Chapter 3 - Synthesized Signal Generators

3.1 - Operating in a Windows Environment

3.1.1 - Referencing the DLL Library

The DLL file is installed in the host PC's system folders using the steps outlined above. In order to use the DLL functionality, some programming environments will require the user to set a reference to the relevant file, usually through a built in GUI in the programming environment.

Once this is done, the user just needs to instantiate a new instance of the USB_Gen object in order to use the signal generator functions. The details of this vary greatly between programming environments and languages but Mini-Circuits can provide detailed support on request. A new signal generator object would need to be initialized for every USB signal generator that the user wishes to control. In the following examples, MyPTE1 and MyPTE2 will be used as names of 2 declared generator objects.

Examples

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
```

3.1.2 - 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.

1) Short Connect (Optional String SN)
2) Short ConnectByAddress (Optional Short Address)
3) Void Disconnect ()
4) Short Read_ModelName (String ModelName)
5) Short Read_SN (String SN)
6) Short Set_Address (Short Address)
7) Short Get_Address ()
8) Short Get_Available_SN_List (String SN_List)
9) Short Get_Available_Address_List (String Add_List)
10) Short SetPowerON ()
11) Short SetPowerOFF ()
12) Short SetFreqAndPower (Double Fr, Float Pr, Short TriggerOut)
13) Short SetFreq (Double Fr, Short TriggerOut)
14) Short SetPower (Float Pr, Short TriggerOut)
15) Short GetGenStatus(Byte Locked, Short PowerIsOn, Double Fr , Float pr,
_ Short UNLEVELHigh, Short UNLEVELLow)
16) Short Set_Noise_Spur_Mode (Short nsm)
17) Short ExtRefDetected ()
18) Short GetTriggerIn_Status ()
19) Short Set_PulseMode (Short T_OFF, Short T_ON, Short Tunit)
20) Short Set_PulseMode_Trigger (Short TriggerType, Short T_ON, Short Tunit)
21) Float GetGenMaxFreq ()
22) Float GetGenMinFreq ()
23) Float GetGenStepFreq ()
24) Float GetGenMaxPower ()
25) Float GetGenMinPower ()
26) Float GetDeviceTemperature ()
27) Short Check_Connection ()
28) Short GetFirmware ()
29) Short FSweep_GetDirection ()
30) Short FSweep_GetDwell ()
31) Short FSweep_GetMaxDwell ()
32) Short FSweep_GetMinDwell ()
33) Float FSweep_GetPower ()
34) Double FSweep_GetStartFreq ()
35) Double FSweep_GetStopFreq ()
36) Double FSweep_GetStepSize ()
37) Short FSweep_GetTriggerIn ()
38) Short FSweep_GetTriggerOut ()
39) Short FSweep_SetDirection (Short SweepDirection)
40) Short FSweep_SetDwell (Short dwell_msec)
41) Short Fsweep_SetMode (Short onoff)
42) Float FSweep_SetPower (Float Pr)
43) Short FSweep_SetStartFreq (Double Fr)

44) Short FSweep_SetStopFreq (Double Fr)
45) Short FSweep_SetStepSize (Double Fr)
46) Short FSweep_SetTriggerIn (Short SweepTriggerIn)
47) Short FSweep_SetTriggerOut (Short SweepTriggerOut)
48) Short Hop_GetDirection ()
49) Short Hop_GetDwell ()
50) Short Hop_GetMaxDwell ()
51) Short Hop_GetMinDwell ()
52) Short Hop_GetMaxNoOfPoints ()
53) Short Hop_GetPoint (Short PointNo, Double HopFreq, float HopPower)
54) Short Hop_GetTriggerIn ()
55) Short Hop_GetTriggerOut ()
56) Short Hop_SetDirection (Short HopDirection)
57) Short Hop_SetDwell (Short dwell_msec)
58) Short Hop_SetMode (Short onoff)
59) Short Hop_SetNoOfPoints (Short HopNoOfPoints)
60) Short Hop_SetPoint (Short PointNo, Double HopFreq, Float HopPower)
61) Short Hop_SetTriggerIn (Short HopTriggerIn)
62) Short Hop_SetTriggerOut (Short HopTriggerOut)
63) Short PSweep_GetDirection ()
64) Short PSweep_GetDwell ()
65) Short PSweep_GetMaxDwell ()
66) Short PSweep_GetMinDwell ()
67) Double PSweep_GetFreq ()
68) Float PSweep_GetStartPower ()
69) Float PSweep_GetStopPower ()
70) Float PSweep_GetStepSize ()
71) Short PSweep_GetTriggerIn ()
72) Short PSweep_GetTriggerOut ()
73) Short PSweep_SetDirection (Short SweepDirection)
74) Short PSweep_SetDwell (Short dwell_msec)
75) Short Psweep_SetMode (Short onoff)
76) Double PSweep_SetFreq (Float Fr)
77) Short PSweep_SetStartPower (Float Pr)
78) Short PSweep_SetStopPower (Float Pr)
79) Short PSweep_SetStepSize (Float Pr)
80) Short PSweep_SetTriggerIn (Short SweepTriggerIn)
81) Short PSweep_SetTriggerOut (Short SweepTriggerOut)

3.1.3 - Detailed Description of DLL Functions

3.1.3 (1) - Connect to Signal Generator

Declaration

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

Description

This function is called to initialize the connection to a USB signal generator. If multiple generators are connected to the same computer, then the serial number should be included, otherwise this can be omitted. The connection process can take a few milliseconds so it is recommended that the connection be made once at the beginning of the routine and left open until the 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 USB 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

3.1.3 (2) - Connect to Signal Generator by Address

Declaration

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

Description

This function is called to initialize the connection to a USB 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 USB 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

3.1.3 (3) - Disconnect from Signal Generator

Declaration

