 BYTE4	= 6,909,296 - 105 * (256 ^ 2) = 28,016 = INTEGER (28,016 / 256) = 109
5	112	BYTE5	= INTEGER (28,016 - (109 * 256)) = 112
6	1	Enable Trigger Out	
7 - 63	Not used	"Don't care" bytes, can be any value	

See Also

Set Power
Set RF Power On/Off
Get Generator Output Status

3.2 (f) - Set Power

Description

Sets the RF output power of the signal generator and enables or disables the "trigger out" function.

The transmit array is made up of the following bytes:

- BYTE0
    - 102 (code for Set Power)
- BYTE1
    - 1 (to set a negative power value) or 0 (to set a positive power value)
- BYTE2 to BYTE3
    - Absolute power in dBm multiplied by 100 (to allow fine resolution)
    - The value is split into MSB (BYTE2) and LSB (BYTE3)
    - BYTE2 = INTEGER VALUE ((ABSOLUTE POWER * 100) / 256)
    - BYTE3 = (ABSOLUTE POWER * 100) - (BYTE2 * 256)
- BYTE4
    - 1 (to enable Trigger Out) or (0 to disable Trigger Out)
- BYTE5 to BYTE63
    - Can be any value ("don't care" bytes)

Transmit Array

Byte	Data	Description
0	102	Interrupt code for Set Power
1	Power (+/-)	Power polarity: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
2	Power_1	Power magnitude (dBm), split over 2 bytes: BYTE2 = INT (Magnitude * 100 / 256)
3	Power_2	Power magnitude (dBm), split over 2 bytes: BYTE3 = Magnitude * 100 - (BYTE2 * 256)
4	Trigger_Out	Trigger Out status: 0 = Disable trigger output 1 = Enable trigger output
5 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	102	Interrupt code for Set Power
1 - 63	Not significant	"Don't care" bytes, can be any value

Example

To following transmit array would set the generator output power to -5.5dBm and enable the Trigger Out:

Byte	Data	Description
0	102	Interrupt code for Set Power
1	1	Power value is negative
2	2	BYTE2 = INT ((5.5 * 100) / 256) = INT (2.15) = 2
3	38	BYTE3 = (5.5 *100) - (2 * 256) = 38
4	1	Enable Trigger Out
5 - 63	Not used	"Don't care" bytes, can be any value

See Also

Set Frequency and Power
Set Frequency
Set RF Power On/Off
Get Generator Output Status

3.2 (g) - Set RF Power On/Off

Description

This function enables or disables the RF output of the signal generator.

Send code 104 in BYTE0 of the transmit array with BYTE1 as 1 to enable or 0 to disable the RF output. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array contains 104 in BYTE0. BYTE1 to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	Data	Description
0	104	Interrupt code for Set RF Power On/Off
1	Status	RF output status: 0 = Disable RF output 1 = Enable RF output
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	104	Interrupt code for Set RF Power On/Off

Example

The below transmit array enables the RF output with the previously defined frequency and power levels:

Byte	Data	Description
0	104	Interrupt code for Set RF Power On/Off
1	1	Enable the RF output

The below transmit array disables the RF output:

Byte	Data	Description
0	104	Interrupt code for Set RF Power On/Off
1	0	Disable the RF output

See Also

Set Frequency and Power
Set Power
Get Generator Output Status

3.2 (h) - Get Generator Output Status

Description

Returns the current output status of the signal generator, covering the following parameters:

- Generator lock status (locked/unlocked)
- RF output status (on/off)
- Current output frequency
- Current output power
- Current output power relative to user requested level

Transmit Array

Byte	Data	Description
0	105	Interrupt code for Get Generator Output Status
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	105	Interrupt code for Get Generator Output Status
1	Output Status	0 = RF output is disabled 1 = RF output is enabled
2	Lock Status	0 = Frequency is not locked 1 = Frequency is locked
3	Freq_1	Frequency (Hz), split over 5 bytes: Frequency = (256^4) * BYTE3 + (256^3) * BYTE4 + (256^2) * BYTE5 + 256 * BYTE6 + BYTE7
4	Freq_2	Frequency (Hz), split over 5 bytes
5	Freq_3	Frequency (Hz), split over 5 bytes
6	Freq_4	Frequency (Hz), split over 5 bytes
7	Freq_5	Frequency (Hz), split over 5 bytes
8	Power (+/-)	Power polarity: 0 = Positive (Power = 1 * Magnitude) 1 = Negative (Power = -1 * Magnitude)
9	Power_1	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * BYTE9 + BYTE10) / 100
10	Power_2	Power magnitude (dBm), split over 2 bytes
11	Unlevel_High	High power request warning: 0 = User requested power level within generator capability 1 = User requested power level is higher than the generator can achieve
12	Unlevel_Low	Low power request warning: 0 = User requested power level within generator capability 1 = User requested power level is lower than the generator can achieve
13 - 63	Not used	"Don't care" bytes, could be any value

Example

The following returned array indicates that the generator was set with the output enabled at 4980.50MHz, +5.5dBm and the power level is within the generator's capability:

Byte	Data	Description
0	105	Interrupt code for Get Generator Output Status
1	1	RF output enabled
2	1	Frequency is locked
3	1	Frequency (Hz), split over 5 bytes: Frequency = 256^4 * 1 + 256^3 * 40 + 256^2 * 220 + 256 * 102 + 32 = 751,250,000 Hz = 751.25 MHz
4	40	Frequency (Hz), split over 5 bytes
5	220	Frequency (Hz), split over 5 bytes
6	102	Frequency (Hz), split over 5 bytes
7	32	Frequency (Hz), split over 5 bytes
8	0	Power value is positive
9	2	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * 2 + 38) / 100 = 5.5dBm
10	38	Power magnitude (dBm), split over 2 bytes
11	0	Power level within range
12	0	Power level within range

See Also

Set Frequency and Power
Set Power
Set RF Power On/Off

3.2 (i) - Get Generator Minimum Frequency

Description

Returns the signal generator minimum frequency specification in Hz.

Transmit Array

Byte	Data	Description
0	42	Interrupt code for Get Generator Minimum Frequency
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	42	Interrupt code for Get Generator Minimum Frequency
1	Freq_1	Frequency (Hz), split over 4 bytes: Frequency = (256^3) * BYTE1 + (256^2) * BYTE2 + 256 * BYTE3 + BYTE4
2	Freq_2	Frequency (Hz), split over 4 bytes
3	Freq_3	Frequency (Hz), split over 4 bytes
4	Freq_4	Frequency (Hz), split over 4 bytes
5 - 63	Not Used	"Don't care" bytes, could be any value

Example

The following array would be returned for SSG-4000HP:

Byte	Data	Description
0	42	Interrupt code for Get Generator Minimum Frequency
1	14	Frequency (Hz), split over 4 bytes: FREQUENCY = 256^3 * 14 + 256^2 * 230 + 256 * 178 + 128 = 250,000,000 Hz = 250 MHz
2	220	Frequency (Hz), split over 4 bytes
3	178	Frequency (Hz), split over 4 bytes
4	128	Frequency (Hz), split over 4 bytes

See Also

Get Generator Maximum Frequency
Get Generator Step Size
Get Generator Minimum Power
Get Generator Maximum Power

3.2 (j) - Get Generator Maximum Frequency

Description

Returns the maximum frequency specification in Hz.

Transmit Array

Byte	Data	Description
0	43	Interrupt code for Get Generator Maximum Frequency
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	43	Interrupt code for Get Generator Maximum Frequency
1	Freq_1	Frequency (Hz), split over 5 bytes: Frequency = 256^4 * BYTE1 + 256^3 + BYTE2 + 256^2 * BYTE3 + 256 * BYTE4 + BYTE5
2	Freq_2	Frequency (Hz), split over 5 bytes
3	Freq_3	Frequency (Hz), split over 5 bytes
4	Freq_4	Frequency (Hz), split over 5 bytes
5	Freq_5	Frequency (Hz), split over 5 bytes
6 - 63	Not Used	"Don't care" bytes, could be any value

Example

The following array would be returned for SSG-6001RC:

Byte	Data	Description
0	43	Interrupt code for Get Generator Maximum Frequency
1		Frequency (Hz), split over 5 bytes: Frequency = 256^4 * 1 + 256^3 * 101 + 256^2 * 160 + 256 * 188 + 0 = 6,000,000,000 Hz = 6,000 MHz
2		Frequency (Hz), split over 5 bytes
3		Frequency (Hz), split over 5 bytes
4		Frequency (Hz), split over 5 bytes
5		Frequency (Hz), split over 5 bytes

See Also

Get Generator Minimum Frequency
Get Generator Step Size
Get Generator Minimum Power
Get Generator Maximum Power

3.2 (k) - Get Generator Step Size

Description

Returns the signal generator's minimum step size in Hz.

Transmit Array

Byte	Data	Description
0	44	Interrupt code for Get Generator Step Size
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	44	Interrupt code for Get Generator Step Size
1	Freq_1	Frequency (Hz), split over 4 bytes: Frequency = (256^3) * BYTE1 + (256^2) * BYTE2 + 256 * BYTE3 + BYTE4
2	Freq_2	Frequency (Hz), split over 4 bytes
3	Freq_3	Frequency (Hz), split over 4 bytes
4	Freq_4	Frequency (Hz), split over 4 bytes
5 - 63	Not Used	"Don't care" bytes, could be any value

Example

The following array would be returned for SSG-4000HP:

Byte	Data	Description
0	44	Interrupt code for Get Generator Step Size
1	0	Frequency (Hz), split over 4 bytes: FREQUENCY = 256^3 * 0 + 256^2 * 0 + 256 * 19 + 136 = 5,000 Hz = 5 KHz
2	0	Frequency (Hz), split over 4 bytes
3	19	Frequency (Hz), split over 4 bytes
4	136	Frequency (Hz), split over 4 bytes

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency
Get Generator Minimum Power
Get Generator Maximum Power

3.2 (l) - Get Generator Minimum Power

Description

Returns the generator's minimum output power specification in dBm. The minimum output power achievable by the generator is guaranteed to be at least as low as this specified level across the full operating frequency and will be even lower in some frequency bands.

Transmit Array

Byte	Data	Description
0	45	Interrupt code for Get Generator Minimum Power
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	45	Interrupt code for Get Generator Minimum Power
1	Power (+/-)	Power polarity: 0 = Positive (Power = 1 * Magnitude) 1 = Negative (Power = -1 * Magnitude)
2	Power_1	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * BYTE2 + BYTE3) / 100
3	Power_2	Power magnitude (dBm), split over 2 bytes
4 - 63	Not used	"Don't care" bytes, could be any value

Example

The following array would be returned for SSG-4000HP:

Byte	Data	Description
0	45	Interrupt code for Get Generator Minimum Power
1	1	Power value is negative
2	19	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * 19 + 136) / 100 = 50 Power = -50 dBm
3	136	Power magnitude (dBm), split over 2 bytes

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency
Get Generator Step Size
Get Generator Maximum Power

3.2 (m) - Get Generator Maximum Power

Description

Returns the generator's maximum output power specification in dBm. The maximum output power achievable by the generator is guaranteed to be at least as high as this specified level across the full operating frequency and will be even higher in some frequency bands.

Transmit Array

Byte	Data	Description
0	46	Interrupt code for Get Generator Maximum Power
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	46	Interrupt code for Get Generator Maximum Power
1	Power (+/-)	Power polarity: 0 = Positive (Power = 1 * Magnitude) 1 = Negative (Power = -1 * Magnitude)
2	Power_1	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * BYTE2 + BYTE3) / 100
3	Power_2	Power magnitude (dBm), split over 2 bytes
4 - 63	Not used	"Don't care" bytes, could be any value

Example

The following array would be returned for SSG-4000HP:

Byte	Data	Description
0	46	Interrupt code for Get Generator Maximum Power
1	0	Power value is negative
2	7	Power magnitude (dBm), split over 2 bytes: Magnitude = (256 * 7 + 208) / 100 = 20 Power = +20dBm
3	208	Power magnitude (dBm), split over 2 bytes

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency
Get Generator Step Size
Get Generator Minimum Power

3.2 (n) - Check External Reference

Description

Indicates which reference source is currently in use. The signal generator will automatically switch from internal to external reference if a valid signal is detected at the Ref In port.

Transmit Array

Byte	Data	Description
0	47	Interrupt code for Check External Reference
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	47	Interrupt code for Check External Reference
1	Ref_Source	The reference source: 0 = Internal reference in use 1 = External reference in use

Example

The below returned array indicates an external reference source is connected at the signal generator's Ref In port and is currently in use:

Byte	Data	Description
0	47	Interrupt code for Check External Reference
1	1	External reference in use

3.2 (o) - Get Firmware

Description

Returns the internal firmware version of the signal generator.

Transmit Array

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

Returned Array

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

Example

The following returned array indicates that the signal generator has firmware version C3:

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

3.2 (p) - Send SCPI Command

Description

Sends a SCPI command to the signal generator 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	121	Interrupt code for Send SCPI Command
1	SCPI_Length	The length (number of ASCII characters) of the SCPI string to send
2 to 63	SCPI Transmit String	The SCPI command to be sent represented as a series of ASCII character codes, one character code per byte

Returned Array

Byte	Data	Description
0	121	Interrupt code for Send SCPI Command
1	SCPI_Length	The length (number of ASCII characters) of the SCPI command sent in the transmit array
2 to 7	Transmit_Array	Bytes 2 to 7 of the transmit array repeated
8 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, could be any value

Example 1 (Get Model Name)

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 2 to 5 of the transmit array as follows (see Appendix A for conversions between decimal, binary and ASCII characters):

Byte	Data	Description
0	121	Interrupt code for Send SCPI Command
1	4	Length of the SCPI command (four ASCII characters)
2	49	ASCII character code for :
3	77	ASCII character code for M
4	78	ASCII character code for N
5	63	ASCII character code for ?

The returned array for SSG-6001RC would be as follows:

Byte	Data	Description
0	121	Interrupt code for Send SCPI Command
1	4	Length of the SCPI command (four ASCII characters) from the transmit array
2	49	Repeated from the transmit array (ASCII character code)
3	77	Repeated from the transmit array (ASCII character code)
4	78	Repeated from the transmit array (ASCII character code)
5	63	Repeated from the transmit array (ASCII character code)
6	0	Repeated from the transmit array (unused byte)
7	0	Repeated from the transmit array (unused byte)
8	83	ASCII character code for S
9	83	ASCII character code for S
10	81	ASCII character code for G
11	45	ASCII character code for -
12	54	ASCII character code for 6
13	48	ASCII character code for 0
14	48	ASCII character code for 0
15	49	ASCII character code for 1
16	82	ASCII character code for R
17	67	ASCII character code for C
18	0	Zero value byte to indicate end of string

See Also

SCPI Functions

3.3 - Modulation Functions

The signal generator can be configured to produce a pulsed output, either in response to an external trigger or continuously using the generator's internal timing systems.

Full details of the commands for configuring a pulsed output are covered in the following sections.

3.3 (a) - Set Pulse Mode

Description

Creates a pulsed output with a user specified pulse duration and time period. The output during the pulse "on" period is a CW signal with a frequency and power level which should be set by the user in advance. The pulse period will repeat indefinitely until any other command is received by the signal generator.

Transmit Array

Byte	Data	Description
0	117	Interrupt code for Set Pulse Mode
1	Off_Time	The pulse off time split over 2 bytes: BYTE1 = INT (Off_Time / 256)
2	Off_Time	The pulse off time split over 2 bytes: BYTE2 = Off_Time - (BYTE1 * 256)
3	On_Time	The pulse duration (on time) split over 2 bytes: BYTE3 = INT (On_Time / 256)
4	On_Time	The pulse duration (on time) split over 2 bytes: BYTE4 = On_Time - (BYTE3 * 256)
5	Time_Units	Units for On_Time and Off_Time: 0 = microseconds (µs) 1 = milliseconds (ms)
6 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	117	Interrupt code for Set Pulse Mode

Example

After configuring the frequency and power of the CW signal to be used during the pulse "on" time, to enable the pulsed output with an on period of 10µs and off period of 1000µs, use the below transmit array:

Byte	Data	Description
0	117	Interrupt code for Set Pulse Mode
1	3	Off_Time = 1000µs BYTE1 = INT (1000 / 256) = 3
2	232	Off_Time = 1000µs BYTE2 = 1000 - (3 * 256) = 232
3	0	On_Time = 10µs BYTE3 = INT (10 / 256) = 0
4	10	On_Time = 10µs BYTE4 = 10 - (0 * 256) = 10
5	0	On_Time and Off_Time are expressed in microseconds (µs)

See Also

Set Frequency and Power
Set Triggered Pulse Mode
Set External Pulse Modulation Mode

3.3 (b) - Set Triggered Pulse Mode

Description

Configures a pulsed output with user specified pulse duration that will start when an external trigger is received at the "Trigger In" input. The output during the pulse "on" period is a CW signal with a frequency and power level which should be set by the user in advance. The pulsed output will be enabled until any other command is received by the signal generator.

Transmit Array

Byte	Data	Description
0	118	Interrupt code for Set Triggered Pulse Mode
1	Trigger_Type	The trigger type to use: 0 = Trigger on the falling edge 1 = Trigger on the rising edge
2	0	Zero value byte
3	On_Time (MSB)	The pulse duration (on time) split over 2 bytes: BYTE3 = INT (On_Time / 256)
4	On_Time (LSB)	The pulse duration (on time) split over 2 bytes: BYTE4 = On_Time - (BYTE3 * 256)
5	Time_Units	Units for On_Time and Off_Time: 0 = microseconds (µs) 1 = milliseconds (ms)
6 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	118	Interrupt code for Set Triggered Pulse Mode

Example

The below transmit array will enable a pulsed output with 25µs, to be triggered at the falling edge of an external trigger:

Byte	Data	Description
0	118	Interrupt code for Set Triggered Pulse Mode
1	0	Trigger on the falling edge
2	0	Zero value byte
3	0	On_Time = 25µs BYTE3 = INT (25 / 256) = 0
4	25	On_Time = 25µs BYTE4 = 25 - (0 * 256) = 25
5	0	On_Time and Off_Time are expressed in microseconds (µs)

See Also

Set Frequency and Power
Set Pulse Mode
Set External Pulse Modulation Mode

3.3 (c) - Set External Pulse Modulation Mode

Description

Enables a pulsed output in response to the generator's Trigger In port. The generator's CW output will be enabled with the specified frequency and power while the Trigger In port is held high. The output will be disabled while the Trigger In port is held low. The CW frequency and power for the "on" period should be set in advance and the external pulse modulation mode will be disabled when any other command is sent to the generator.

Transmit Array

Byte	Data	Description
0	128	Interrupt code for Set External Pulse Modulation Mode
1 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	128	Interrupt code for Set External Pulse Modulation Mode

Example

After configuring the frequency and power of the CW signal to be used during the pulse "on" time, external modulation is enable with the below transmit array:

Byte	Data	Description
0	128	Interrupt code for Set External Pulse Modulation Mode

See Also

Set Frequency and Power
Set Pulse Mode
Set Triggered Pulse Mode

3.4 - Frequency/Power Hop Functions

These functions define the frequency and power hop capabilities of the generators. The signal generator can be configured to automatically hop through a series of user defined frequency and power outputs using the generator's internal timing systems. The user stores the parameters of the hop sequence in the generator's memory and can then enable/disable the output as required.

Full details of the specific commands are covered in the following sections.