```
Void Disconnect()
```

Description

This function is called to close the connection to the signal generator. It is strongly recommended that this function is used prior to ending the program. Failure to do so may result in a connection problem with the device. Should this occur, shut down the program and unplug the signal generator from the computer, then reconnect the signal generator before attempting to start again.

Parameters

Data Type	Variable	Description
None		

Return Values

Data Type	Value	Description
None		

Examples

Visual Basic

```
MyPTE1.Disconnect()
```

Visual C++

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

Visual C#

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

Matlab

```
MyPTE1.Disconnect
```

See Also

Connect to Signal Generator
Connect to Signal Generator by Address

3.1.3 (4) - 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 then
        {
                msgbox('The connected generator is ', ModelName)
                        % Display a message stating the model name
        }
```

See Also

Read Serial Number of Signal Generator

3.1.3 (5) - 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 then
        {
                msgbox('The connected generator is ', SN)
                        % Display a message stating the serial number
        }
```

See Also

Connect to Signal Generator
Get List of Connected Serial Numbers

3.1.3 (6) - 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

3.1.3 (7) - 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

3.1.3 (8) - 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 available (currently connected) signal generators.

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 then
{
        % split the List into array of SN's
}
```

See Also

Get List of Available Addresses

3.1.3 (9) - 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 connected generator matrices.

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 then
{       % split the List into array of Addresses
}
```

See Also

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

3.1.3 (10) - 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

3.1.3 (11) - 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

3.1.3 (12) - Set Output Frequency and Power

Declaration

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

Description

This function sets the RF ouput frequency and power level of the signal generator and enables or disables the “trigger out” function.

Note: For SSG-4000LH and SSG-4000HP models with serial numbers up to 11207100000, enabling the Trigger Out function will disable Trigger In since a common port is used. SSG-4000LH and SSG-4000HP models with serial numbers greater than 11207100000 (and all other SSG models) have separate Trigger ports so enabling Trigger Out has no affect on Trigger In.

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

3.1.3 (13) - Set Output Frequency

Declaration

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

Description

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

Note: For SSG-4000LH and SSG-4000HP models with serial numbers up to 11207100000, enabling the Trigger Out function will disable Trigger In since a common port is used. SSG-4000LH and SSG-4000HP models with serial numbers greater than 11207100000 (and all other SSG models) have separate Trigger ports so enabling Trigger Out has no affect on Trigger In.

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

3.1.3 (14) - Set Output Power

Declaration

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

Description

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

Note: For SSG-4000LH and SSG-4000HP models with serial numbers up to 11207100000, enabling the Trigger Out function will disable Trigger In since a common port is used. SSG-4000LH and SSG-4000HP models with serial numbers greater than 11207100000 (and all other SSG models) have separate Trigger ports so enabling Trigger Out has no affect on Trigger In.

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

3.1.3 (15) - 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

3.1.3 (16) - Set Low Noise or Low Spur Mode

Declaration

```
Short Set_Noise_Spur_Mode(Short nsm)
```

Description

This function applies to SSG-4000LH and SSG-4000HP only. It sets the generator in either "low noise" mode (for best phase noise performance) or "low spur" mode (for best spurious performance). The generator defaults to "low noise" mode.

Parameters

Data Type	Variable	Description
Short	nsm	Required. Integer value 1 or 0 to set the generator noise/spur mode; 1 = low spur mode, 0 = low noise mode.

Return Values

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

Examples

Visual Basic

```
status = MyPTE1.Set_Noise_Spur_Mode(nsm)
```

Visual C++

```
status = MyPTE1->Set_Noise_Spur_Mode(nsm);
```

Visual C#

```
status = MyPTE1.Set_Noise_Spur_Mode(nsm);
```

Matlab

```
status = MyPTE1.Set_Noise_Spur_Mode(nsm)
```

See Also

Get Trigger In Status
Check External Reference
Get Temperature of Signal Generator

3.1.3 (17) - Check External Reference

Declaration

```
Short ExtRefDetected()
```

Description

This function checks whether an external 10MHz reference is connected to the generator. The generator automatically uses the external reference if it is present, otherwise the internal reference is used.

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

Set Low Noise or Low Spur Mode
Get Trigger In Status
Get Temperature of Signal Generator

3.1.3 (18) - 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.

For SSG-4000LH and SSG-4000HP models with serial numbers up to 11207100000, the Trigger In function is only active when Trigger Out is disabled (see Set Output Frequency and Power). SSG-4000LH and SSG-4000HP models with serial numbers greater than 11207100000 (and all other SSG models) have separate Trigger In and Trigger Out ports so both can be used independently.

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 Low Noise or Low Spur Mode
Check External Reference
Get Temperature of Signal Generator

3.1.3 (19) - 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

3.1.3 (20) - 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).

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

3.1.3 (21) - 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

3.1.3 (22) - 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

3.1.3 (23) - Get Generator Step Size Spec

Declaration

```
Float GetGenStepFreq()
```

Description

This function reports the generator's 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

3.1.3 (24) - Get Generator Maximum Power Spec

Declaration

```
Float GetGenMaxPower()
```

Description

This function reports the maximum output power specification in dBm for the active generator. Requesting a higher output level than this value will result in the generator setting the UnLevelHigh flag in order to report an inaccurate power level (see Get Generator Output Status). Actual minimum power will vary between units.

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

3.1.3 (25) - Get Generator Minimum Power Spec

Declaration

```
Float GetGenMinPower()
```

Description

This function reports the minimum output power specification in dBm for the active generator. Requesting a lower output level than this value will result in the generator setting the UnLevelLow flag in order to report an inaccurate power level (see Get Generator Output Status). Actual minimum power will vary between units.

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

3.1.3 (26) - 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)
        Msgbox('Temperature is ', temp)
               % Display a message box with the device temperature
```

3.1.3 (27) - Check Connection

Declaration

```
Short Check_Connection()
```

Description

This function checks whether the USB connection to the signal generator is still 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

3.1.3 (28) - Get Firmware Version

Declaration

```
Short GetFirmware()
```

Description

This function returns a numeric value which indicates the internal firmware version of the signal 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()
```

3.1.3 (29) - 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

Applies to SSG-6000 only.

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

3.1.3 (30) - 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.

Applies to SSG-6000 only.

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

3.1.3 (31) - Frequency Sweep – Get Maximum Dwell Time

Declaration