3.4 (a) - Frequency/Power Hop - Get Hop Direction

Description

Returns the direction that the generator will run through the list of hop values.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	7	Code for Get Hop Direction
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_Direction	The direction set for the current hop sequence: 0 = From first to last in the list 1 = From last to first in the list 2 = From first to last in the list, then back from last to first

Example

The below returned array indicates that the signal generator will hop bi-directionally through the list, from start to finish, then back to start:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	2	The generator will set the frequency/power values in the hop list from first to last, then last to first

See Also

Frequency/Power Hop - Set Hop Direction

3.4 (b) - Frequency/Power Hop - Get Hop Dwell Time

Description

Returns the dwell time to be used by the generator between each frequency/power hop point in the sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	4	Code for Get Hop Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_Dwell	The dwell time in milliseconds split over 2 bytes: Hop_Dwell = (256 * BYTE1) + BYTE2
2	Hop_Dwell	The dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's dwell time is set to 300ms:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	1	The dwell time in milliseconds split over 2 bytes: Hop_Dwell = (256 * 1) + 44 = 300 ms
2	44	The dwell time in milliseconds split over 2 bytes

See Also

Frequency/Power Hop - Get Maximum Hop Dwell Time
Frequency/Power Hop - Get Minimum Hop Dwell Time
Frequency/Power Hop - Set Hop Dwell Time

3.4 (c) - Frequency/Power Hop - Get Maximum Hop Dwell Time

Description

Returns the maximum allowed dwell time in milliseconds for any point in a hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	5	Code for Get Maximum Hop Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes: Hop_MaxDwell = (256 * BYTE1) + BYTE2
2	Hop_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's maximum dwell time is 10,000ms (10s):

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	39	The maximum dwell time in milliseconds split over 2 bytes: Hop_MaxDwell = (256 * 39) + 16 = 10,000 ms
2	16	The maximum dwell time in milliseconds split over 2 bytes

See Also

Frequency/Power Hop - Get Hop Dwell Time
Frequency/Power Hop - Get Minimum Hop Dwell Time

3.4 (d) - Frequency/Power Hop - Get Minimum Hop Dwell Time

Description

Returns the minimum allowed dwell time in milliseconds for any point in a hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	6	Code for Get Minimum Hop Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_MinDwell	The minimum dwell time in milliseconds split over 2 bytes: Hop_MinDwell = (256 * BYTE1) + BYTE2
2	Hop_MinDwell	The minimum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's minimum dwell time is 20ms:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	0	The minimum dwell time in milliseconds split over 2 bytes: Hop_MinDwell = (256 * 0) + 20 = 20 ms
2	20	The minimum dwell time in milliseconds split over 2 bytes

See Also

Frequency/Power Hop - Get Hop Dwell Time
Frequency/Power Hop - Get Maximum Hop Dwell Time

3.4 (e) - Frequency/Power Hop - Get Maximum Number of Hop Points

Description

Returns the maximum number of points allowed in the hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	10	Code for Get Maximum Number of Hop Points
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_MaxPoints	The maximum number of points allowed in the hop sequence

Example

The below returned array indicates that the maximum number of hop points allowed is 60,000:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	60000	The maximum number of hop points allowed is 60,000

See Also

Frequency/Power Hop - Get Number of Hop Points
Frequency/Power Hop - Set Number of Hop Points

3.4 (f) - Frequency/Power Hop - Get Number of Hop Points

Description

Returns the number of points specified for the current the hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	0	Code for Get Number of Hop Points
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Hop_Points	The number of points in the current hop sequence

Example

The below returned array indicates that the current hop is configured for 100 points:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	100	The current hop sequence has 100 points

See Also

Frequency/Power Hop - Get Maximum Number of Hop Points
Frequency/Power Hop - Set Number of Hop Points

3.4 (g) - Frequency/Power Hop - Get Specific Hop Setting

Description

Returns the frequency and power setting for a specified point within the hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	1	Code for Get Specific Hop Setting
2	Hop_Point	Index number of the hop point, from 0 to (n - 1) where n equals the number of points specified for the hop
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Freq (1st Byte)	Frequency (Hz) of the hop point split over 5 bytes: Freq = (256^4) * BYTE1 + (256^3) * BYTE2 + (256^2) * BYTE3 + (256) * BYTE4 + BYTE5
2	Freq (2nd Byte)	Frequency (Hz) of the hop point split over 5 bytes
3	Freq (3rd Byte)	Frequency (Hz) of the hop point split over 5 bytes
4	Freq (4th Byte)	Frequency (Hz) of the hop point split over 5 bytes
5	Freq (5th Byte)	Frequency (Hz) of the hop point split over 5 bytes
6	Power (+/-)	Power polarity of the hop point: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
7	Power_Mag (1st Byte)	Power magnitude (dBm) of the hop point split over 2 bytes: Power_Mag = (256 * BYTE7 + BYTE8) / 100
8	Power_Mag (2nd Byte)	Power magnitude (dBm) of the hop point split over 2 bytes

Example

Send the below transmit array to query the frequency/power settings of point 3 in the hop list:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	1	Code for Get Specific Hop Setting
2	3	Query point 3 in the hop list

The below example array is returned:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	0	Frequency (Hz) of the hop point split over 5 bytes: Freq = (256^4) * 0 + (256^3) * 203 + (256^2) * 100 + (256) * 250 + 45 = 3,412,392,493 Hz
2	203	Frequency (Hz) of the hop point split over 5 bytes
3	100	Frequency (Hz) of the hop point split over 5 bytes
4	250	Frequency (Hz) of the hop point split over 5 bytes
5	45	Frequency (Hz) of the hop point split over 5 bytes
6	1	Power value is negative
7	10	Power magnitude (dBm) of the hop point split over 2 bytes: Power_Mag = (256 * 10 + 105) / 100 = 26.65 dBm Power = (-1) * Power_Mag = -26.65 dBm
8	105	Power magnitude (dBm) of the hop point split over 2 bytes

The above example indicates that hop point 3 is set for 3,412,392,493 Hz (3412.392493 MHz) at -26.65 dBm.

See Also

Frequency/Power Hop - Get Number of Hop Points
Frequency/Power Hop - Set Specific Hop Parameters

3.4 (h) - Frequency/Power Hop - Get Hop Trigger-In Mode

Description

Returns the trigger-in mode which specifies how the generator will respond to an external trigger during the hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	8	Code for Get Hop Trigger-In Mode
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each hop point 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the hop sequence

Example

The below returned array indicates that the generator is configured to wait for an external trigger input before setting each point in the hop sequence:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	1	Wait for Trigger-In at each hop point

See Also

Frequency/Power Hop - Get Hop Trigger-Out Mode
Frequency/Power Hop - Set Hop Trigger-In Mode
Frequency/Power Hop - Set Hop Trigger-Out Mode

3.4 (i) - Frequency/Power Hop - Get Hop Trigger-Out Mode

Description

Returns the trigger-out mode which specifies when the generator will provide an external trigger signal during the hop sequence.

Transmit Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	9	Code for Get Hop Trigger-Out Mode
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	Trigger_Mode	The trigger-out mode: 0 = Disable trigger output 1 = Set trigger output (logic 1) on setting each hop point 2 = Set trigger output (logic 1) only at the start of the hop sequence

Example

The below returned array indicates that the generator is configured to produce an external trigger output at the beginning of the hop sequence:

Byte	Data	Description
0	205	Interrupt code for Get Hop Parameter
1	2	Set Trigger-Out at start of hop sequence

See Also

Frequency/Power Hop - Get Hop Trigger-In Mode
Frequency/Power Hop - Set Hop Trigger-In Mode
Frequency/Power Hop - Set Hop Trigger-Out Mode

3.4 (j) - Frequency/Power Hop - Set Hop Direction

Description

Sets the direction that the generator will run through the list of hop values.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	5	Code for Set Hop Direction
2	Hop_Direction	The direction to execute the current hop sequence: 0 = From first to last in the list 1 = From last to first in the list 2 = From first to last in the list, then back from last to first
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

The below transmit array will set the generator to hop through the list from the first value to the last:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	0	Execute the hop list from first to last

See Also

Frequency/Power Hop - Get Hop Direction

3.4 (k) - Frequency/Power Hop - Set Hop Dwell Time

Description

Sets the dwell time to be used by the generator between each frequency/power hop point in the sequence.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	4	Code for Set Hop Dwell Time
2	Hop_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (Hop_Dwell / 256)
3	Hop_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE3 = Hop_Dwell - (BYTE2 * 256)
4 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

The below transmit array sets the signal generator's dwell time to 300 ms:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	4	Code for Set Hop Dwell Time
2	1	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (300 / 256) = 1
3	44	The dwell time in milliseconds split into 2 bytes: BYTE3 = 300 - (1 * 256) = 44

See Also

Frequency/Power Hop - Get Hop Dwell Time
Frequency/Power Hop - Get Maximum Hop Dwell Time
Frequency/Power Hop - Get Minimum Hop Dwell Time

3.4 (l) - Frequency/Power Hop - Start/Stop Hop Sequence

Description

Starts or stops the hop sequence using the previously defined parameters. The hop sequence will stop automatically if any other command is sent.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	8	Code for Start/Stop Hop Sequence
2	Hop_Mode	Set the hop mode: 0 = Hop sequence is disabled 1 = Start hop sequence (the sequence will continue until Hop_Mode is set to 0 or any other command is sent)
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

The below transmit array enables the hop sequence (all hop parameters must be set first):

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	8	Code for Start/Stop Hop Sequence
2	1	Start the hop sequence

3.4 (m) - Frequency/Power Hop - Set Number of Hop Points

Description

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

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	0	Code for Set Number of Hop Points
2	Hop_Points	The number of points to configure in the hop sequence
3 - 63	Not used	"Don’t care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

The below transmit array configures a hop sequence with 3 points:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	0	Code for Set Number of Hop Points
2	3	Enable 3 points in the hop sequence

See Also

Frequency/Power Hop - Get Maximum Number of Hop Points
Frequency/Power Hop - Get Number of Hop Points

3.4 (n) - Frequency/Power Hop - Set Specific Hop Parameters

Description

Sets the frequency and power for a specified point within the hop sequence.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	1	Code for Set Specific Hop Parameters
2	Hop_Point	Index number of the hop point, from 0 to (n -1) where n equals the maximum allowed number of points
3	Freq	Frequency (Hz) of the hop point split over 5 bytes: BYTE3 = INT (Freq / 256^4)
4	Freq	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER1 = Freq - BYTE3 * 256^4 BYTE4 = INT (REMAINDER1 / 256^3)
5	Freq	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE4 * 256^3 BYTE5 = INT (REMAINDER2 / 256^2)
6	Freq	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER3 = REMAINDER2 - BYTE5 * 256^2 BYTE6 = INT (REMAINDER3 / 256)
7	Freq	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER4 = REMAINDER3 - BYTE6 * 256 If REMAINDER4 is greater than 0 then: BYTE7 = INT (REMAINDER4) If REMAINDER4 is less than or equal to 0 then: BYTE7 = 0
8	Power (+/-)	Power polarity of the hop point: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
9	Power_Mag	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE9 = INT (Power_Mag *100 / 256)
10	Power_Mag	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE10 = Power_Mag * 100 - (BYTE9 * 256)
11 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

To set point 3 in the hop sequence to 3500.25 MHz (3,500,250,000 Hz), -15.5 dBm, send the following transmit array:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	1	Code for Set Specific Hop Parameters
2	3	Set point 3 in the hop sequence
3	0	Frequency (Hz) of the hop point split over 5 bytes: BYTE3 = INT (3500250000 / 256^4) = 0
4	208	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER1 = 3500250000 - 0 * 256^4 = 3500250000 BYTE4 = INT (3500250000 / 256^3) = 208
5	161	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER2 = 3500250000 - 208 * 256^3 = 10589072 BYTE5 = INT (10589072 / 256^2) = 161
6	147	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER3 = 10589072 - 161 * 256^2 = 37776 BYTE6 = INT (37776 / 256) = 147
7	144	Frequency (Hz) of the hop point split over 5 bytes: REMAINDER4 = 37776 - 147 * 256 = 144 REMAINDER4 is greater than 0, therefore: BYTE7 = INT (144) = 144
8	1	Power value is negative
9	6	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE9 = INT (15.5 *100 / 256) = 6
10	14	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE10 = 15.5 * 100 - (6 * 256) = 14
11 - 63	Not used	"Don't care" bytes, can be any value

See Also

Frequency/Power Hop - Get Specific Hop Setting

3.4 (o) - Frequency/Power Hop - Set Hop Trigger-In Mode

Description

Sets the trigger-in mode, specifying how the generator will respond to an external trigger during the hop sequence.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	6	Code for Set Hop Trigger-In Mode
2	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each hop point 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the hop sequence
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter

Example

The below transmit array will set the generator to wait for an external trigger input before setting each point in the hop sequence:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	6	Code for Set Hop Trigger-In Mode
2	1	Wait for Trigger-In at each hop point

See Also

Frequency/Power Hop - Get Hop Trigger-In Mode
Frequency/Power Hop - Get Hop Trigger-Out Mode
Frequency/Power Hop - Set Hop Trigger-Out Mode

3.4 (p) - Frequency/Power Hop - Set Hop Trigger-Out Mode

Description

Sets the trigger-out mode, specifying when the generator will provide an external trigger signal during the hop sequence.

Transmit Array

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	7	Code for Set Hop Trigger-Out Mode
2	Trigger_Mode	The trigger-out mode: 0 = Disable trigger output 1 = Set trigger output (logic 1) on setting each hop point 2 = Set trigger output (logic 1) only at the start of the hop sequence
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	205	Interrupt code for Set Hop Parameter

Example

The below transmit array will set the generator to produce an external trigger output at the beginning of the hop sequence:

Byte	Data	Description
0	204	Interrupt code for Set Hop Parameter
1	7	Code for Set Hop Trigger-Out Mode
2	2	Set Trigger-Out at start of hop sequence

See Also

Frequency/Power Hop - Get Hop Trigger-In Mode
Frequency/Power Hop - Get Hop Trigger-Out Mode
Frequency/Power Hop - Set Hop Trigger-In Mode

3.5 - Frequency Sweep Functions

These functions define the frequency sweep capabilities of the generators. The signal generator can be configured to produce an automatic, swept frequency output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

Full details of the commands for configuring a frequency sweep sequence are covered in the following sections.

3.5 (a) - Frequency Sweep - Get Sweep Direction

Description

Returns the direction in which the generator will execute the frequency sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	7	Code for Get Sweep Direction
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Sweep_Direction	The direction set for the current frequency sweep: 0 = From start value to stop value 1 = From stop value to start value 2 = From start to stop, then back from stop to start

Example

The below returned array indicates that the signal generator will sweep bi-directionally, from start to finish, then back to start:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	2	The generator will sweep from start to stop, then back to start

See Also

Frequency Sweep - Set Sweep Direction

3.5 (b) - Frequency Sweep - Get Sweep Dwell Time

Description

Returns the dwell time to be used by the generator between each frequency in the sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	4	Code for Get Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Sweep_Dwell	The dwell time in milliseconds split over 2 bytes: Sweep_Dwell = (256 * BYTE1) + BYTE2
2	Sweep_Dwell	The dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's dwell time is set to 300ms:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	1	The dwell time in milliseconds split over 2 bytes: Sweep_Dwell = (256 * 1) + 44 = 300 ms
2	44	The dwell time in milliseconds split over 2 bytes

See Also

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

3.5 (c) - Frequency Sweep - Get Maximum Sweep Dwell Time

Description

Returns the maximum allowed dwell time in milliseconds for each frequency in the sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	5	Code for Get Maximum Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Sweep_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes: Sweep_MaxDwell = (256 * BYTE1) + BYTE2
2	Sweep_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's maximum dwell time is 10,000ms (10s):

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	39	The maximum dwell time in milliseconds split over 2 bytes: Sweep_MaxDwell = (256 * 39) + 16 = 10,000 ms
2	16	The maximum dwell time in milliseconds split over 2 bytes

See Also

Frequency Sweep - Get Sweep Dwell Time
Frequency Sweep - Get Minimum Sweep Dwell Time
Frequency Sweep - Set Sweep Dwell Time

3.5 (d) - Frequency Sweep - Get Minimum Sweep Dwell Time

Description

Returns the minimum allowed dwell time in milliseconds for all points in the frequency sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	6	Code for Get Minimum Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Sweep_MinDwell	The minimum dwell time in milliseconds split over 2 bytes: Sweep_MinDwell = (256 * BYTE1) + BYTE2
2	Sweep_MinDwell	The minimum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's minimum dwell time is 20ms:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	0	The minimum dwell time in milliseconds split over 2 bytes: Sweep_MinDwell = (256 * 0) + 20 = 20 ms
2	20	The minimum dwell time in milliseconds split over 2 bytes

See Also

Frequency Sweep - Get Sweep Dwell Time
Frequency Sweep - Get Maximum Sweep Dwell Time
Frequency Sweep - Set Sweep Dwell Time

3.5 (e) - Frequency Sweep - Get Sweep Power

Description

Returns the constant power level for the frequency sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	3	Code for Get Sweep Power
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Power (+/-)	Power polarity of the sweep: 0 = Positive (Power = 1 * Power_Mag) 1 = Negative (Power = -1 * Power_Mag)
2	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 * BYTE2 + BYTE3) / 100
3	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes

Example

The below example returned array indicates that the constant output level set for the current frequency sweep is -12.25 dBm:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	1	Power value is negative
2	4	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 * 4 + 201) / 100 = 12.25 dBm Power = (-1) * Power_Mag = -12.25 dBm
3	201	Power magnitude (dBm) of the sweep, split over 2 bytes

See Also

Frequency Sweep - Set Sweep Power

3.5 (f) - Frequency Sweep - Get Sweep Start Frequency

Description

Returns the start frequency for the current sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	0	Code for Get Sweep Start Frequency
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Freq	Start frequency (Hz) of the sweep split over 5 bytes: Freq = (256^4) * BYTE1 + (256^3) * BYTE2 + (256^2) * BYTE3 + (256) * BYTE4 + BYTE5
2	Freq	Start frequency (Hz) of the sweep split over 5 bytes
3	Freq	Start frequency (Hz) of the sweep split over 5 bytes
4	Freq	Start frequency (Hz) of the sweep split over 5 bytes
5	Freq	Start frequency (Hz) of the sweep split over 5 bytes

Example

The below returned array indicates that the start frequency for the sweep is 1000 MHz:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	0	Start frequency (Hz) of the sweep split over 5 bytes: Freq = (256^4) * 0 + (256^3) * 59 + (256^2) * 154 + (256) * 202 + 0 = 1,000,000,000 Hz = 1,000 MHz
2	59	Start frequency (Hz) of the sweep split over 5 bytes
3	154	Start frequency (Hz) of the sweep split over 5 bytes
4	202	Start frequency (Hz) of the sweep split over 5 bytes
5	0	Start frequency (Hz) of the sweep split over 5 bytes

See Also

Frequency Sweep - Set Sweep Start Frequency

3.5 (g) - Frequency Sweep - Get Sweep Stop Frequency

Description

Returns the stop frequency for the current sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	1	Code for Get Sweep Stop Frequency
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Freq	Stop frequency (Hz) of the sweep split over 5 bytes: Freq $= (256^4) * \text{BYTE1}$ $+ (256^3) * \text{BYTE2}$ $+ (256^2) * \text{BYTE3}$ $+ (256) * \text{BYTE4}$ $+ \text{BYTE5}$
2	Freq	Stop frequency (Hz) of the sweep split over 5 bytes
3	Freq	Stop frequency (Hz) of the sweep split over 5 bytes
4	Freq	Stop frequency (Hz) of the sweep split over 5 bytes
5	Freq	Stop frequency (Hz) of the sweep split over 5 bytes

Example

The below example returned array indicates that the stop frequency for the sweep is 1,999.999999 MHz:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	0	Stop frequency (Hz) of the sweep split over 5 bytes: Freq $= (256^4) * 0$ $+ (256^3) * 119$ $+ (256^2) * 53$ $+ (256) * 147$ $+ 255$ = 1,999,999,999 Hz = 1,999.999999 MHz
2	119	Stop frequency (Hz) of the sweep split over 5 bytes
3	53	Stop frequency (Hz) of the sweep split over 5 bytes
4	147	Stop frequency (Hz) of the sweep split over 5 bytes
5	255	Stop frequency (Hz) of the sweep split over 5 bytes

See Also

Frequency Sweep - Set Sweep Stop Frequency

3.5 (h) - Frequency Sweep - Get Sweep Step Size

Description

Returns the frequency step size for the current sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	2	Code for Get Sweep Step Size
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Freq	Step frequency (Hz) of the sweep split over 5 bytes: Freq = (256^4) * BYTE1 + (256^3) * BYTE2 + (256^2) * BYTE3 + (256) * BYTE4 + BYTE5
2	Freq	Step frequency (Hz) of the sweep split over 5 bytes
3	Freq	Step frequency (Hz) of the sweep split over 5 bytes
4	Freq	Step frequency (Hz) of the sweep split over 5 bytes
5	Freq	Step frequency (Hz) of the sweep split over 5 bytes

Example

The below example returned array indicates that the step size for the sweep is 100 MHz:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	0	Step frequency (Hz) of the sweep split over 5 bytes: Freq = (256^4) * 0 + (256^3) * 5 + (256^2) * 245 + (256) * 225 + 0 = 100,000,000 Hz = 100 MHz
2	5	Step frequency (Hz) of the sweep split over 5 bytes
3	245	Step frequency (Hz) of the sweep split over 5 bytes
4	225	Step frequency (Hz) of the sweep split over 5 bytes
5	0	Step frequency (Hz) of the sweep split over 5 bytes

See Also

Frequency Sweep - Set Sweep Step Size

3.5 (i) - Frequency Sweep - Get Sweep Trigger-In Mode

Description

Returns the trigger-in mode which specifies how the generator will respond to an external trigger during the frequency sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	8	Code for Get Sweep Trigger-In Mode
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each frequency 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the sweep

Example

The below returned array indicates that the generator is configured to wait for an external trigger input before setting each frequency in the sweep:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	1	Wait for Trigger-In before each frequency point

See Also

Frequency Sweep - Get Sweep Trigger-Out Mode
Frequency Sweep - Set Sweep Trigger-In Mode
Frequency Sweep - Set Sweep Trigger-Out Mode

3.5 (j) - Frequency Sweep - Get Sweep Trigger-Out Mode

Description

Returns the trigger-out mode which specifies when the generator will provide an external trigger signal during the frequency sweep.

Transmit Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	9	Code for Get Sweep Trigger-Out Mode
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	Trigger_Mode	The trigger-out mode: 0 = Disable trigger output 1 = Set trigger output (logic 1) on setting each frequency 2 = Set trigger output (logic 1) only at the start of the sweep

Example

The below returned array indicates that the generator is configured to produce an external trigger output at the beginning of the frequency sweep:

Byte	Data	Description
0	201	Interrupt code for Get Frequency Sweep Parameter
1	2	Set Trigger-Out at start of sweep

See Also

Frequency Sweep - Get Sweep Trigger-In Mode
Frequency Sweep - Set Sweep Trigger-In Mode
Frequency Sweep - Set Sweep Trigger-Out Mode

3.5 (k) - Frequency Sweep - Set Sweep Direction

Description

Sets the direction in which the generator will execute the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	5	Code for Set Sweep Direction
2	Sweep_Direction	The direction to execute the sweep: 0 = From start to stop frequency 1 = From stop to start frequency 2 = From start to stop, then back to start
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The below transmit array will set the generator to sweep backwards, from stop frequency to start frequency:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	1	Execute the sweep backwards

See Also

Frequency Sweep - Get Sweep Direction

3.5 (l) - Frequency Sweep - Set Sweep Dwell Time

Description

Sets the dwell time to be used by the generator between setting each frequency in the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	4	Code for Set Sweep Dwell Time
2	Sweep_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (Sweep_Dwell / 256)
3	Sweep_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE3 = Sweep_Dwell - (BYTE2 * 256)
4 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The below transmit array sets the signal generator's dwell time to 300 ms:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	4	Code for Set Sweep Dwell Time
2	1	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (300 / 256) = 1
3	44	The dwell time in milliseconds split into 2 bytes: BYTE3 = 300 - (1 * 256) = 44

See Also

Frequency Sweep - Get Sweep Dwell Time
Frequency Sweep - Get Maximum Sweep Dwell Time
Frequency Sweep - Get Minimum Sweep Dwell Time

3.5 (m) - Frequency Sweep - Start/Stop Sweep Sequence

Description

Starts or stops the frequency sweep using the previously defined parameters. The sweep will stop automatically if any other command is sent.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	8	Code for Start/Stop Sweep Sequence
2	Sweep_Mode	Set the hop mode: 0 = Sweep sequence is disabled 1 = Start sweep (the sequence will continue until Sweep_Mode is set to 0 or any other command is sent)
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The below transmit array enables the frequency sweep (all sweep parameters must be set first):

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	8	Code for Start/Stop Sweep Sequence
2	1	Start the sweep

3.5 (n) - Frequency Sweep - Set Sweep Power

Description

Sets the constant power level for the frequency sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	3	Code for Set Sweep Power
2	Power (+/-)	Power polarity of the sweep: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
3	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes: BYTE3 = INT (Power_Mag *100 / 256)
4	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes: BYTE4 = Power_Mag * 100 - (BYTE3 * 256)
5 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The following transmit array sets a constant output power level of 5.75dBm to be used for the frequency sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	3	Code for Set Sweep Power
2	0	Power value is positive
3	2	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE3 = INT (5.75 *100 / 256) = 2
4	63	Power magnitude (dBm) of the hop point split over 2 bytes: BYTE4 = 5.75 * 100 - (2 * 256) = 63
5 - 63	Not used	"Don't care" bytes, can be any value

See Also

Frequency Sweep - Get Sweep Power

3.5 (o) - Frequency Sweep - Set Sweep Start Frequency

Description

Sets the start frequency for the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	0	Code for Set Sweep Start Frequency
2	Freq	Start frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (Freq / 256^4)
3	Freq	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = Freq - BYTE2 * 256^4 BYTE3 = INT (REMAINDER1 / 256^3)
4	Freq	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE3 * 256^3 BYTE4 = INT (REMAINDER2 / 256^2)
5	Freq	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = REMAINDER2 - BYTE4 * 256^2 BYTE5 = INT (REMAINDER3 / 256)
6	Freq	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = REMAINDER3 - BYTE5 * 256 If REMAINDER4 is greater than 0 then: BYTE6 = INT (REMAINDER4) If REMAINDER4 is less than or equal to 0 then: BYTE6 = 0
7 - 63	Not used	"Don’t care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The following transmit array sets 1000 MHz as the start frequency for the sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	0	Code for Set Sweep Start Frequency
2	0	Start frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (1000000000 / 256^4) = 0
3	59	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = 1000000000 - 0 * 256^4 = 1000000000 BYTE3 = INT (1000000000 / 256^3) = 59
4	154	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = 1000000000 - 59 * 256^3 = 10144256 BYTE4 = INT (10144256 / 256^2) = 154
5	202	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = 10144256 - 154 * 256^2 = 51712 BYTE5 = INT (51712 / 256) = 202
6	0	Start frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = 51712 - 202 * 256 = 0 BYTE6 = INT (0) = 0
7 - 63	Not used	"Don’t care" bytes, can be any value

See Also

Frequency Sweep - Get Sweep Start Frequency

3.5 (p) - Frequency Sweep - Set Sweep Stop Frequency

Description

Sets the stop frequency for the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	1	Code for Set Sweep Stop Frequency
2	Freq	Stop frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (Freq / 256^4)
3	Freq	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = Freq - BYTE2 * 256^4 BYTE3 = INT (REMAINDER1 / 256^3)
4	Freq	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE3 * 256^3 BYTE4 = INT (REMAINDER2 / 256^2)
5	Freq	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = REMAINDER2 - BYTE4 * 256^2 BYTE5 = INT (REMAINDER3 / 256)
6	Freq	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = REMAINDER3 - BYTE5 * 256 If REMAINDER4 is greater than 0 then: BYTE6 = INT (REMAINDER4) If REMAINDER4 is less than or equal to 0 then: BYTE6 = 0
7 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The following transmit array sets 1,999,999,999 Hz (1999.999999 MHz) as the stop frequency for the sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	1	Code for Set Sweep Stop Frequency
2	0	Stop frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (1999999999 / 256^4) = 0
3	119	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = 1999999999 - 0 * 256^4 = 1999999999 BYTE3 = INT (1999999999 / 256^3) = 119
4	53	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = 1999999999 - 119 * 256^3 = 3511295 BYTE4 = INT (3511295 / 256^2) = 53
5	147	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = 3511295 - 53 * 256^2 = 37887 BYTE5 = INT (37887 / 256) = 147
6	255	Stop frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = 37887 - 147 * 256 = 255 BYTE6 = INT (255) = 255
7 - 63	Not used	"Don't care" bytes, can be any value

See Also

Frequency Sweep - Set Sweep Stop Frequency

3.5 (q) - Frequency Sweep - Set Sweep Step Size

Description

Sets the frequency step size for the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	2	Code for Set Sweep Step Size
2	Freq	Step frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (Freq / 256^4)
3	Freq	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = Freq - BYTE2 * 256^4 BYTE3 = INT (REMAINDER1 / 256^3)
4	Freq	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE3 * 256^3 BYTE4 = INT (REMAINDER2 / 256^2)
5	Freq	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = REMAINDER2 - BYTE4 * 256^2 BYTE5 = INT (REMAINDER3 / 256)
6	Freq	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = REMAINDER3 - BYTE5 * 256 If REMAINDER4 is greater than 0 then: BYTE6 = INT (REMAINDER4) If REMAINDER4 is less than or equal to 0 then: BYTE6 = 0
7 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The following transmit array sets a step size of 100 MHz for the frequency sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	2	Code for Set Sweep Step Size
2	0	Step frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (100000000 / 256^4) = 0
3	5	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = 100000000 - 0 * 256^4 = 100000000 BYTE3 = INT (100000000 / 256^3) = 5
4	245	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = 100000000- 5 * 256^3 = 16113920 BYTE4 = INT (16113920 / 256^2) = 245
5	225	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = 16113920 - 245 * 256^2 = 57600 BYTE5 = INT (57600 / 256) = 225
6	0	Step frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = 57600 - 225 * 256 = 0 BYTE6 = 0
7 - 63	Not used	"Don't care" bytes, can be any value

See Also

Frequency Sweep - Get Sweep Step Size

3.5 (r) - Frequency Sweep - Set Sweep Trigger-In Mode

Description

Sets the trigger-in mode, specifying how the generator will respond to an external trigger during the frequency sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	6	Code for Set Sweep Trigger-In Mode
2	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each frequency 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the sweep
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The below transmit array will set the generator to wait for an external trigger input before setting each frequency in the sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	6	Code for Set Sweep Trigger-In Mode
2	1	Wait for Trigger-In before each frequency is set

See Also

Frequency Sweep - Get Sweep Trigger-In Mode
Frequency Sweep - Get Sweep Trigger-Out Mode
Frequency Sweep - Set Sweep Trigger-Out Mode

3.5 (s) - Frequency Sweep - Set Sweep Trigger-Out Mode

Description

Sets the trigger-out mode, specifying when the generator will provide an external trigger signal during the sweep.

Transmit Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	7	Code for Set Sweep Trigger-Out Mode
2	Trigger_Mode	The trigger-out mode: 0 = Disable trigger output 1 = Set trigger output (logic 1) on setting each frequency 2 = Set trigger output (logic 1) only at the start of the sweep
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter

Example

The below transmit array will set the generator to produce an external trigger output at the beginning of the frequency sweep:

Byte	Data	Description
0	200	Interrupt code for Set Frequency Sweep Parameter
1	7	Code for Set Sweep Trigger-Out Mode
2	2	Set Trigger-Out at start of sweep

See Also

Frequency Sweep - Get Sweep Trigger-In Mode
Frequency Sweep - Get Sweep Trigger-Out Mode
Frequency Sweep - Set Sweep Trigger-In Mode

3.6 - Power Sweep Functions

These functions define the power sweep capabilities of the generators. The signal generator can be configured to produce an automatic, swept power output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

Full details of the commands for configuring a power sweep sequence are covered in the following sections.

3.6 (a) - Power Sweep - Get Sweep Direction

Description

Returns the direction in which the generator will execute the power sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	7	Code for Get Sweep Direction
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Sweep_Direction	The direction set for the current power sweep: 0 = From start value to stop value 1 = From stop value to start value 2 = From start to stop, then back from stop to start

Example

The below returned array indicates that the signal generator will sweep bi-directionally, from start to finish, then back to start:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	2	The generator will sweep from start to stop, then back to start

See Also

Power Sweep - Set Sweep Direction

3.6 (b) - Power Sweep - Get Sweep Dwell Time

Description

Returns the dwell time to be used by the generator between each power level in the sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	4	Code for Get Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Sweep_Dwell	The dwell time in milliseconds split over 2 bytes: Sweep_Dwell = (256 * BYTE1) + BYTE2
2	Sweep_Dwell	The dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's dwell time is set to 300ms:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	1	The dwell time in milliseconds split over 2 bytes: Sweep_Dwell = (256 * 1) + 44 = 300 ms
2	44	The dwell time in milliseconds split over 2 bytes

See Also

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

3.6 (c) - Power Sweep - Get Maximum Sweep Dwell Time

Description

Returns the maximum allowed dwell time in milliseconds for each power level in the sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	5	Code for Get Maximum Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Sweep_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes: Sweep_MaxDwell = (256 * BYTE1) + BYTE2
2	Sweep_MaxDwell	The maximum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's maximum dwell time is 10,000ms (10s):

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	39	The maximum dwell time in milliseconds split over 2 bytes: Sweep_MaxDwell = (256 * 39) + 16 = 10,000 ms
2	16	The maximum dwell time in milliseconds split over 2 bytes

See Also

Power Sweep - Get Sweep Dwell Time
Power Sweep - Get Minimum Sweep Dwell Time
Power Sweep - Set Sweep Dwell Time

3.6 (d) - Power Sweep - Get Minimum Sweep Dwell Time

Description

Returns the minimum allowed dwell time in milliseconds for all points in the power sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	6	Code for Get Minimum Sweep Dwell Time
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Sweep_MinDwell	The minimum dwell time in milliseconds split over 2 bytes: Sweep_MinDwell = (256 * BYTE1) + BYTE2
2	Sweep_MinDwell	The minimum dwell time in milliseconds split over 2 bytes

Example

The below returned array indicates that the signal generator's minimum dwell time is 20ms:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	0	The minimum dwell time in milliseconds split over 2 bytes: Sweep_MinDwell = (256 * 0) + 20 = 20 ms
2	20	The minimum dwell time in milliseconds split over 2 bytes

See Also

Power Sweep - Get Sweep Dwell Time
Power Sweep - Get Maximum Sweep Dwell Time
Power Sweep - Set Sweep Dwell Time

3.6 (e) - Power Sweep - Get Sweep Frequency

Description

Returns the constant frequency used for the power sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	3	Code for Get Sweep Frequency
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: Freq = (256^4) * BYTE1 + (256^3) * BYTE2 + (256^2) * BYTE3 + (256) * BYTE4 + BYTE5
2	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes
3	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes
4	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes
5	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes

Example

The below example returned array indicates that the constant frequency to be used for the current power sweep is 1,000 MHz:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	0	Constant frequency (Hz) of the sweep, split over 5 bytes: Freq = (256^4) * 0 + (256^3) * 59 + (256^2) * 154 + (256) * 202 + 0 = 1,000,000,000 Hz = 1,000 MHz
2	59	Constant frequency (Hz) of the sweep, split over 5 bytes
3	154	Constant frequency (Hz) of the sweep, split over 5 bytes
4	202	Constant frequency (Hz) of the sweep, split over 5 bytes
5	0	Constant frequency (Hz) of the sweep, split over 5 bytes

See Also

Power Sweep - Set Sweep Frequency

3.6 (f) - Power Sweep - Get Sweep Start Power

Description

Returns the start power for the current sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	0	Code for Get Sweep Start Power
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Power (+/-)	Power polarity of the sweep: 0 = Positive (Power = 1 * Power_Mag) 1 = Negative (Power = -1 * Power_Mag)
2	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 * BYTE2 + BYTE3) / 100
3	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes

Example

The below returned array indicates that the start power for the sweep is -12.25 dBm:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	1	Power value is negative (Power = (-1) * Power_Mag)
2	4	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 * 4 + 201) / 100 = 12.25 dBm Power = (-1) * Power_Mag = -12.25 dBm
3	201	Power magnitude (dBm) of the sweep, split over 2 bytes

See Also

Power Sweep - Set Sweep Start Power
Power Sweep - Set Sweep Stop Power
Power Sweep - Set Sweep Power Step Size

3.6 (g) - Power Sweep - Get Sweep Stop Power

Description

Returns the stop power for the current sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	1	Code for Get Sweep Stop Power
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Power (+/-)	Power polarity of the sweep: 0 = Positive (Power = 1 * Power_Mag) 1 = Negative (Power = -1 * Power_Mag)
2	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 * BYTE2 + BYTE3) / 100
3	Power_Mag	Power magnitude (dBm) of the sweep, split over 2 bytes

Example

The below returned array indicates that the stop power for the sweep is +10.00 dBm:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	0	Power value is negative (Power = 1 * Power_Mag)
2	3	Power magnitude (dBm) of the sweep, split over 2 bytes: Power_Mag = (256 *3 + 232) / 100 = 10.00 dBm Power = (+1) * Power_Mag = 10.00 dBm
3	232	Power magnitude (dBm) of the sweep, split over 2 bytes

See Also

Power Sweep - Set Sweep Start Power
Power Sweep - Set Sweep Stop Power
Power Sweep - Set Sweep Power Step Size

3.6 (h) - Power Sweep - Get Sweep Power Step Size

Description

Returns the power step size for the current sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	2	Code for Get Sweep Power Step Size
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Power_Step	Power step size (dBm), split over 2 bytes: Power_Step = (256 * BYTE1 + BYTE2) / 100
2	Power_Step	Power step size (dBm), split over 2 bytes

Example

The below returned array indicates that the power step size for the sweep is 0.25 dBm:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	0	Power step size (dBm), split over 2 bytes: Power_Step = (256 * 0 + 25) / 100 = 0.25 dBm
2	25	Power step size (dBm), split over 2 bytes

See Also

Power Sweep - Set Sweep Start Power
Power Sweep - Set Sweep Stop Power
Power Sweep - Set Sweep Power Step Size

3.6 (i) - Power Sweep - Get Sweep Trigger-In Mode

Description

Returns the trigger-in mode which specifies how the generator will respond to an external trigger during the power sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	8	Code for Get Sweep Trigger-In Mode
2 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each power 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the sweep

Example

The below returned array indicates that the generator is configured to wait for an external trigger input before setting each power in the sweep:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	1	Wait for Trigger-In before each power point

See Also

Power Sweep - Get Sweep Trigger-Out Mode
Power Sweep - Set Sweep Trigger-In Mode
Power Sweep - Set Sweep Trigger-Out Mode

3.6 (j) - Power Sweep - Get Sweep Trigger-Out Mode

Description

Returns the trigger-out mode which specifies when the generator will provide an external trigger signal during the power sweep.

Transmit Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	9	Code for Get Sweep Trigger-Out Mode
2 - 63	Not used	"Don’t care" bytes, can be any value

Returned Array

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	Trigger_Mode	The trigger-out mode: 2 = Disable trigger output 3 = Set trigger output (logic 1) on setting each power 4 = Set trigger output (logic 1) only at the start of the sweep

Example

The below returned array indicates that the generator is configured to produce an external trigger output at the beginning of the power sweep:

Byte	Data	Description
0	203	Interrupt code for Get Power Sweep Parameter
1	2	Set Trigger-Out at start of sweep

See Also

Power Sweep - Get Sweep Trigger-In Mode
Power Sweep - Set Sweep Trigger-In Mode
Power Sweep - Set Sweep Trigger-Out Mode

3.6 (k) - Power Sweep - Set Sweep Direction

Description

Sets the direction in which the generator will execute the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	5	Code for Set Sweep Direction
2	Sweep_Direction	The direction to execute the sweep: 0 = From start to stop power 1 = From stop to start power 2 = From start to stop, then back to start
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The below transmit array will set the generator to sweep backwards, from stop power to start power:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	1	Execute the sweep backwards

See Also

Power Sweep - Get Sweep Direction

3.6 (l) - Power Sweep - Set Sweep Dwell Time

Description

Sets the dwell time to be used by the generator between setting each power in the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	4	Code for Set Sweep Dwell Time
2	Sweep_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (Sweep_Dwell / 256)
3	Sweep_Dwell	The dwell time in milliseconds split into 2 bytes: BYTE3 = Sweep_Dwell - (BYTE2 * 256)
4 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The below transmit array sets the signal generator's dwell time to 300 ms:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	4	Code for Set Sweep Dwell Time
2	1	The dwell time in milliseconds split into 2 bytes: BYTE2 = INT (300 / 256) = 1
3	44	The dwell time in milliseconds split into 2 bytes: BYTE3 = 300 - (1 * 256) = 44

See Also

Power Sweep - Get Sweep Dwell Time
Power Sweep - Get Maximum Sweep Dwell Time
Power Sweep - Get Minimum Sweep Dwell Time

3.6 (m) - Power Sweep - Start/Stop Sweep Sequence

Description

Starts or stops the power sweep using the previously defined parameters. The sweep will stop automatically if any other command is sent.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	8	Code for Start/Stop Sweep Sequence
2	Sweep_Mode	Set the hop mode: 0 = Sweep sequence is disabled 1 = Start sweep (the sequence will continue until Sweep_Mode is set to 0 or any other command is sent)
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The below transmit array enables the power sweep (all sweep parameters must be set first):

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	8	Code for Start/Stop Sweep Sequence
2	1	Start the sweep

3.6 (n) - Power Sweep - Set Sweep Frequency

Description

Sets the constant frequency to be used during the power sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	3	Code for Set Sweep Frequency
2	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (Freq / 256^4)
3	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = Freq - BYTE2 * 256^4 BYTE3 = INT (REMAINDER1 / 256^3)
4	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = REMAINDER1 - BYTE3 * 256^3 BYTE4 = INT (REMAINDER2 / 256^2)
5	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = REMAINDER2 - BYTE4 * 256^2 BYTE5 = INT (REMAINDER3 / 256)
6	Freq	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = REMAINDER3 - BYTE5 * 256 If REMAINDER4 is greater than 0 then: BYTE6 = INT (REMAINDER4) If REMAINDER4 is less than or equal to 0 then: BYTE6 = 0
7 - 63	Not used	"Don’t care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The following transmit array sets a constant frequency of 1000 MHz to be used for the power sweep:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	3	Code for Set Sweep Frequency
2	0	Constant frequency (Hz) of the sweep, split over 5 bytes: BYTE2 = INT (1000000000 / 256^4) = 0
3	59	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER1 = 1000000000 - 0 * 256^4 = 1000000000 BYTE3 = INT (1000000000 / 256^3) = 59
4	154	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER2 = 1000000000 - 59 * 256^3 = 10144256 BYTE4 = INT (10144256 / 256^2) = 154
5	202	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER3 = 10144256 - 154 * 256^2 = 51712 BYTE5 = INT (51712 / 256) = 202
6	0	Constant frequency (Hz) of the sweep, split over 5 bytes: REMAINDER4 = 51712 - 202 * 256 = 0 BYTE6 = INT (0) = 0
7 - 63	Not used	"Don't care" bytes, can be any value

See Also

Power Sweep - Set Sweep Frequency

3.6 (o) - Power Sweep - Set Sweep Start Power

Description

Sets the start power for the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	0	Code for Set Sweep Start Power
2	Power (+/-)	Power polarity of the start power: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
3	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE3 = INT (Power_Mag * 100 / 256)
4	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE4 = Power_Mag * 100 - (BYTE3 * 256)
5 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The following transmit array sets -12.25dBm as the start power for the sweep:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	0	Code for Set Sweep Start Power
2	1	Power value is negative (Power = (-1) * Power_Mag)
3	4	Power magnitude (dBm), split over 2 bytes: BYTE3 = INT (12.25 * 100 / 256) = 4
4	201	Power magnitude (dBm), split over 2 bytes: BYTE4 = 12.25 * 100 - (4 * 256) = 201
5 - 63	Not used	"Don't care" bytes, can be any value

See Also

Power Sweep - Get Sweep Start Power

3.6 (p) - Power Sweep - Set Sweep Stop Power

Description

Sets the stop power for the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	1	Code for Set Sweep Stop Power
2	Power (+/-)	Power polarity of the stop power: 0 = Positive (Power = 1 * Power) 1 = Negative (Power = -1 * Power)
3	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE3 = INT (Power_Mag * 100 / 256)
4	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE4 = Power_Mag * 100 - (BYTE3 * 256)
5 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The following transmit array sets +10.00dBm as the start power for the sweep:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	1	Code for Set Sweep Stop Power
2	0	Power value is positive (Power = 1 * Power_Mag)
3	3	Power magnitude (dBm), split over 2 bytes: BYTE3 = INT (10 * 100 / 256) = 3
4	232	Power magnitude (dBm), split over 2 bytes: BYTE4 = 10 * 100 - (3 * 256) = 232
5 - 63	Not used	"Don't care" bytes, can be any value

See Also

Power Sweep - Get Sweep Stop Power

3.6 (q) - Power Sweep - Set Sweep Power Step Size

Description

Sets the power step size for the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	2	Code for Set Sweep Power Step Size
2	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE2 = INT (Power_Mag * 100 / 256)
3	Power_Mag	Power magnitude (dBm), split over 2 bytes: BYTE3 = Power_Mag * 100 - (BYTE2 * 256)
4 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The following transmit array sets 0.25dBm as the sweep power step size:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	2	Code for Set Sweep Power Step Size
2	0	Power magnitude (dBm), split over 2 bytes: BYTE2 = INT (0.25 * 100 / 256) = 0
3	25	Power magnitude (dBm), split over 2 bytes: BYTE3 = 0.25 * 100 - (0 * 256) = 25
4 - 63	Not used	"Don't care" bytes, can be any value

See Also

Power Sweep - Get Sweep Power Step Size

3.6 (r) - Power Sweep - Set Sweep Trigger-In Mode

Description

Sets the trigger-in mode, specifying how the generator will respond to an external trigger during the power sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	6	Code for Set Sweep Trigger-In Mode
2	Trigger_Mode	The trigger-in mode: 0 = Ignore trigger input 1 = Wait for external trigger (Trigger In = logic 1) before setting each power 2 = Wait for external trigger (Trigger In = logic 1) only at the start of the sweep
3 - 63	Not used	"Don’t care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The below transmit array will set the generator to wait for an external trigger input before setting each power in the sweep:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	6	Code for Set Sweep Trigger-In Mode
2	1	Wait for Trigger-In before each power is set

See Also

Power Sweep - Get Sweep Trigger-In Mode
Power Sweep - Get Sweep Trigger-Out Mode
Power Sweep - Set Sweep Trigger-Out Mode

3.6 (s) - Power Sweep - Set Sweep Trigger-Out Mode

Description

Sets the trigger-out mode, specifying when the generator will provide an external trigger signal during the sweep.

Transmit Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	7	Code for Set Sweep Trigger-Out Mode
2	Trigger_Mode	The trigger-out mode: 0 = Disable trigger output 1 = Set trigger output (logic 1) on setting each power 2 = Set trigger output (logic 1) only at the start of the sweep
3 - 63	Not used	"Don't care" bytes, can be any value

Returned Array

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter

Example

The below transmit array will set the generator to produce an external trigger output at the beginning of the frequency sweep:

Byte	Data	Description
0	202	Interrupt code for Set Power Sweep Parameter
1	7	Code for Set Sweep Trigger-Out Mode
2	2	Set Trigger-Out at start of sweep

See Also

Power Sweep - Get Sweep Trigger-In Mode
Power Sweep - Get Sweep Trigger-Out Mode
Power Sweep - Set Sweep Trigger-In Mode

3.7 - Ethernet Configuration Functions

These commands and queries apply to Mini-Circuits Ethernet enabled series of signal generators for configuring the Ethernet parameters.

3.7 (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

3.7 (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

3.7 (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

3.7 (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

3.7 (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

3.7 (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

3.7 (g) - Set Password

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use Password
Get Password Status
Get Password
Reset Ethernet Configuration

3.7 (h) - Use DHCP

Description

Enables or disables DHCP (dynamic host control protocol). With DHCP enabled, the generators 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

3.7 (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

3.7 (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

3.7 (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

3.7 (l) - 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

3.7 (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

3.7 (n) - Get Password Status

Description

Checks whether the generators 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

3.7 (o) - Get Password

Description

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

Transmit Array

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

Returned Array

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

Example

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

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

See Also

Use Password
Set Password
Get Password Status

3.7 (p) - Get DHCP Status

Description

Checks whether DHCP (dynamic host control protocol) is enabled or disabled. With DHCP enabled, the generators 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

3.7 (q) - Get Dynamic Ethernet Configuration

Description

Returns the IP address, subnet mask and default gateway currently used by the signal generator. 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

3.7 (r) - Get MAC Address

Description

Returns the MAC address of the signal generator.

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

3.7 (s) - Reset Ethernet Configuration

Description

Forces the signal generator 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	111	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 signal generators 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 generators 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 switch unless the default port 80 is used
2. The password must be included in every HTTP command to the switch if password security is enabled
3. Port 23 is reserved for Telnet communication

4.1 - Configuring Ethernet Settings

The generator must be connected via the USB interface in order to configure the Ethernet settings. Following initial configuration, the device can be controlled via the Ethernet interface with no further need for a USB connection. The API DLL provides the below functions for configuring the Ethernet settings, please see Ethernet Configuration DLL Functions for full details:

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

4.2 - Ethernet Communication

Communication over Ethernet is 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. See SCPI Functions for the SCPI commands that can be sent to Mini-Circuits signal generators.

4.2 (a) - Setting Generator Properties Using HTTP and SCPI

The basic format of the HTTP command to communicate with the generator is as below (the text "COMMAND=" can be omitted with firmware A5 and later).

http://[*address*]:[*port*]/PWD=[*password*];COMMAND=[*command*]

Where

- [*address*] = IP address (required)
- [*port*] = TCP/IP port (can be omitted if port 80 is used)
- [*password*] = Password (can be omitted if password security is not enabled)
- [*command*] = SCPI command to send to the generator

Example 1:

http://192.168.100.100:800/PWD=123;COMMAND=:FREQ:2105MHZ
http://192.168.100.100:800/PWD=123;:FREQ:2105MHZ

Explanation:

- The generator 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 output frequency to 2105MHz (see below for the full explanation of all commands/queries)

Example 2:

http://10.10.10.10/COMMAND=:POWER:8.5
http://10.10.10.10/:POWER:8.5

Explanation:

- The generator has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled
- The command is to set the output power to 8.5dBm (see below for the full explanation of all commands/queries)

4.2 (b) - Querying Generator Properties Using HTTP and SCPI

The basic format of the HTTP query to communicate with the generator is as below (the text "QUERY=" can be omitted with firmware A5 and later).

http://[*address*]:[*port*]/PWD=[*password*];QUERY=[*query*]

Where

- [*address*] = IP address (required)
- [*port*] = TCP/IP port (can be omitted if port 80 is used)
- [*password*] = Password (can be omitted if password security is not enabled)
- [*query*] = SCPI query to send to the generator

Example 1:

http://192.168.100.100:800/PWD=123;QUERY=:FREQ?
http://192.168.100.100:800/PWD=123;:FREQ?

Explanation:

- The generator 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 current output frequency of the generator (see below for the full explanation of all commands/queries)

Example 2:

http://10.10.10.10/QUERY=:PWR?
http://10.10.10.10/:PWR?

Explanation:

- The generator has IP address 10.10.10.10 and uses the default port 80
- Password security is disabled
- The query is to return the current output power of the generator (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 and SCPI

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

The full list of all SCPI 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 a generator with IP address 192.168.9.60:

```
C:\Windows\system32\cmd.exe - telnet
Welcome to Microsoft Telnet Client

Escape Character is 'CTRL+]'

Microsoft Telnet> open 192.168.9.60
```

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

```
Telnet 192.168.9.60
_
```

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

```
Telnet 192.168.9.60
PWD=123;
1
```

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

```
Telnet 192.168.9.60
MN=SSG-6000RC
:SN?
SN=11402040043
:FREQ:2105MHZ
1
:FREQ?
2105.000000
:PWR:-10
1
:PWR?
-10.0
:PWR:RF:ON
1
:PWR:RF?
ON
```

4.3 - Device Discovery Using UDP

In addition to HTTP and Telnet, Mini-Circuits' Ethernet enabled signal generators 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 generators connected on the network; full control of those units is then accomplished using SCPI over HTTP or Telnet, as detailed previously.

Alternatively, the IP configuration can be identified or changed by connecting the generator 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 signal generators 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 generator's IP address is already known it is not necessary to use UDP.

Transmission

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

Receipt

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

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

Example

Sent Data:

MCL_SSG?

Received Data:

Model Name: SSG-15G-RC
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-D8-03

Model Name: SSG-15G-RC
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-D8-04

Model Name: SSG-15G-RC
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-D8-05

5 - SCPI Functions

This section details the control functions applicable to Mini-Circuits' Ethernet enabled signal generators using SCPI communication. SCPI (Standard Commands for Programmable Instruments) is a common method for controlling instrumentation products.

The SCPI commands are sent as an ASCII text string 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. The return value will be an ASCII text string. If an unrecognized command/query is received the generator will return:

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

These commands and queries can be sent using the DLL functions SCPI_Command and SCPI_Query when the generator is connected through the USB interface in a Microsoft Windows environment. In addition, these functions can be called using HTTP get/post commands or Telnet over a TCP/IP network when the device is connected via the Ethernet RJ45 port (see Ethernet Control over IP Networks).

5.1 - Core Commands & Queries

5.1 (a) - Get Model Name

Description

This function returns the model name of the signal generator.

Command Syntax

:MN?

Return String

MN=[model]

Variable	Description
[model]	Model name of the generator

Examples

String to Send	String Returned
:MN?	MN=SSG-15G-RC

DLL Implementation: `SCPI_Query(":MN?", RetSTR)`
Ethernet Implementation: `:MN?`

See Also

Get Serial Number
Get MAC Address

5.1 (b) - Get Serial Number

Description

This function returns the serial number of the signal generator.

Command Syntax

:SN?

Return String

SN=[serial]

Variable	Description
[serial]	Serial number of the generator (for example, "11401010001")

Examples

String to Send	String Returned
:SN?	SN=11401010001

DLL Implementation: `SCPI_Query(":SN?", RetSTR)`
Ethernet Implementation: `:SN?`

See Also

Get Model Name
Get MAC Address

5.1 (c) - Get MAC Address

Description

This function returns the MAC address of the signal generator.

Command Syntax

`:MAC?`

Return String

`MAC=[address]`

Variable	Description
[address]	MAC address of the generator as 6 hexadecimal pairs separated by hyphen characters (for example, "D0-73-7F-86-5C-2B")

Examples

String to Send	String Returned
:MAC?	MAC=D0-73-7F-86-5C-2B

DLL Implementation: `SCPI_Query(":MAC?", RetSTR)`
Ethernet Implementation: `:MAC?`

See Also

Get Model Name
Get Serial Number

5.1 (d) - Set Frequency

Description

This function sets the output frequency of the signal generator.

Command Syntax

`:FREQ:[freq][units]`

Variable	Description
[freq]	The frequency to set
[units]	The units for the frequency, "Hz", "kHz", "MHz", or "GHz"

Return String

`[status]`

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

Examples

String to Send	String Returned
:FREQ:2.105GHz	Frequency Set
:FREQ:2105MHz	Frequency Set

DLL Implementation: `SCPI_Command(":FREQ:2105MHZ")`
Ethernet Implementation: `:FREQ:2105MHZ`

See Also

Set Frequency
Query Frequency
Query Frequency Specification

5.1 (e) - Query Frequency

Description

This function returns the output frequency of the signal generator.

Query Syntax

`:FREQ?`

Return String

`[freq]`

String	Description
[freq]	The generator output frequency in MHz

Example

String to Send	String Returned
:FREQ?	2105.000000

DLL Implementation: `SCPI_Query(":FREQ?", RetSTR)`
Ethernet Implementation: `:FREQ?`

See Also

Set Frequency
Query Frequency Specification

5.1 (f) - Query Frequency Specification

Description

This function returns the specified frequency limits of the signal generator; the minimum frequency, maximum frequency and minimum step size frequency.

Query Syntax

`:FREQ:[spec]?`

Variable	Value	Description
[spec]	MAX	Return the maximum frequency specification
	MIN	Return the minimum frequency specification
	STEP	Return the minimum frequency step specification

Return String

`[freq] [units]`

String	Description
[freq]	The maximum or minimum output frequency specification
[units]	The units for the frequency specification, "Hz", "kHz", "MHz", or "GHz"

Examples

String to Send	String Returned
:FREQ:MAX?	6728.00 MHz
:FREQ:STEP?	3 Hz
:FREQ:MIN?	0.10 MHz

DLL Implementation: `SCPI_Query(":FREQ:MAX?", RetSTR)`
Ethernet Implementation: `:FREQ:MAX?`

See Also

Set Frequency
Query Frequency

5.1 (g) - Set Power

Description

This function sets the output power of the signal generator.

Command Syntax

`:PWR:[power]`

Variable	Description
[power]	The power to set in dBm

Return String

`[status]`

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

Examples

String to Send	String Returned
:PWR:8.5	1
:PWR:-10	1

DLL Implementation: `SCPI_Command(":PWR:8.5")`
Ethernet Implementation: `:PWR:8.5`

See Also

Query Power
Query Frequency Specification

5.1 (h) - Query Power

Description

This function returns the output power of the signal generator.

Query Syntax

`:PWR?`

Return String

`[power]`

String	Description
[power]	The generator output power in dBm, to 2 decimal places

Example

String to Send	String Returned
:PWR?	8.5
:PWR?	-10.0

DLL Implementation: `SCPI_Query(":PWR?", RetSTR)`
Ethernet Implementation: `:PWR?`

See Also

Set Power
Query Frequency Specification

5.1 (i) - Query Power Specification

Description

This function returns the specified power limits of the signal generator; minimum output power and maximum output power. The true minimum / maximum output power achievable by the generator is guaranteed to be good as the specified level across the full operating frequency and will be exceeded in some frequency bands.

Query Syntax

```
:PWR:[spec]?
```

Variable	Description
[spec]	The power spec to query, either "MAX" or "MIN"

Return String

```
[power]
```

String	Description
[power]	The maximum or minimum output power specification

Examples

String to Send	String Returned
:PWR:MAX?	15.0
:PWR:MIN?	-100.0

DLL Implementation: `SCPI_Query(":PWR:MAX?", RetSTR)`
Ethernet Implementation: `:PWR:MAX?`

See Also

Set Power
Query Power

5.1 (j) - Set RF Output On/Off

Description

This function sets the RF output, either on or off.

Command Syntax

:PWR:RF:[state]

Variable	Description
[state]	The output state to set, either "ON" or "OFF"

Return String

[status]

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

Examples

String to Send	String Returned
:PWR:RF:ON	1
:PWR:RF:OFF	1

DLL Implementation: SCPI_Command(":PWR:RF:ON")
Ethernet Implementation: :PWR:RF:ON

See Also

Set Frequency
Set Power
Query RF Output State (On/Off)

5.1 (k) - Query RF Output State (On/Off)

Description

This function returns the RF output state, either on or off.

Query Syntax

:PWR:RF?

Return String

[state]

String	Description
[state]	The output state, either "ON" or "OFF"

Examples

String to Send	String Returned
:PWR:RF?	ON
:PWR:RF?	OFF

DLL Implementation: `SCPI_Query(":PWR:RF?", RetSTR)`
Ethernet Implementation: `:PWR:RF?`

See Also

Set Frequency
Set Power
Set RF Output On/Off

5.1 (l) - Check External Reference

Description

This function indicates whether the signal generator has detected an external reference on the Ref In port. The signal generator will automatically switch from the internal to the external reference if a valid signal is detected.

Command Syntax

:EXTREFDETECT?

Return String

[status]

Variable	Value	Description
[status]	0	No external reference detected therefore internal reference will be used
	1	External reference detected and will be used

Examples

String to Send	String Returned
:EXTREFDETECT?	0

DLL Implementation: `SCPI_Query(":EXTREFDETECT?", RetSTR)`
Ethernet Implementation: `:EXTREFDETECT?`

5.1 (m) - Set Address

Description

This function sets the address of the signal generator for USB communication (1-255, default is 255).

Command Syntax

:ADD:[address]

Variable	Description
[address]	The preferred address, 1-255

Return String

[status]

Variable	Value	Description
[status]	Invalid Command	Command failed, possibly due to invalid address
	1	Command completed successfully

Examples

String to Send	String Returned
:ADD:3	1

DLL Implementation: `SCPI_Command(":ADD:3")`
Ethernet Implementation: `:ADD:3`

See Also

Get Address

5.1 (n) - Get Address

Description

This function returns the address of the signal generator for USB communication (1-255, default is 255).

Command Syntax

:ADD?

Return String

[address]

Variable	Description
[address]	Integer value (1-255) indicating the USB address of the device

Examples

String to Send	String Returned
:ADD?	3

DLL Implementation: SCPI_Query(":ADD?", RetSTR)
Ethernet Implementation: :ADD:?

See Also

Set Address

5.1 (o) - Set Trigger Out

Description

This function manually sets the generator's Trigger Out port to logic low, logic high, or 5ms pulses.

Command Syntax

`:TRIGGEROUT:STATE:[mode]`

Variable	Value	Description
[mode]	HIGH	Set Trigger Out to logic high
	LOW	Set Trigger Out to logic low
	PULSE	Pulse Trigger Out (5ms at logic high, 5ms at logic low)

Return String

`[status]`

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

Examples

String to Send	String Returned
:TRIGGEROUT:STATE:HIGH	1
:TRIGGEROUT:STATE:LOW	1
:TRIGGEROUT:STATE:PULSE	1

DLL Implementation: `SCPI_Command(":TRIGGEROUT:STATE:HIGH")`
Ethernet Implementation: `:TRIGGEROUT:STATE:HIGH`

See Also

Get Trigger In

5.1 (p) - Get Trigger In

Description

This function returns the logic state of the generator's Trigger In port (logic low or logic high).

Command Syntax

`:TRIGGERIN:STATE?`

Return String

`[state]`

Variable	Value	Description
[state]	1	Trigger In is logic high
	0	Trigger In is logic low

Examples

String to Send	String Returned
:TRIGGERIN:STATE?	1
:TRIGGERIN:STATE?	0

DLL Implementation: `SCPI_Query(":TRIGGERIN?", RetSTR)`
Ethernet Implementation: `:TRIGGERIN?`

See Also

Set Trigger Out

5.1 (q) - Get On/Off Counter

Description

This function returns a counter value indicating the number of times that the signal generator has been powered on in its lifetime.

Command Syntax

```
:ONOFFCOUNTER?
```

Return String

```
[count]
```

Variable	Description
[count]	The power on count

Examples

String to Send	String Returned
:ONOFFCOUNTER?	150

DLL Implementation: `SCPI_Query(":ONOFFCOUNTER?", RetSTR)`
Ethernet Implementation: `:ONOFFCOUNTER?`

See Also

Get Operation Timer

5.1 (r) - Get Operation Timer

Description

This function returns a timer value indicating the number of minutes that the signal generator has been powered on in its lifetime.

Command Syntax

:OPERATIONTIME?

Return String

[timer] minutes

Variable	Description
[timer]	The total "on" time of the generator in minutes

Examples

String to Send	String Returned
:OPERATIONTIME?	7503 minutes

DLL Implementation: `SCPI_Query(":OPERATIONTIME?", RetSTR)`
Ethernet Implementation: `:OPERATIONTIME?`

See Also

Get On/Off Counter

5.1 (s) - Get Firmware

Description

This function returns the revision code of the signal generator's internal firmware.

Command Syntax

:FIRMWARE?

Return String

[internal][firmware]

Variable	Description
[internal]	Three pairs of ASCII characters with no user meaning (reserved for internal use), for example "13 26 HS"
[firmware]	Pair of ASCII characters indicating the firmware version, for example "A9"

Examples

String to Send	String Returned
:FIRMWARE?	13 26 HSA9

DLL Implementation: `SCPI_Query(":FIRMWARE?", RetSTR)`
Ethernet Implementation: `:FIRMWARE?`

5.1 (t) - Check Internal Temperature

Description

Returns the temperature (degrees Celsius) from the generator's internal temperature sensor.

Applies To

Model Name
SSG-15G-RC

Command Syntax

`:TSENSOR?`

Return String

`[temperature]`

Variable	Description
[temperature]	The internal temperature in degrees Celsius

Examples

String to Send	String Returned
:TSENSOR?	+24.25

DLL Implementation: `SCPI_Query(":TSENSOR?", RetSTR)`
Ethernet Implementation: `:TSENSOR?`

5.2 - Power-Up Settings

These settings determine the initial output frequency, power level and state (on/off) that the signal generator will load when the DC power is connected.

The following models are supported:

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

5.2 (a) - Set Power-Up Mode

Description

Sets the generator's power-up mode, the initial output frequency and power level to be loaded when DC power is connected.
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

`:ONPOWERUP:CONFIG:[mode]`

Variable	Value	Description
[mode]	D	Factory default (maximum frequency and minimum power)
	L	Last known frequency and power (these parameters are saved to permanent memory every 3 minutes, when the GUI application is closed, or when the Save Data command is issued)
	U	User defined frequency and power (see Set Power-Up Frequency and Set Power-Up Power Level)

Return String

`[status]`

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

Examples

String to Send	String Returned
:ONPOWERUP:CONFIG:D	1
:ONPOWERUP:CONFIG:L	1
:ONPOWERUP:CONFIG:U	1

DLL Implementation: `SCPI_Command(":ONPOWERUP:CONFIG:L")`
Ethernet Implementation: `:ONPOWERUP:CONFIG:L`

See Also

Get Power-Up Mode
Set Power-Up Frequency
Set Power-Up Power Level
Set Power-Up Output State
Save Data

5.2 (b) - Get Power-Up Mode

Description

Returns the generator's power-up mode, the initial output frequency and power level to be loaded when DC power is connected.
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:ONPOWERUP:CONFIG?

Return String

[mode]

Variable	Value	Description
[mode]	D	Factory default (maximum frequency and minimum power)
	L	Last known frequency and power (these parameters are saved to permanent memory every 3 minutes, when the GUI application is closed, or when the Save Data command is issued)
	U	User defined frequency and power (see Get Power-Up Frequency and Get Power-Up Power Level)

Examples

String to Send	String Returned
:ONPOWERUP:CONFIG?	D
:ONPOWERUP:CONFIG?	L
:ONPOWERUP:CONFIG?	U

DLL Implementation: `SCPI_Query("ONPOWERUP:CONFIG?", RetSTR)`
Ethernet Implementation: `:ONPOWERUP:CONFIG?`

See Also

Set Power-Up Mode
Get Power-Up Frequency
Get Power-Up Power Level
Get Power-Up Output State
Save Data

5.2 (c) - Set Power-Up Frequency

Description

Sets the initial output frequency to be loaded when the generator's DC power is connected. This only applies when the power-up mode is set to "U" (user defined).
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:ONPOWERUP:FREQ:[freq]

Variable	Description
[freq]	Initial output frequency (MHz) to load when the generator is set to "user defined" power-up mode.

Return String

[status]

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

Examples

String to Send	String Returned
:ONPOWERUP:FREQ:1000.5	1

DLL Implementation: `SCPI_Command(":ONPOWERUP:FREQ:1000.5")`
Ethernet Implementation: `:ONPOWERUP:FREQ:1000.5`

See Also

Set Power-Up Mode
Get Power-Up Frequency
Set Power-Up Power Level
Set Power-Up Output State

5.2 (d) - Get Power-Up Frequency

Description

Returns the initial output frequency that will be loaded when the generator's DC power is connected. This only applies when the power-up mode is set to "U" (user defined).
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:ONPOWERUP:FREQ?

Return String

[mode]

Variable	Description
[freq]	Initial output frequency (MHz) that will load when the generator is set to "user defined" power-up mode.

Examples

String to Send	String Returned
:ONPOWERUP:FREQ?	1000.5

DLL Implementation: SCPI_Query(":ONPOWERUP:FREQ?", RetSTR)
Ethernet Implementation: :ONPOWERUP:FREQ?

See Also

Get Power-Up Mode
Set Power-Up Frequency
Get Power-Up Power Level
Get Power-Up Output State

5.2 (e) - Set Power-Up Power Level

Description

Sets the initial RF output power level to be loaded when the generator's DC power is connected. This only applies when the power-up mode is set to "U" (user defined).
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

```
:ONPOWERUP:PWR:[power]
```

Variable	Description
[power]	Initial output power level (dBm) to load when the generator is set to "user defined" power-up mode.

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:ONPOWERUP:PWR:-3.25	1

DLL Implementation: `SCPI_Command(":ONPOWERUP:PWR:-3.25")`
Ethernet Implementation: `:ONPOWERUP:PWR:-3.25`

See Also

Set Power-Up Mode
Set Power-Up Frequency
Get Power-Up Power Level
Set Power-Up Output State

5.2 (f) - Get Power-Up Power Level

Description

Returns the initial RF output power level that will be loaded when the generator's DC power is connected. This only applies when the power-up mode is set to "U" (user defined).
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load. See Set Power-Up Output State.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:ONPOWERUP:PWR?

Return String

[power]

Variable	Description
[power]	Initial output power level (MHz) that will load when the generator is set to "user defined" power-up mode.

Examples

String to Send	String Returned
:ONPOWERUP:PWR?	-3.25

DLL Implementation: `SCPI_Query(":ONPOWERUP:PWR?", RetSTR)`
Ethernet Implementation: `:ONPOWERUP:PWR?`

See Also

Get Power-Up Mode
Get Power-Up Frequency
Set Power-Up Power Level
Get Power-Up Output State

5.2 (g) - Set Power-Up Output State

Description

Sets the initial RF output state that applies when the generator's DC power is connected. The RF output can be enabled or disabled.
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

`:ONPOWERUP:PWRSTATE:[state]`

Variable	Value	Description
[state]	OFF	RF output will be disabled on power-up (recommended)
	ON	RF output will be enabled on power up

Return String

`[status]`

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

Examples

String to Send	String Returned
:ONPOWERUP:PWRSTATE:OFF	1
:ONPOWERUP:PWRSTATE:ON	1

DLL Implementation: `SCPI_Command(":ONPOWERUP:PWRSTATE:OFF")`
Ethernet Implementation: `:ONPOWERUP:PWRSTATE:OFF`

See Also

Set Power-Up Mode
Set Power-Up Frequency
Set Power-Up Power Level
Get Power-Up Output State

5.2 (h) - Get Power-Up Output State

Description

Returns the initial RF output state that applies when the generator's DC power is connected. The RF output can be enabled or disabled.
Note: By default, the RF output is disabled on power-up in order to avoid potential damage to the generator output and load in the event of excessive power or a mismatched load.

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:ONPOWERUP:PWRSTATE?

Return String

[state]

Variable	Value	Description
[state]	OFF	RF output will be disabled on power-up
	ON	RF output will be enabled on power up

Examples

String to Send	String Returned
:ONPOWERUP:PWRSTATE?	OFF
:ONPOWERUP:PWRSTATE?	ON

DLL Implementation: SCPI_Query("ONPOWERUP:PWRSTATE?", RetSTR)
Ethernet Implementation: :ONPOWERUP:PWRSTATE?

See Also

Get Power-Up Mode
Get Power-Up Frequency
Get Power-Up Power Level
Set Power-Up Output State

5.2 (i) - Save Data

Description

Saves the latest CW output state (frequency and power) to permanent memory so that the settings can be recalled then next time the generator is powered on. This only applies when the power-up mode is set to "L" (last known).

Applies To

Model Name	Firmware Requirement
SSG-6000RC	Firmware A4 or later
SSG-15G-RC	Firmware B6 or later

Command Syntax

:OPERATIONDATA:SAVE

Return String

[status]

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

Examples

String to Send	String Returned
:OPERATIONDATA:SAVE	1

DLL Implementation: SCPI_Command(":OPERATIONDATA:SAVE")
Ethernet Implementation: :OPERATIONDATA:SAVE

See Also

Set Power-Up Mode
Get Power-Up Mode

5.3 - Frequency Sweep Commands & Queries

The signal generator can be configured to produce an automatic, swept frequency output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

An example programming sequence to configure a sweep using SCPI on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.SCPI_Command(":FSWEEP:STARTFREQ:1000")
MyPTE1.SCPI_Command(":FSWEEP:STOPFREQ:6000")
MyPTE1.SCPI_Command(":FSWEEP:STEPSIZE:100")

' Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":FSWEEP:POWER:10")
MyPTE1.SCPI_Command(":FSWEEP:DWELL:10")

' Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":FSWEEP:MODE:ON")
```

Visual C++

```
// Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1->SCPI_Command(":FSWEEP:STARTFREQ:1000");
MyPTE1->SCPI_Command(":FSWEEP:STOPFREQ:6000");
MyPTE1->SCPI_Command(":FSWEEP:STEPSIZE:100");

// Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1->SCPI_Command(":FSWEEP:POWER:10");
MyPTE1->SCPI_Command(":FSWEEP:DWELL:10");

// Start the sweep
MyPTE1->SCPI_Command(":PWR:RF:ON");
MyPTE1->SCPI_Command(":FSWEEP:MODE:ON");
```

Visual C#

```
// Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.SCPI_Command(":FSWEEP:STARTFREQ:1000");
MyPTE1.SCPI_Command(":FSWEEP:STOPFREQ:6000");
MyPTE1.SCPI_Command(":FSWEEP:STEPSIZE:100");

// Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":FSWEEP:POWER:10");
MyPTE1.SCPI_Command(":FSWEEP:DWELL:10");

// Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON");
MyPTE1.SCPI_Command(":FSWEEP:MODE:ON");
```

Matlab

```
% Set sweep for 1000-6000MHz in 100MHz steps
MyPTE1.SCPI_Command(":FSWEEP:STARTFREQ:1000")
MyPTE1.SCPI_Command(":FSWEEP:STOPFREQ:6000")
MyPTE1.SCPI_Command(":FSWEEP:STEPSIZE:100")

% Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":FSWEEP:POWER:10")
MyPTE1.SCPI_Command(":FSWEEP:DWELL:10")

% Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":FSWEEP:MODE:ON")
```

The same sequence could be sent as a series of HTTP commands as below if the signal generator is connected through the Ethernet interface:

```
http://10.10.10.10/:FSWEEP:STARTFREQ:1000
http://10.10.10.10/:FSWEEP:STOPFREQ:6000
http://10.10.10.10/:FSWEEP:STEPSIZE:100

http://10.10.10.10/:FSWEEP:POWER:10
http://10.10.10.10/:FSWEEP:DWELL:10

http://10.10.10.10/:PWR:RF:ON
http://10.10.10.10/:FSWEEP:MODE:ON
```

The above assumes the generator has IP address 10.10.10.10 with HTTP communication configured for port 80 and password security disabled.

Full details of the commands for configuring a frequency sweep are covered in the following sections.

5.3 (a) - Frequency Sweep – Set Frequencies

Description

This function sets the start, stop or step frequency for the sweep (in MHz).

Command Syntax

```
:FSWEEP:[parameter]:[frequency]
```

Variable	Value	Description
[parameter]	STARTFREQ	Set the start frequency for the sweep
	STOPFREQ	Set the stop frequency for the sweep
	STEPSIZE	Set the step frequency for the sweep
[frequency]		The frequency to set in MHz

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:FSWEEP:STARTFREQ:10.50	1
:FSWEEP:STOPFREQ:5500	1
:FSWEEP:STEPSIZE:0.25	1

DLL Implementation: `SCPI_Command(":FSWEEP:STARTFREQ:10.50")`
Ethernet Implementation: `:FSWEEP:STARTFREQ:10.50`

See Also

Frequency Sweep – Get Frequencies
Frequency Sweep – Set Power
Frequency Sweep – Get Power
Frequency Sweep – Start/Stop Sweep

5.3 (b) - Frequency Sweep – Get Frequencies

Description

This function returns the start, stop or step frequency for the sweep (in MHz).

Command Syntax

:FSWEEP:[parameter]?

Variable	Value	Description
[parameter]	STARTFREQ	Return the start frequency for the sweep
	STOPFREQ	Return the stop frequency for the sweep
	STEPSIZE	Return the step frequency for the sweep

Return String

[freq]

Variable	Description
[freq]	The frequency in MHz

Examples

String to Send	String Returned
:FSWEEP:STARTFREQ?	1000.000000
:FSWEEP:STOPFREQ?	2000.000000
:FSWEEP:STEPSIZE?	25.000000

DLL Implementation: `SCPI_Query(":FSWEEP:STARTFREQ?", RetSTR)`
Ethernet Implementation: `:FSWEEP:STARTFREQ?`

See Also

Frequency Sweep – Set Frequencies
Frequency Sweep – Set Power
Frequency Sweep – Get Power

5.3 (c) - Frequency Sweep – Set Power

Description

This function sets a constant power level (in dBm) for a frequency sweep.

Command Syntax

`:FSWEEP:POWER:[power]`

Variable	Description
[power]	The power to set in dBm

Return String

`[status]`

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

Examples

String to Send	String Returned
:FSWEEP:POWER:10.50	1

DLL Implementation: `SCPI_Command(":FSWEEP:POWER:10.50")`
Ethernet Implementation: `:FSWEEP:POWER:10.50`

See Also

Frequency Sweep – Set Frequencies
Frequency Sweep – Get Frequencies
Frequency Sweep – Get Power
Frequency Sweep – Start/Stop Sweep

5.3 (d) - Frequency Sweep – Get Power

Description

This function returns the constant output power level (in dBm) for a frequency sweep.

Command Syntax

`:FSWEEP:POWER?`

Return String

`[power]`

Variable	Description
[power]	The output power level in dBm

Examples

String to Send	String Returned
:FSWEEP:POWER?	-10.5

DLL Implementation: `SCPI_Query(":FSWEEP:POWER?", RetStr)`
Ethernet Implementation: `:FSWEEP:POWER?`

See Also

Frequency Sweep – Set Frequencies
Frequency Sweep – Get Frequencies
Frequency Sweep – Set Power

5.3 (e) - Frequency Sweep – Set Dwell Time

Description

This function sets the dwell time to be used in a frequency sweep (the time in milliseconds for the generator to pause at each frequency point).

Command Syntax

`:FSWEEP:DWELL:[time]`

Variable	Description
[time]	The dwell time in ms

Return String

`[status]`

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

Examples

String to Send	String Returned
:FSWEEP:DWELL:100	1

DLL Implementation: `SCPI_Command(":FSWEEP:DWELL:100")`
Ethernet Implementation: `:FSWEEP:DWELL:100`

See Also

Frequency Sweep – Get Dwell Time
Frequency Sweep – Get Dwell Time Specification

5.3 (f) - Frequency Sweep – Get Dwell Time

Description

This function returns the dwell time to be used in a frequency sweep (the time in milliseconds that the generator will pause at each frequency point).

Command Syntax

`:FSWEEP:DWELL?`

Return String

`[time]`

Variable	Description
`[time]`	The dwell time in ms

Examples

String to Send	String Returned
`:FSWEEP:DWELL?`	`100`

DLL Implementation: `SCPI_Query(":FSWEEP:DWELL?", RetStr)`
Ethernet Implementation: `:FSWEEP:DWELL?`

See Also

Frequency Sweep – Set Dwell Time
Frequency Sweep – Get Dwell Time Specification

5.3 (g) - Frequency Sweep – Get Dwell Time Specification

Description

This function returns the minimum or maximum dwell time specifications for the generator.

Command Syntax

:FSWEEP:[spec]?

Variable	Value	Description
[spec]	MAXDWELL	Return the maximum specified dwell time
	MINDWELL	Return the minimum specified dwell time

Return String

[time]

Variable	Description
[time]	The dwell time in ms

Examples

String to Send	String Returned
:FSWEEP:MINDWELL?	20
:FSWEEP:MAXDWELL?	10000

DLL Implementation: `SCPI_Query(":FSWEEP:MINDWELL?", RetStr)`
Ethernet Implementation: `:FSWEEP:MINDWELL?`

See Also

Frequency Sweep – Set Dwell Time
Frequency Sweep – Get Dwell Time

5.3 (h) - Frequency Sweep – Set Direction

Description

This function sets the direction of a frequency sweep.

Command Syntax

`:FSWEEP:DIRECTION:[mode]`

Variable	Value	Description
[mode]	0	Forward (sweep from start to stop frequency). This is the default.
	1	Reverse (sweep from stop to start frequency)
	2	Bi-directional (sweep from start to stop, then stop to start frequency)

Return String

`[status]`

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

Examples

String to Send	String Returned
:FSWEEP:DIRECTION:1	1

DLL Implementation: `SCPI_Command(":FSWEEP:DIRECTION:1")`
Ethernet Implementation: `:FSWEEP:DIRECTION:1`

See Also

Frequency Sweep – Get Direction

5.3 (i) - Frequency Sweep – Get Direction

Description

This function returns the direction of a frequency sweep.

Command Syntax

`:FSWEEP:DIRECTION?`

Return String

`[mode]`

Variable	Value	Description
[mode]	0	Forward (sweep from start to stop frequency)
	1	Reverse (sweep from stop to start frequency)
	2	Bi-directional (sweep from start to stop, then stop to start frequency)

Examples

String to Send	String Returned
:FSWEEP:DIRECTION?	0

DLL Implementation: `SCPI_Query(":FSWEEP:DIRECTION?", RetStr)`
Ethernet Implementation: `:FSWEEP:DIRECTION?`

See Also

Frequency Sweep – Set Direction

5.3 (j) - Frequency Sweep – Set Trigger In Mode

Description

This function specifies how the frequency sweep should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each frequency point
2 – Wait for external trigger (Trigger In = logic 1) before starting each frequency sweep

Command Syntax

:FSWEEP:TRIGGERIN:[mode]

Variable	Value	Description
[mode]	0	Ignore trigger input (default)
	1	Wait for trigger before setting each frequency
	2	Wait for trigger before starting each sweep

Return String

[status]

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

Examples

String to Send	String Returned
:FSWEEP:TRIGGERIN:1	1

DLL Implementation: `SCPI_Command(":FSWEEP:TRIGGERIN:1")`
Ethernet Implementation: `:FSWEEP:TRIGGERIN:1`

See Also

Frequency Sweep – Set Trigger Out Mode
Frequency Sweep – Get Trigger In Mode
Frequency Sweep – Get Trigger Out Mode

5.3 (k) - Frequency Sweep – Set Trigger Out Mode

Description

This function specifies how the Trigger Out port will be used during the frequency sweep. The modes are:

0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each frequency point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each frequency sweep is initiated

Command Syntax

:FSWEEP:TRIGGEROUT:[mode]

Variable	Value	Description
[mode]	0	Trigger output disabled (default)
	1	Set Trigger Out on setting each frequency
	2	Set Trigger Out on starting each sweep

Return String

[status]

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

Examples

String to Send	String Returned
:FSWEEP:TRIGGEROUT:1	1

DLL Implementation: SCPI_Command(":FSWEEP:TRIGGEROUT:1")
Ethernet Implementation: :FSWEEP:TRIGGEROUT:1

See Also

Frequency Sweep – Set Trigger In Mode
Frequency Sweep – Get Trigger In Mode
Frequency Sweep – Get Trigger Out Mode

5.3 (l) - Frequency Sweep – Get Trigger In Mode

Description

This function returns a code to indicate how the frequency sweep will respond to an external trigger. The modes are:
0 – Trigger input ignored
1 – Wait for external trigger (Trigger In = logic 1) before setting each frequency point
2 – Wait for external trigger (Trigger In = logic 1) before starting each frequency sweep

Command Syntax

:FSWEEP:TRIGGERIN?

Return String

[mode]

Variable	Value	Description
[mode]	0	Ignore trigger input
	1	Wait for trigger before setting each frequency
	2	Wait for trigger before starting each sweep

Examples

String to Send	String Returned
:FSWEEP:TRIGGERIN?	0

DLL Implementation: `SCPI_Query(":FSWEEP:TRIGGERIN?", RetStr)`
Ethernet Implementation: `:FSWEEP:TRIGGERIN?`

See Also

Frequency Sweep – Set Trigger In Mode
Frequency Sweep – Set Trigger Out Mode
Frequency Sweep – Get Trigger Out Mode

5.3 (m) - Frequency Sweep – Get Trigger Out Mode

Description

This function returns a code to indicate how the Trigger Out port will be used during the frequency sweep. The modes are:
0 – Trigger output disabled
1 – Provide a trigger output (Trigger Out = logic 1) as each frequency point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each frequency sweep is initiated

Command Syntax

```
:FSWEEP:TRIGGEROUT?
```

Return String

```
[mode]
```

Variable	Value	Description
[mode]	0	Trigger output disabled
	1	Set Trigger Out on setting each frequency
	2	Set Trigger Out on starting each sweep

Examples

String to Send	String Returned
:FSWEEP:TRIGGEROUT?	0

DLL Implementation: `SCPI_Query(":FSWEEP:TRIGGEROUT?", RetStr)`
Ethernet Implementation: `:FSWEEP:TRIGGEROUT?`

See Also

Frequency Sweep – Set Trigger In Mode
Frequency Sweep – Set Trigger Out Mode
Frequency Sweep – Get Trigger In Mode

5.3 (n) - Frequency Sweep – Start/Stop Sweep

Description

This function starts or stops the frequency sweep using the previously defined parameters.

Command Syntax

`:FSWEEP:MODE:[mode]`

Variable	Value	Description
[mode]	ON	Start frequency sweep
	OFF	Stop frequency sweep

Return String

`[status]`

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

Examples

String to Send	String Returned
:FSWEEP:MODE:ON	1

DLL Implementation: `SCPI_Command(":FSWEEP:MODE:ON")`
Ethernet Implementation: `:FSWEEP:MODE:ON`

See Also

Frequency Sweep – Set Frequencies
Frequency Sweep – Set Power

5.4 - Power Sweep Commands & Queries

The signal generator can be configured to produce an automatic, swept power output, using the generator's internal timing systems. The user stores the parameters of the sweep in the generator's memory and can then enable/disable the sweep as required.

An example programming sequence to configure a sweep using SCPI on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.SCPI_Command(":PSWEEP:STARTPOWER:-20")
MyPTE1.SCPI_Command(":PSWEEP:STOPPOWER:20")
MyPTE1.SCPI_Command(":PSWEEP:STEPSIZE:0.5")

' Set fixed 1000MHz frequency and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":PSWEEP:FREQ:1000")
MyPTE1.SCPI_Command(":PSWEEP:DWELL:10")

' Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":PSWEEP:MODE:ON")
```

Visual C++

```
// Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1->SCPI_Command(":PSWEEP:STARTPOWER:-20");
MyPTE1->SCPI_Command(":PSWEEP:STOPPOWER:20");
MyPTE1->SCPI_Command(":PSWEEP:STEPSIZE:0.5");

// Set fixed 1000MHz frequency and 10ms dwell time for the sweep
MyPTE1->SCPI_Command(":PSWEEP:FREQ:1000");
MyPTE1->SCPI_Command(":PSWEEP:DWELL:10");

// Start the sweep
MyPTE1->SCPI_Command(":PWR:RF:ON");
MyPTE1->SCPI_Command(":PSWEEP:MODE:ON");
```

Visual C#

```
// Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.SCPI_Command(":PSWEEP:STARTPOWER:-20");
MyPTE1.SCPI_Command(":PSWEEP:STOPPOWER:+20");
MyPTE1.SCPI_Command(":PSWEEP:STEPSIZE:0.5");

// Set fixed 1000MHz frequency and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":PSWEEP:FREQ:1000");
MyPTE1.SCPI_Command(":PSWEEP:DWELL:10");

// Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON");
MyPTE1.SCPI_Command(":PSWEEP:MODE:ON");
```

Matlab

```
% Set sweep for -20dBm to +20dBm in 0.5dB steps
MyPTE1.SCPI_Command(":PSWEEP:STARTPOWER:-20")
MyPTE1.SCPI_Command(":PSWEEP:STOPPOWER:20")
MyPTE1.SCPI_Command(":PSWEEP:STEPSIZE:0.5")

% Set fixed 1000MHz frequency and 10ms dwell time for the sweep
MyPTE1.SCPI_Command(":PSWEEP:FREQ:1000")
MyPTE1.SCPI_Command(":PSWEEP:DWELL:10")

% Start the sweep
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":PSWEEP:MODE:ON")
```

The same sequence could be sent as a series of HTTP commands as below if the signal generator is connected through the Ethernet interface:

```
http://10.10.10.10/:PSWEEP:STARTPOWER:-20
http://10.10.10.10/:PSWEEP:STOPPOWER:20
http://10.10.10.10/:PSWEEP:STEPSIZE:0.5

http://10.10.10.10/:PSWEEP:FREQ:1000
http://10.10.10.10/:PSWEEP:DWELL:10

http://10.10.10.10/:PWR:RF:ON
http://10.10.10.10/:PSWEEP:MODE:ON
```

The above assumes the generator has IP address 10.10.10.10 with HTTP communication configured for port 80 and password security disabled.

Full details of the commands for configuring a power sweep are covered in the following sections.

5.4 (a) - Power Sweep – Set Power

Description

This function sets the start, stop or step power for the sweep (in dBm).

Command Syntax

`:PSWEEP:[parameter]:[power]`

Variable	Value	Description
[parameter]	STARTPOWER	Set the start power for the sweep
	STOPPOWER	Set the stop power for the sweep
	STEPSIZE	Set the power step for the sweep
[power]		The power to set in dBm

Return String

`[status]`

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

Examples

String to Send	String Returned
:PSWEEP:STARTPOWER:10.50	1
:PSWEEP:STOPPOWER:5500	1
:PSWEEP:STEPSIZE:0.25	1

DLL Implementation: `SCPI_Command(":PSWEEP:STARTPOWER:10.50")`
Ethernet Implementation: `:PSWEEP:STARTPOWER:10.50`

See Also

Power Sweep – Get Power
Power Sweep – Set Frequency
Power Sweep – Get Frequency
Power Sweep – Start/Stop Sweep

5.4 (b) - Power Sweep – Get Power

Description

This function returns the start, stop or step power for the sweep (in dBm).

Command Syntax

:PSWEEP:[parameter]?

Variable	Value	Description
[parameter]	STARTPOWER	Return the start power for the sweep in dBm
	STOPPOWER	Return the stop power for the sweep in dBm
	STEPSIZE	Return the power step for the sweep in dBm

Return String

[power]

Variable	Description
[power]	Power in dBm

Examples

String to Send	String Returned
:PSWEEP:STARTPOWER?	-20
:PSWEEP:STOPPOWER?	20
:PSWEEP:STEPSIZE?	0.5

DLL Implementation: `SCPI_Query(":PSWEEP:STARTPOWER?", RetStr)`
Ethernet Implementation: `:PSWEEP:STARTPOWER?`

See Also

Power Sweep – Set Power
Power Sweep – Set Frequency
Power Sweep – Get Frequency
Power Sweep – Start/Stop Sweep

5.4 (c) - Power Sweep – Set Frequency

Description

This function sets a constant frequency (in MHz) for a power sweep.

Command Syntax

:PSWEEP:FREQ:[frequency]

Variable	Description
[frequency]	The frequency to set in MHz

Return String

[status]

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

Examples

String to Send	String Returned
:PSWEEP:FREQ:1050.5	1

DLL Implementation: `SCPI_Command(":PSWEEP:FREQ:1050.5")`
Ethernet Implementation: `:PSWEEP:FREQ:1050.5`

See Also

Power Sweep – Set Power
Power Sweep – Get Power
Power Sweep – Get Frequency
Power Sweep – Start/Stop Sweep

5.4 (d) - Power Sweep – Get Frequency

Description

This function returns the constant frequency (in MHz) for a power sweep.

Command Syntax

:PSWEEP:FREQ?

Return String

[frequency]

Variable	Description
[frequency]	The frequency in MHz

Examples

String to Send	String Returned
:PSWEEP:FREQ?	1050.500000

DLL Implementation: SCPI_Command(":PSWEEP:FREQ?
Ethernet Implementation: :PSWEEP:FREQ?

See Also

Power Sweep – Set Power
Power Sweep – Get Power
Power Sweep – Set Frequency
Power Sweep – Start/Stop Sweep

5.4 (e) - Power Sweep – Set Dwell Time

Description

This function sets the dwell time to be used in a power sweep (the time in milliseconds for the generator to pause at each power point).

Command Syntax

```
:PSWEEP:DWELL:[time]
```

Variable	Description
[time]	The dwell time in ms

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:PSWEEP:DWELL:100	1

DLL Implementation: `SCPI_Command(":PSWEEP:DWELL:100")`
Ethernet Implementation: `:PSWEEP:DWELL:100`

See Also

Power Sweep – Get Dwell Time
Power Sweep – Get Dwell Time Specification

5.4 (f) - Power Sweep – Get Dwell Time

Description

This function returns the dwell time to be used in a power sweep (the time in milliseconds for the generator to pause at each power point).

Command Syntax

:PSWEEP:DWELL?

Return String

[time]

Variable	Description
[time]	The dwell time in ms

Examples

String to Send	String Returned
:PSWEEP:DWELL?	20

DLL Implementation: `SCPI_Query(":PSWEEP:DWELL?", RetStr)`
Ethernet Implementation: `:PSWEEP:DWELL?`

See Also

Power Sweep – Set Dwell Time
Power Sweep – Get Dwell Time Specification

5.4 (g) - Power Sweep – Get Dwell Time Specification

Description

This function returns the minimum or maximum dwell time specifications for the generator.

Command Syntax

```
:PSWEEP:[spec]?
```

Variable	Value	Description
[spec]	MAXDWELL	Return the maximum specified dwell time
	MINDWELL	Return the minimum specified dwell time

Return String

```
[time]
```

Variable	Description
[time]	The dwell time in ms

Examples

String to Send	String Returned
:PSWEEP:MINDWELL?	20
:PSWEEP:MAXDWELL?	10000

DLL Implementation: `SCPI_Query(":PSWEEP:MINDWELL?", RetStr)`
Ethernet Implementation: `:PSWEEP:MINDWELL?`

See Also

Power Sweep – Set Dwell Time
Power Sweep – Get Dwell Time

5.4 (h) - Power Sweep – Set Direction

Description

This function sets the direction of a power sweep.

Command Syntax

:PSWEEP:DIRECTION:[mode]

Variable	Value	Description
[mode]	0	Forward (sweep from start to stop power). This is the default.
	1	Reverse (sweep from stop to start power)
	2	Bi-directional (sweep from start to stop, then stop to start power)

Return String

[status]

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

Examples

String to Send	String Returned
:PSWEEP:DIRECTION:1	1

DLL Implementation: `SCPI_Command(":PSWEEP:DIRECTION:1")`
Ethernet Implementation: `:PSWEEP:DIRECTION:1`

See Also

Power Sweep – Get Direction

5.4 (i) - Power Sweep – Get Direction

Description

This function returns the direction of a power sweep.

Command Syntax

:PSWEEP:DIRECTION?

Return String

[mode]

Variable	Value	Description
[mode]	0	Forward (sweep from start to stop power)
	1	Reverse (sweep from stop to start power)
	2	Bi-directional (sweep from start to stop, then stop to start power)

Examples

String to Send	String Returned
:PSWEEP:DIRECTION?	0

DLL Implementation: `SCPI_Query(":PSWEEP:DIRECTION?", RetStr)`
Ethernet Implementation: `:PSWEEP:DIRECTION?`

See Also

Power Sweep – Set Direction

5.4 (j) - Power Sweep – Set Trigger In Mode

Description

This function specifies how the power sweep should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each point
2 – Wait for external trigger (Trigger In = logic 1) before starting each sweep

Command Syntax

```
:PSWEEP:TRIGGERIN:[mode]
```

Variable	Value	Description
[mode]	0	Ignore trigger input (default)
	1	Wait for trigger before setting each power
	2	Wait for trigger before starting each sweep

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:PSWEEP:TRIGGERIN:1	1

DLL Implementation: `SCPI_Command(":PSWEEP:TRIGGERIN:1")`
Ethernet Implementation: `:PSWEEP:TRIGGERIN:1`

See Also

Power Sweep – Set Trigger Out Mode
Power Sweep – Get Trigger In Mode
Power Sweep – Get Trigger Out Mode

5.4 (k) - Power Sweep – Set Trigger Out Mode

Description

This function specifies how the Trigger Out port will be used during the power sweep. The modes are:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each power is set
2 – Provide a trigger output (Trigger Out = logic 1) as each power sweep is initiated

Command Syntax

`:PSWEEP:TRIGGEROUT:[mode]`

Variable	Value	Description
[mode]	0	Trigger output disabled (default)
	1	Set Trigger Out on setting each power
	2	Set Trigger Out on starting each sweep

Return String

`[status]`

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

Examples

String to Send	String Returned
:PSWEEP:TRIGGEROUT:1	1

DLL Implementation: `SCPI_Command(":PSWEEP:TRIGGEROUT:1")`
Ethernet Implementation: `:PSWEEP:TRIGGEROUT:1`

See Also

Power Sweep – Set Trigger In Mode
Power Sweep – Get Trigger In Mode
Power Sweep – Get Trigger Out Mode

5.4 (l) - Power Sweep – Get Trigger In Mode

Description

This function returns a code to indicate how the power sweep will respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each point
2 – Wait for external trigger (Trigger In = logic 1) before starting each sweep

Command Syntax

:PSWEEP:TRIGGERIN?

Return String

[mode]

Variable	Value	Description
[mode]	0	Ignore trigger input
	1	Wait for trigger before setting each power
	2	Wait for trigger before starting each sweep

Examples

String to Send	String Returned
:PSWEEP:TRIGGERIN?	0

DLL Implementation: `SCPI_Query(":PSWEEP:TRIGGERIN?", RetStr)`
Ethernet Implementation: `:PSWEEP:TRIGGERIN?`

See Also

Power Sweep – Set Trigger In Mode
Power Sweep – Set Trigger Out Mode
Power Sweep – Get Trigger Out Mode

5.4 (m) - Power Sweep – Get Trigger Out Mode

Description

This function indicates how the Trigger Out port will be used during the power sweep. The modes are:
0 – Trigger output disabled
1 – Provide a trigger output (Trigger Out = logic 1) as each power is set
2 – Provide a trigger output (Trigger Out = logic 1) as each power sweep is initiated

Command Syntax

:PSWEEP:TRIGGEROUT?

Return String

[mode]

Variable	Value	Description
[mode]	0	Trigger output disabled
	1	Set Trigger Out on setting each power
	2	Set Trigger Out on starting each sweep

Examples

String to Send	String Returned
:PSWEEP:TRIGGEROUT?	0

DLL Implementation: `SCPI_Query(":PSWEEP:TRIGGEROUT?", RetStr)`
Ethernet Implementation: `:PSWEEP:TRIGGEROUT?`

See Also

Power Sweep – Set Trigger In Mode
Power Sweep – Set Trigger Out Mode
Power Sweep – Get Trigger In Mode

5.4 (n) - Power Sweep – Start/Stop Sweep

Description

This function starts or stops the power sweep using the previously defined parameters.

Command Syntax

`:PSWEEP:MODE:[mode]`

Variable	Value	Description
[mode]	ON	Start power sweep
	OFF	Stop power sweep

Return String

`[status]`

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

Examples

String to Send	String Returned
:PSWEEP:MODE:ON	1

DLL Implementation: `SCPI_Command(":PSWEEP:MODE:ON")`
Ethernet Implementation: `:PSWEEP:MODE:ON`

See Also

Power Sweep – Set Power
Power Sweep – Set Frequency

5.5 - Frequency/Power Hop Commands & Queries

The signal generator can be configured to automatically hop through a series of user defined frequency and power outputs using the generator's internal timing systems. The user stores the parameters of the hop sequence in the generator's memory and can then enable/disable the output as required.

An example programming sequence to configure a hop sequence using SCPI on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.SCPI_Command(":HOP:POINTS:50")
MyPTE1.SCPI_Command(":HOP:DWELL:10")

' Index point 1 and set to 1000MHz, -10dBm
MyPTE1.SCPI_Command(":HOP:POINT:0")
MyPTE1.SCPI_Command(":HOP:FREQ:1000")
MyPTE1.SCPI_Command(":HOP:POWER:-10")

' Index point 2 and set to 1100MHz, -8dBm
MyPTE1.SCPI_Command(":HOP:POINT:1")
MyPTE1.SCPI_Command(":HOP:FREQ:1100")
MyPTE1.SCPI_Command(":HOP:POWER:-8")

' Index and set points 2 TO 49 in the same way

' Start the hop sequence
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":HOP:MODE:ON")
```

Visual C++

```
// Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1->SCPI_Command(":HOP:POINTS:50");
MyPTE1->SCPI_Command(":HOP:DWELL:10");

// Index point 1 and set to 1000MHz, -10dBm
MyPTE1->SCPI_Command(":HOP:POINT:0");
MyPTE1->SCPI_Command(":HOP:FREQ:1000");
MyPTE1->SCPI_Command(":HOP:POWER:-10");

// Index point 2 and set to 1100MHz, -8dBm
MyPTE1->SCPI_Command(":HOP:POINT:1");
MyPTE1->SCPI_Command(":HOP:FREQ:1100");
MyPTE1->SCPI_Command(":HOP:POWER:-8");

// Index and set points 2 to 49 in the same way

// Start the hop sequence
MyPTE1->SCPI_Command(":PWR:RF:ON");
MyPTE1->SCPI_Command(":HOP:MODE:ON");
```

Visual C#

```
// Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.SCPI_Command(":HOP:POINTS:50");
MyPTE1.SCPI_Command(":HOP:DWELL:10");

// Index point 1 and set to 1000MHz, -10dBm
MyPTE1.SCPI_Command(":HOP:POINT:0");
MyPTE1.SCPI_Command(":HOP:FREQ:1000");
MyPTE1.SCPI_Command(":HOP:POWER:-10");

// Index point 2 and set to 1100MHz, -8dBm
MyPTE1.SCPI_Command(":HOP:POINT:1");
MyPTE1.SCPI_Command(":HOP:FREQ:1100");
MyPTE1.SCPI_Command(":HOP:POWER:-8");

// Index and set points 2 to 49 in the same way

// Start the hop sequence
MyPTE1.SCPI_Command(":PWR:RF:ON");
MyPTE1.SCPI_Command(":HOP:MODE:ON");
```

Matlab

```
% Declare a sequence of 50 points, set dwell time of 10ms
MyPTE1.SCPI_Command(":HOP:POINTS:50")
MyPTE1.SCPI_Command(":HOP:DWELL:10")

% Index point 1 and set to 1000MHz, -10dBm
MyPTE1.SCPI_Command(":HOP:POINT:0")
MyPTE1.SCPI_Command(":HOP:FREQ:1000")
MyPTE1.SCPI_Command(":HOP:POWER:-10")

% Index point 2 and set to 1100MHz, -8dBm
MyPTE1.SCPI_Command(":HOP:POINT:1")
MyPTE1.SCPI_Command(":HOP:FREQ:1100")
MyPTE1.SCPI_Command(":HOP:POWER:-8")

% Index and set points 2 to 49 in the same way

% Start the hop sequence
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":HOP:MODE:ON")
```

The same sequence could be sent as a series of HTTP commands as below if the signal generator is connected through the Ethernet interface:

```
http://10.10.10.10/:HOP:POINTS:50
http://10.10.10.10/:HOP:DWELL:10

http://10.10.10.10/:HOP:POINT:0
http://10.10.10.10/:HOP:FREQ:1000
http://10.10.10.10/:HOP:POWER:-10

http://10.10.10.10/:HOP:POINT:1
http://10.10.10.10/:HOP:FREQ:1100
http://10.10.10.10/:HOP:POWER:-8

… (Declare points 2 to 49) …

http://10.10.10.10/:PWR:RF:ON
http://10.10.10.10/:HOP:MODE:ON
```

The above assumes the generator has IP address 10.10.10.10 with HTTP communication configured for port 80 and password security disabled.

Full details of the commands for configuring a frequency/power hop series are covered in the following sections.

5.5 (a) - Hop – Set Number of Points

Description

This function sets the number of points to be used in a frequency/power hop.

Command Syntax

```
:HOP:POINTS:[number]
```

Variable	Description
[number]	The number of points in the hop (maximum is 500)

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:HOP:POINTS:50	1

DLL Implementation: `SCPI_Command(":HOP:POINTS:50")`
Ethernet Implementation: `:HOP:POINTS:50`

See Also

Hop – Get Number of Points
Hop – Get Maximum Number of Points

5.5 (b) - Hop – Get Number of Points

Description

This function returns the number of points to be used in the frequency/power hop.

Command Syntax

```
:HOP:POINTS?
```

Return String

```
[number]
```

Variable	Description
[number]	The number of points in the hop

Examples

String to Send	String Returned
:HOP:POINTS?	250

DLL Implementation: `SCPI_Query(":HOP:POINTS?", RetStr)`
Ethernet Implementation: `:HOP:POINTS?`

See Also

Hop – Get Number of Points
Hop – Get Maximum Number of Points

5.5 (c) - Hop – Get Maximum Number of Points

Description

This function returns the maximum number of points that can be used in a frequency/power hop.

Command Syntax

:HOP:MAXPOINTS?

Return String

[number]

Variable	Description
[number]	The maximum number of points allowed in a hop sequence

Examples

String to Send	String Returned
:HOP:MAXPOINTS?	100

DLL Implementation: SCPI_Query(":HOP:MAXPOINTS?", RetStr)
Ethernet Implementation: :HOP:MAXPOINTS?

See Also

Hop – Set Number of Points
Hop – Get Number of Points

5.5 (d) - Hop – Set Index Point

Description

This function specifies which point in the hop sequence is to be indexed so that it's frequency/power can be set (see Hop – Set Frequency of Indexed Point and Hop – Set Power of Indexed Point).

Command Syntax

:HOP:POINT:[index]

Variable	Description
[index]	The point to be indexed.

Return String

[status]

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

Examples

String to Send	String Returned
:HOP:POINT:1	1

DLL Implementation: `SCPI_Command(":HOP:POINT:1")`
Ethernet Implementation: `:HOP:POINT:1`

See Also

Hop – Get Index Point
Hop – Set Frequency of Indexed Point
Hop – Get Frequency of Indexed Point
Hop – Set Power of Indexed Point
Hop – Get Power of Indexed Point

5.5 (e) - Hop – Get Index Point

Description

This function returns the index number of the "active" point in the hop sequence, this is the point which is currently accessible (see Hop – Set Frequency of Indexed Point and Hop – Set Power of Indexed Point).

Command Syntax

`:HOP:POINT?`

Return String

`[index]`

Variable	Description
[index]	The index number of the active point in the hop sequence

Examples

String to Send	String Returned
:HOP:POINT?	5

DLL Implementation: `SCPI_Query(":HOP:POINT?", RetStr)`
Ethernet Implementation: `:HOP:POINT?`

See Also

Hop – Set Index Point
Hop – Set Frequency of Indexed Point
Hop – Get Frequency of Indexed Point
Hop – Set Power of Indexed Point
Hop – Get Power of Indexed Point

5.5 (f) - Hop – Set Frequency of Indexed Point

Description

This function sets the frequency in MHz of the "active" point in the hop sequence (the point that is currently indexed). Hop – Set Indexed Point should be called first to specify which point in the hop sequence is indexed.

Command Syntax

`:HOP:FREQ:[frequency]`

Variable	Description
[frequency]	The frequency (MHz) to set for 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:FREQ:1000.5	1

DLL Implementation: `SCPI_Command(":HOP:FREQ:1000.5")`
Ethernet Implementation: `:HOP:FREQ:1000.5`

See Also

Hop – Set Index Point
Hop – Get Index Point
Hop – Get Frequency of Indexed Point
Hop – Set Power of Indexed Point
Hop – Get Power of Indexed Point

5.5 (g) - Hop – Get Frequency of Indexed Point

Description

This function returns the frequency in MHz of the "active" point in the hop sequence (the point that is currently indexed). Hop – Set Indexed Point should be called first to specify which point in the hop sequence is indexed.

Command Syntax

```
:HOP:FREQ?
```

Return String

```
[frequency]
```

Variable	Description
[frequency]	The frequency (MHz) for the "active" (indexed) point in the hop sequence

Examples

String to Send	String Returned
:HOP:FREQ?	1500

DLL Implementation: `SCPI_Query(":HOP:FREQ?", RetStr)`
Ethernet Implementation: `:HOP:FREQ?`

See Also

Hop – Set Index Point
Hop – Get Index Point
Hop – Set Frequency of Indexed Point
Hop – Set Power of Indexed Point
Hop – Get Power of Indexed Point

5.5 (h) - Hop – Set Power of Indexed Point

Description

This function sets the power in dBm of the "active" point in the hop sequence (the point that is currently indexed). Hop – Set Indexed Point should be called first to specify which point in the hop sequence is indexed.

Command Syntax

`:HOP:POWER:[power]`

Variable	Description
[power]	The power (dBm) to set for 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:POWER:10	1

DLL Implementation: `SCPI_Command(":HOP:POWER:10")`
Ethernet Implementation: `:HOP:POWER:10`

See Also

Hop – Set Index Point
Hop – Get Index Point
Hop – Set Frequency of Indexed Point
Hop – Get Frequency of Indexed Point
Hop – Get Power of Indexed Point

5.5 (i) - Hop – Get Power of Indexed Point

Description

This function returns the power in dBm of the "active" point in the hop sequence (the point that is currently indexed). Hop – Set Indexed Point should be called first to specify which point in the hop sequence is indexed.

Command Syntax

`:HOP:POWER?`

Return String

`[power]`

Variable	Description
[power]	The power (dBm) for the "active" (indexed) point in the hop sequence

Examples

String to Send	String Returned
:HOP:POWER?	10

DLL Implementation: `SCPI_Query(":HOP:POWER?", RetStr)`
Ethernet Implementation: `:HOP:POWER?`

See Also

Hop – Set Index Point
Hop – Get Index Point
Hop – Set Frequency of Indexed Point
Hop – Get Frequency of Indexed Point
Hop – Set Power of Indexed Point

5.5 (j) - Hop – Set Dwell Time

Description

This function sets the dwell time to be used in a frequency/power hop sequence (the time in milliseconds for the generator to pause at each point).

Command Syntax

```
:HOP:DWELL:[time]
```

Variable	Description
[time]	The dwell time in ms

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:HOP:DWELL:100	1

DLL Implementation: `SCPI_Command(":HOP:DWELL:100")`
Ethernet Implementation: `:HOP:DWELL:100`

See Also

Hop – Get Dwell Time
Hop – Get Dwell Time Specification

5.5 (k) - Hop – Get Dwell Time

Description

This function returns the dwell time to be used in a frequency/power hop sequence (the time in milliseconds for the generator to pause at each power point).

Command Syntax

:HOP:DWELL?

Return String

[time]

Variable	Description
[time]	The dwell time in ms

Examples

String to Send	String Returned
:HOP:DWELL?	20

DLL Implementation: SCPI_Query(":HOP:DWELL?", RetStr)
Ethernet Implementation: :HOP:DWELL?

See Also

Hop – Set Dwell Time
Hop – Get Dwell Time Specification

5.5 (l) - Hop – Get Dwell Time Specification

Description

This function returns the minimum or maximum dwell time specifications for the generator.

Command Syntax

`:HOP:[spec]?`

Variable	Value	Description
[spec]	MAXDWELL	Return the maximum specified dwell time
	MINDWELL	Return the minimum specified dwell time

Return String

`[time]`

Variable	Description
[time]	The dwell time in ms

Examples

String to Send	String Returned
:HOP:MINDWELL?	10
:HOP:MAXDWELL?	1000

DLL Implementation: `SCPI_Query(":HOP:MINDWELL?", RetStr)`
Ethernet Implementation: `:HOP:MINDWELL?`

See Also

Hop – Set Dwell Time
Hop – Get Dwell Time

5.5 (m) - Hop – Set Direction

Description

This function sets the direction of a frequency/power hop sequence.

Command Syntax

`:HOP:DIRECTION:[mode]`

Variable	Value	Description
[mode]	0	Forward (hop in sequence from the first point in list to the last). This is the default.
	1	Reverse (hop in sequence from the last point in the list to the first)
	2	Bi-directional (hop in sequence from first to last, then last to first)

Return String

`[status]`

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

Examples

String to Send	String Returned
:HOP:DIRECTION:1	1

DLL Implementation: `SCPI_Command(":HOP:DIRECTION:1")`
Ethernet Implementation: `:HOP:DIRECTION:1`

See Also

Hop – Get Direction

5.5 (n) - Hop – Get Direction

Description

This function returns the direction of a frequency/power hop sequence.

Command Syntax

:HOP:DIRECTION?

Return String

[mode]

Variable	Value	Description
[mode]	0	Forward (hop in sequence from the first point in list to the last)
	1	Reverse (hop in sequence from the last point in list to the first)
	2	Bi-directional (hop in sequence first to last, then last to first)

Examples

String to Send	String Returned
:HOP:DIRECTION?	0

DLL Implementation: `SCPI_Query(":HOP:DIRECTION?", RetStr)`
Ethernet Implementation: `:HOP:DIRECTION?`

See Also

Hop – Set Direction

5.5 (o) - Hop – Set Trigger In Mode

Description

This function specifies how the hop sequence should respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each point
2 – Wait for external trigger (Trigger In = logic 1) before starting each hop sequence

Command Syntax

`:HOP:TRIGGERIN:[mode]`

Variable	Value	Description
[mode]	0	Ignore trigger input (default)
	1	Wait for trigger before setting each point
	2	Wait for trigger before starting each hop sequence

Return String

`[status]`

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

Examples

String to Send	String Returned
:HOP:TRIGGERIN:1	1

DLL Implementation: `SCPI_Command(":HOP:TRIGGERIN:1")`
Ethernet Implementation: `:HOP:TRIGGERIN:1`

See Also

Hop – Set Trigger Out Mode
Hop – Get Trigger In Mode
Hop – Get Trigger Out Mode

5.5 (p) - Hop – Set Trigger Out Mode

Description

This function specifies how the Trigger Out port will be used during the hop sequence. The modes are:
0 – Disable trigger output
1 – Provide a trigger output (Trigger Out = logic 1) as each point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each hop sequence is initiated

Command Syntax

```
:HOP:TRIGGEROUT:[mode]
```

Variable	Value	Description
[mode]	0	Trigger output disabled (default)
	1	Set Trigger Out on setting each point
	2	Set Trigger Out on starting each hop sequence

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:HOP:TRIGGEROUT:1	1

DLL Implementation: `SCPI_Command(":HOP:TRIGGEROUT:1")`
Ethernet Implementation: `:HOP:TRIGGEROUT:1`

See Also

Hop – Set Trigger In Mode
Hop – Get Trigger In Mode
Hop – Get Trigger Out Mode

5.5 (q) - Hop – Get Trigger In Mode

Description

This function returns a code to indicate how the hop sequence will respond to an external trigger. The modes are:
0 – Ignore trigger input
1 – Wait for external trigger (Trigger In = logic 1) before setting each point
2 – Wait for external trigger (Trigger In = logic 1) before starting each hop sequence

Command Syntax

```
:HOP:TRIGGERIN?
```

Return String

```
[mode]
```

Variable	Value	Description
[mode]	0	Ignore trigger input
	1	Wait for trigger before setting each point
	2	Wait for trigger before starting each hop sequence

Examples

String to Send	String Returned
:HOP:TRIGGERIN?	0

DLL Implementation: `SCPI_Query(":HOP:TRIGGERIN?", RetStr)`
Ethernet Implementation: `:HOP:TRIGGERIN?`

See Also

Hop – Set Trigger In Mode
Hop – Set Trigger Out Mode
Hop – Get Trigger Out Mode

5.5 (r) - Hop – Get Trigger Out Mode

Description

This function indicates how the Trigger Out port will be used during the hop sequence. The modes are:
0 – Trigger output disabled
1 – Provide a trigger output (Trigger Out = logic 1) as each point is set
2 – Provide a trigger output (Trigger Out = logic 1) as each hop sequence is initiated

Command Syntax

:HOP:TRIGGEROUT?

Return String

[mode]

Variable	Value	Description
[mode]	0	Trigger output disabled
	1	Set Trigger Out on setting each hop
	2	Set Trigger Out on starting each hop sequence

Examples

String to Send	String Returned
:HOP:TRIGGEROUT?	0

DLL Implementation: `SCPI_Query(":HOP:TRIGGEROUT?", RetStr)`
Ethernet Implementation: `:HOP:TRIGGEROUT?`

See Also

Hop – Set Trigger In Mode
Hop – Set Trigger Out Mode
Hop – Get Trigger In Mode

5.5 (s) - Hop – Start/Stop Hop Sequence

Description

This function starts or stops the hop sequence using the previously defined parameters.

Command Syntax

`:HOP:MODE:[mode]`

Variable	Value	Description
[mode]	ON	Start hop sequence
	OFF	Stop 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

DLL Implementation: `SCPI_Command(":HOP:MODE:ON")`
Ethernet Implementation: `:HOP:MODE:ON`

See Also

Hop – Set Index Point
Hop – Set Frequency of Indexed Point
Hop – Set Power of Indexed Point

5.6 - Regular Pulse Modulation Commands & Queries

The signal generator can be configured to produce repetitive RF pulse sequences with fixed frequency and power, supporting internal or external modulation and input / output trigger options. The user stores the parameters of the pulse sequence in the generator's memory and can then enable / disable the output as required.

An example programming sequence to configure a pulse sequence using SCPI on a signal generator connected via the USB interface would be as follows:

Visual Basic

```
' Set generator to 6GHz and 0.5dBm output
MyPTE1.SCPI_Command(":FREQ:6GHz")
MyPTE1.SCPI_Command(":PWR:0.5")

' Configure a pulse of 50us duration, 500us off time
MyPTE1.SCPI_Command(":PULSE:TIMEUNITS=USEC")
MyPTE1.SCPI_Command(":PULSE:TIMEON:50")
MyPTE1.SCPI_Command(":PULSE:TIMEOFF:500")

' Enable the output (continuous pulses)
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":PULSE:MODE:FREERUN:ON")
```

Visual C++

```
// Set generator to 6GHz and 0.5dBm output
MyPTE1->SCPI_Command(":FREQ:6GHz");
MyPTE1->SCPI_Command(":PWR:0.5");

// Set fixed 10dBm output power level and 10ms dwell time for the sweep
MyPTE1->SCPI_Command(":PULSE:TIMEUNITS=USEC");
MyPTE1->SCPI_Command(":PULSE:TIMEON:50");
MyPTE1->SCPI_Command(":PULSE:TIMEOFF:500");

// Enable the output (continuous pulses)
MyPTE1->SCPI_Command(":PWR:RF:ON");
MyPTE1->SCPI_Command(":PULSE:MODE:FREERUN:ON");
```

Visual C#

```
// Set generator to 6GHz and 0.5dBm output
MyPTE1.SCPI_Command(":FREQ:6GHz");
MyPTE1.SCPI_Command(":PWR:0.5");

// Configure a pulse of 50us duration, 500us off time
MyPTE1.SCPI_Command(":PULSE:TIMEUNITS=USEC");
MyPTE1.SCPI_Command(":PULSE:TIMEON:50");
MyPTE1.SCPI_Command(":PULSE:TIMEOFF:500");

// Enable the output (continuous pulses)
MyPTE1.SCPI_Command(":PWR:RF:ON");
MyPTE1.SCPI_Command(":PULSE:MODE:FREERUN:ON");
```

Matlab

```
% Set generator to 6GHz and 0.5dBm output
MyPTE1.SCPI_Command(":FREQ:6GHz")
MyPTE1.SCPI_Command(":PWR:0.5")

% Configure a pulse of 50us duration, 500us off time
MyPTE1.SCPI_Command(":PULSE:TIMEUNITS=USEC")
MyPTE1.SCPI_Command(":PULSE:TIMEON:50")
MyPTE1.SCPI_Command(":PULSE:TIMEOFF:500")

% Enable the output (continuous pulses)
MyPTE1.SCPI_Command(":PWR:RF:ON")
MyPTE1.SCPI_Command(":PULSE:MODE:FREERUN:ON")
```

The same sequence could be sent as a series of HTTP commands as below if the signal generator is connected through the Ethernet interface:

```
http://10.10.10.10/:FREQ:6GHz
http://10.10.10.10/:PWR:0.5

http://10.10.10.10/:PULSE:TIMEUNITS=USEC
http://10.10.10.10/:PULSE:TIMEON:50
http://10.10.10.10/:PULSE:TIMEOFF:500

http://10.10.10.10/:PWR:RF:ON
http://10.10.10.10/:PULSE:MODE:FREERUN:ON
```

The above assumes the generator has IP address 10.10.10.10 with HTTP communication configured for port 80 and password security disabled.

Full details of the commands for configuring a pulsed output are covered in the following sections.

5.6 (a) - Pulse Modulation – Set Pulse Period

Description

This function sets the on and off times of the pulse. The time will be interpreted in milliseconds or microseconds, as specified by Pulse Modulation – Set Time Units (the default is milliseconds).

Command Syntax

`:PULSE:[segment]:[time]`

Variable	Value	Description
[segment]	TIMEON	Set the pulse on period
	TIMEOFF	Set the pulse off period
[time]		The time to set (units determined by Pulse Modulation – Set Time Units, default is ms)

Return String

`[status]`

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

Examples

String to Send	String Returned
:PULSE:TIMEON:10	1
:PULSE:TIMEOFF:100	1

DLL Implementation: `SCPI_Command(":PULSE:TIMEON:10")`
Ethernet Implementation: `:PULSE:TIMEON:10`

See Also

Pulse Modulation – Set Time Units
Pulse Modulation – Enable Output & Trigger Mode

5.6 (b) - Pulse Modulation – Set Time Units

Description

This function sets the units to be used by the generator to interpret the pulse on and off times (the default is milliseconds).

Command Syntax

:PULSE:TIMEUNITS=[units]

Variable	Value	Description
[units]	MSEC	Set the time units as milliseconds
	USEC	Set the time units as microseconds

Return String

[status]

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

Examples

String to Send	String Returned
:PULSE:TIMEUNITS=MSEC	1
:PULSE:TIMEUNITS=USEC	1

DLL Implementation: `SCPI_Command(":PULSE:TIMEUNITS=USEC")`
Ethernet Implementation: `:PULSE:TIMEUNITS=USEC`

See Also

Pulse Modulation – Set Pulse Period
Pulse Modulation – Enable Output & Trigger Mode

5.6 (c) - Pulse Modulation – Enable Output & Trigger Mode

Description

This function sets the pulse modulation in to "free-running", "source triggered", or "externally modulated" mode and enables the RF output.

- Free-running mode will start a continuous series of pulses, at the previously specified CW frequency and power level, according to the previously specified pulse on and off times. The external trigger input is ignored in this mode.
- Source triggered mode will cause the generator to emit a single pulse, at the previously specified CW frequency and power level, every time an external trigger (logic high) is detected at the Trigger In port
- Externally modulated mode enables a pulsed output at the previously specified CW frequency and power level. The RF output will be enabled for as long as the Trigger In port is held at logic high and disabled when the Trigger In port is held at logic low.

Notes:

1 - When pulsed output is enabled via a USB connection, any subsequent command sent to the generator will turn off the RF output and disable pulsed mode.

2 - When pulsed output is enabled over an Ethernet, the generator cannot receive any further commands via HTTP or Telnet; a UDP "Magic Packet" must be sent in order to turn off the RF output and disable pulsed mode.

Command Syntax

```
:PULSE:MODE:[mode]:ON
```

Variable	Value	Description
[mode]	FREERUN	Enable the generator in free-running mode (continuous pulsed output with no external trigger)
	STRIGGER	Enable the generator in source triggered mode (single output pulse when an external trigger is received)
	SEXTERNAL	Enable the generator in externally modulated mode (continuous RF output while Trigger In is held at logic high; RF output disabled while Trigger In is at logic low)

Return String

```
[status]
```

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

Examples

String to Send	String Returned
:PULSE:MODE:FREERUN:ON	1
:PULSE:MODE:STRIGGER:ON	1
:PULSE:MODE:SEXTERNAL:ON	1

DLL Implementation: SCPI_Command(":PULSE:MODE:FREERUN:ON")
Ethernet Implementation: :PULSE:MODE:FREERUN:ON

See Also

Set Frequency
Set Power
Pulse Modulation – Turn Off Output (USB Mode)
Pulse Modulation – Turn Off Output (Ethernet Mode)

5.6 (d) - Pulse Modulation – Turn Off Output (USB Mode)

Description

Disables the RF output in pulsed mode.

Note: This function can only be used in conjunction with the DLL over a USB connection. To turn off a pulsed output when communicating over Ethernet, a "Magic Packet" must be sent using UDP (User Datagram Protocol).

Command Syntax

:PULSE:MODE:OFF

Return String

status]

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

Examples

String to Send	String Returned
:PULSE:MODE:OFF	1

DLL Implementation: SCPI_Command(":PULSE:MODE:OFF")
Ethernet Implementation:

See Also

Pulse Modulation – Enable Output & Trigger Mode
Pulse Modulation – Turn Off Output (Ethernet Mode)

5.6 (e) - Pulse Modulation – Turn Off Output (Ethernet Mode)

Description

Disables the RF output in pulsed mode when operating over an Ethernet connection. In this mode of operation the generator will not register incoming commands using HTTP or Telnet so a "Magic Packet" must be sent using UDP (User Datagram Protocol) to the signal generator's IP address.

The Magic Packet is made up of 6 bytes of decimal 255 (hex FF) followed by 16 repetitions of the signal generator's MAC address (1 byte per hex octet). The signal generator listens for UDP data on port 4950.

UDP does not offer a guarantee of service so it may be necessary to check that the generator's RF output was disabled.

Magic Packet

An example Magic Packet is represented below in hexadecimal notation for a signal generator with MAC address "D0-73-7F-86-5C-2B":

```
{FF FF FF FF FF FF D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86
5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F
86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B D0 73
7F 86 5C 2B D0 73 7F 86 5C 2B D0 73 7F 86 5C 2B}
```

See Also

Get MAC Address
Pulse Modulation – Enable Output & Trigger Mode
Pulse Modulation – Turn Off Output (USB Mode)

Example (Visual Basic.NET)

```
Private Sub Send_Magic_Packet()
  On Error GoTo err_Send_Magic_Packet

  ' Send a UDP Magic Packet to turn off the generator when pulsed output enabled
  ' Note: the generator will not accept HTTP/Telnet commands in this mode

  ' The generator's MAC and IP addresses are required
  Dim GenIPAddress As String = "192.168.9.59"
  Dim GenMacAddress As String = "D0-73-7F-86-5C-2B"
  Dim MacByte() As String = Split(GenMacAddress, "-")
  Dim intCounter As Integer = 0
  Dim sendBytes(0 To 101) As Byte

  ' Create the first 6 bytes of the magic packet (all decimal 255/hex FF)
  For i = 1 To 6
    sendBytes(intCounter) = &HFF
    intCounter += 1
  Next

  ' Create rest of packet, 16 repetitions of the MAC address (byte per octet)
  For i = 1 To 16
    For J = 0 To 5
      sendBytes(intCounter) = Byte.Parse(MacByte(J), Globalization.NumberStyles.HexNumber)
      intCounter += 1
    Next
  Next

  Dim BCIP As System.Net.IPAddress
  Dim EP As System.Net.IPEndPoint
  Dim UDP As New System.Net.Sockets.UdpClient

  ' Send to the generator's IP address
  BCIP = System.Net.IPAddress.Parse(GenIPAddress)
  ' Create the IP end point (the generator listens on port 4950)
  EP = New System.Net.IPEndPoint(BCIP, 4950)
  ' Send the magic packet
  UDP.Send(sendBytes, sendBytes.Length, EP)
  UDP.Close()

  MsgBox("Generator output should be disabled." & vbCrLf & vbCrLf & _
               "Note: UDP does not offer guarantee of delivery.", , "Magic Packet Sent")

exit_Send_Magic_Packet:
  UDP = Nothing
  Exit Sub

err_Send_Magic_Packet:
  MsgBox(Err.Description)
  Resume exit_Send_Magic_Packet
End Sub
```

5.7 - Dynamic Pulse Modulation Commands & Queries

The signal generator can be configured to create flexible RF pulse sequences with varying frequency, power, pulse width and interval per pulse. The user stores the parameters of the hop sequence in the generator's memory and can then enable/disable the output as required.

An example programming sequence to configure a dynamic pulse sequence could be sent as a series of HTTP commands as below if the signal generator is connected through the Ethernet interface:

```
--- Create a sequence of 3 pulses, with 100 repetitions ---
http://10.10.10.10/:DFS:NoOfPulses:3
http://10.10.10.10/:DFS:Cont:0
http://10.10.10.10/:DFS:NoOfCycles:100

--- Set point 0 ---
http://10.10.10.10/:DFS:Pulse_IDX:0
http://10.10.10.10/:DFS:RF:FREQ:1000
http://10.10.10.10/:DFS:RF:Power:5
http://10.10.10.10/:DFS:PulseWidth:1
http://10.10.10.10/:DFS:Interval:100

--- Set point 1 ---
http://10.10.10.10/:DFS:Pulse_IDX:1
http://10.10.10.10/:DFS:RF:FREQ:1100
http://10.10.10.10/:DFS:RF:Power:5
http://10.10.10.10/:DFS:PulseWidth:2
http://10.10.10.10/:DFS:Interval:150

--- Set point 2 ---
http://10.10.10.10/:DFS:Pulse_IDX:2
http://10.10.10.10/:DFS:RF:FREQ:1200
http://10.10.10.10/:DFS:RF:Power:5
http://10.10.10.10/:DFS:PulseWidth:3
http://10.10.10.10/:DFS:Interval:200

--- Start the sequence ---
http://10.10.10.10/:DFS:MODE:ON
```

The above assumes the generator has IP address 10.10.10.10 with HTTP communication configured for port 80 and password security disabled.

Full details of the commands for configuring a dynamic pulse sequence are covered in the following sections.

5.7 (a) - Dynamic Pulses – Set Number of Points

Description

Sets the number of points to be used in a dynamic pulse sequence.

Command Syntax

:DFS:NoOfPulses:[number]

Variable	Description
[number]	The number of points in the sequence (from 1 to 100)

Return String

[status]

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

Examples

String to Send	String Returned
:DFS:NoOfPulses:50	1

DLL Implementation: `SCPI_Command(":DFS:NoOfPulses:50")`
Ethernet Implementation: `:DFS:NoOfPulses:50`

See Also

5.7 (b) - Dynamic Pulses – Get Number of Points

Description

Returns the number of points to be used in a dynamic pulse sequence.

Command Syntax

:DFS:NoOfPulses?

Return String

[number]

Variable	Description
[number]	The number of points in the hop

Examples

String to Send	String Returned
:DFS:NoOfPulses?	50

DLL Implementation: SCPI_Query(":DFS:NoOfPulses?", RetStr)
Ethernet Implementation: :DFS:NoOfPulses?

See Also

5.7 (c) - Dynamic Pulses – Set Index Point

Description

Specifies which point in the pulse sequence is to be indexed so that its parameters can be set.

Command Syntax

`:DFS:Pulse_IDX:[index]`

Variable	Description
[index]	The point to be indexed (from 0 to 99)

Return String

`[status]`

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

Examples

String to Send	String Returned
:DFS:Pulse_IDX:1	1

DLL Implementation: `SCPI_Command(":DFS:Pulse_IDX:1")`
Ethernet Implementation: `:DFS:Pulse_IDX:1`

See Also

5.7 (d) - Dynamic Pulses– Get Index Point

Description

Returns the index number of the "active" point in the pulse sequence.

Command Syntax

```
:DFS:Pulse_IDX?
```

Return String

```
[index]
```

Variable	Description
[index]	The index number of the active point in the sequence

Examples

String to Send	String Returned
:DFS:Pulse_IDX?	1

DLL Implementation: `SCPI_Query(":DFS:Pulse_IDX?", RetStr)`
Ethernet Implementation: `:DFS:Pulse_IDX?`

See Also

5.7 (e) - Dynamic Pulses – Set Frequency of Indexed Point

Description

Sets the frequency in MHz of the "active" point in the pulse sequence (the point that is currently indexed).

Command Syntax

:DFS:RF:FREQ:[frequency]

Variable	Description
[frequency]	The frequency (MHz) to set for the indexed point

Return String

[status]

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

Examples

String to Send	String Returned
:DFS:RF:FREQ:1000	1

DLL Implementation: SCPI_Command(":DFS:RF:FREQ:1000.5")
Ethernet Implementation: :DFS:RF:FREQ:1000.5

See Also

5.7 (f) - Dynamic Pulses – Get Frequency of Indexed Point

Description

Returns the frequency in MHz of the "active" point in the pulse sequence (the point that is currently indexed).

Command Syntax

:DFS:RF:FREQ?

Return String

[frequency]

Variable	Description
[frequency]	The frequency (MHz) for the "active" (indexed) point in the sequence

Examples

String to Send	String Returned
:DFS:FREQ?	1000

DLL Implementation: SCPI_Query(":DFS:RF:FREQ?", RetStr)
Ethernet Implementation: :DFS:RF:FREQ?

See Also

5.7 (g) - Dynamic Pulses – Set Power of Indexed Point

Description

Sets the power in dBm of the "active" point in the sequence (the point that is currently indexed).

Command Syntax

`:DFS:RF:POWER:[power]`

Variable	Description
[power]	The power (dBm) to set for the indexed point

Return String

`[status]`

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

Examples

String to Send	String Returned
:DFS:RF:POWER:10	1

DLL Implementation: `SCPI_Command(":DFS:RF:POWER:10")`
Ethernet Implementation: `:DFS:RF:POWER:10`

See Also

5.7 (h) - Dynamic Pulses – Get Power of Indexed Point

Description

Returns the power in dBm of the "active" point in the sequence (the point that is currently indexed).

Command Syntax

:DFS:RF:POWER?

Return String

[power]

Variable	Description
[power]	The power (dBm) for the "active" (indexed) point in the sequence

Examples

String to Send	String Returned
:DFS:RF:POWER?	10

DLL Implementation: `SCPI_Query(":DFS:RF:POWER?", RetStr)`
Ethernet Implementation: `:DFS:RF:POWER?`

See Also

5.7 (i) - Dynamic Pulses – Set Pulse Width

Description

Sets the pulse width ("on" time) in microseconds for the indexed point in the sequence.

Command Syntax

:DFS:PulseWidth:[time]

Variable	Description
[time]	The pulse width in microseconds

Return String

[status]

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

Examples

String to Send	String Returned
:DFS:PulseWidth:100	1

DLL Implementation: `SCPI_Command(":DFS:PulseWidth:100")`
Ethernet Implementation: `:DFS:PulseWidth:100`

See Also

5.7 (j) - Dynamic Pulses – Get Pulse Width

Description

Returns the pulse width (“on” time) in microseconds for the indexed point in the sequence.

Command Syntax

:DFS:PulseWidth?

Return String

[time]

Variable	Description
[time]	The pulse width in microseconds

Examples

String to Send	String Returned
:DFS:PulseWidth?	100

DLL Implementation: `SCPI_Query(":DFS:PulseWidth?", RetStr)`
Ethernet Implementation: `:DFS:PulseWidth?`

See Also

5.7 (k) - Dynamic Pulses – Set Pulse Interval

Description

Sets the pulse interval (delay before the next pulse) in microseconds for the indexed point in the sequence.

Command Syntax

`:DFS:Interval:[time]`

Variable	Description
[time]	The pulse interval in microseconds

Return String

`[status]`

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

Examples

String to Send	String Returned
:DFS:Interval:200	1

DLL Implementation: `SCPI_Command(":DFS:Interval:200")`
Ethernet Implementation: `:DFS:Interval:200`

See Also

5.7 (l) - Dynamic Pulses – Get Pulse Interval

Description

Returns the pulse interval (delay before the next pulse) in microseconds for the indexed point in the sequence.

Command Syntax

:DFS:Interval?

Return String

[time]

Variable	Description
[time]	The pulse interval in microseconds

Examples

String to Send	String Returned
:DFS:Interval?	200

DLL Implementation: `SCPI_Query(":DFS:Interval?", RetStr)`
Ethernet Implementation: `:DFS:Interval?`

See Also

5.7 (m) - Dynamic Pulses – Set Number of Cycles

Description

Sets the number of cycles for which the complete dynamic pulse sequence should be repeated. Continuous mode must be disabled in order for this setting to take effect.

Command Syntax

`:DFS:NoOfCycles:[time]`

Variable	Description
[cycles]	The number of cycles

Return String

`[status]`

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

Examples

String to Send	String Returned
:DFS:NoOfCycles:100	1

DLL Implementation: `SCPI_Command(":DFS:NoOfCycles:100")`
Ethernet Implementation: `:DFS:NoOfCycles:100`

See Also

5.7 (n) - Dynamic Pulses – Get Number of Cycles

Description

Returns the number of cycles for which the complete dynamic pulse sequence should be repeated. Continuous mode must be disabled in order for this setting to take effect.

Command Syntax

:DFS:NoOfCycles?

Return String

[cycles]

Variable	Description
[cycles]	The number of cycles

Examples

String to Send	String Returned
:DFS:NoOfCycles?	100

DLL Implementation: `SCPI_Query(":DFS:NoOfCycles?", RetStr)`
Ethernet Implementation: `:DFS:NoOfCycles?`

See Also

5.7 (o) - Dynamic Pulses – Set Continuous Mode

Description

When enabled the sequence will run indefinitely (until another command is received by the generator). When disabled the sequence will repeat for the number of cycles specified in "Set Number of Cycles".

Command Syntax

:DFS:CONT:[mode]

Variable	Value	Description
[mode]	0	Disable continuous mode
	1	Enable continuous mode

Return String

[status]

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

Examples

String to Send	String Returned
:DFS:CONT:1	1

DLL Implementation: SCPI_Command(":DFS:CONT:1")
Ethernet Implementation: :DFS:CONT:1

See Also

5.7 (p) - Dynamic Pulses – Check Continuous Mode

Description

Checks whether continuous mode is enabled or disabled. When enabled the sequence will run indefinitely (until another command is received by the generator). When disabled the sequence will repeat for the number of cycles specified in “Set Number of Cycles”.

Command Syntax

:DFS:CONT?

Return String

[mode]

Variable	Value	Description
[mode]	0	Continuous mode is disabled
	1	Continuous mode is enabled

Examples

String to Send	String Returned
:DFS:CONT?	1

DLL Implementation: `SCPI_Query(":DFS:CONT?", RetStr)`
Ethernet Implementation: `:DFS:CONT?`

See Also

5.7 (q) - Dynamic Pulses – Start/Stop Hop Sequence

Description

Starts or stops the hop sequence using the previously defined parameters.

Command Syntax

:DFS:MODE:[mode]

Variable	Value	Description
[mode]	ON	Start sequence
	OFF	Stop sequence

Return String

[status]

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

Examples

String to Send	String Returned
:DFS:MODE:ON	1

DLL Implementation: SCPI_Command(":DFS:MODE:ON")
Ethernet Implementation: :DFS:MODE:ON

See Also