```
Short FSweep_GetMaxDwell()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (32) - Frequency Sweep – Get Minimum Dwell Time

Declaration

```
Short FSweep_GetMinDwell()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (33) - Frequency Sweep – Get Power

Declaration

```
Float FSweep_GetPower()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (34) - Frequency Sweep – Get Start Frequency

Declaration

```
Double FSweep_GetStartFreq()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (35) - Frequency Sweep – Get Stop Frequency

Declaration

```
Double FSweep_GetStopFreq()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (36) - Frequency Sweep – Get Step Size

Declaration

```
Double FSweep_GetStepSize()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (37) - 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 the frequency sweep

Applies to SSG-6000 only.

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 commencing 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

3.1.3 (38) - 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 the frequency sweep is initiated

Applies to SSG-6000 only.

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 the 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

3.1.3 (39) - 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

Applies to SSG-6000 only.

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

3.1.3 (40) - 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.

Applies to SSG-6000 only.

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

3.1.3 (41) - 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.

Applies to SSG-6000 only.

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

3.1.3 (42) - 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.

Applies to SSG-6000 only.

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

3.1.3 (43) - Frequency Sweep – Set Start Frequency

Declaration

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

Description

This function sets the frequency in MHz at which the sweep will start.

Applies to SSG-6000 only.

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

3.1.3 (44) - Frequency Sweep – Set Stop Frequency

Declaration

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

Description

This function sets the frequency in MHz at which the sweep will stop.

Applies to SSG-6000 only.

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

3.1.3 (45) - 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.

Applies to SSG-6000 only.

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

3.1.3 (46) - 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 the frequency sweep

Applies to SSG-6000 only.

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

3.1.3 (47) - Frequency Sweep – Set Trigger Out Mode

Declaration

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

Description

This function specified 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 the frequency sweep is initiated

Applies to SSG-6000 only.

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

3.1.3 (48) - Frequency/Power Hop – Get Direction

Declaration

```
Short Hop_GetDirection()
```

Description

This function returns the direction setting for the current hop sequence:
0 – Ascending from first to last
1 – Descending from last to first
2 – Ascending from first to last, then descending from last to first

Applies to SSG-6000 only.

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

3.1.3 (49) - 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.

Applies to SSG-6000 only.

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

3.1.3 (50) - 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.

Applies to SSG-6000 only.

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

3.1.3 (51) - 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.

Applies to SSG-6000 only.

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

3.1.3 (52) - Frequency/Power Hop – Get Maximum Number of Points

Declaration

```
Short Hop_GetMaxPoints()
```

Description

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

Applies to SSG-6000 only.

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_GetMaxPoints
```

Visual C++

```
Hops = MyPTE1->Hop_GetMaxPoints();
```

Visual C#

```
Hops = MyPTE1.Hop_GetMaxPoints();
```

Matlab

```
Hops = MyPTE1.Hop_GetMaxPoints
```

See Also

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

3.1.3 (53) - 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).

Applies to SSG-6000 only.

Parameters

Data Type	Variable	Description
Short	PointNo	Required. The point number; for example, 1 for the first point in the sequence, 2 for the second.
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

3.1.3 (54) - 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 the hop sequence

Applies to SSG-6000 only.

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

3.1.3 (55) - 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 point is set
2 – Provide a trigger output (Trigger Out = logic 1) as hop is initiated

Applies to SSG-6000 only.

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

3.1.3 (56) - Frequency/Power Hop – Set Direction

Declaration

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

Description

This function sets the direction of the hop sequence:
0 – Ascending from first to last
1 – Descending from last to first
2 – Ascending from first to last, then descending from last to first

Applies to SSG-6000 only.

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

3.1.3 (57) - 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.

Applies to SSG-6000 only.

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

3.1.3 (58) - 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.

Applies to SSG-6000 only.

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

3.1.3 (59) - 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.

Applies to SSG-6000 only.

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

3.1.3 (60) - 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.

Applies to SSG-6000 only.

Parameters

Data Type	Variable	Description
Short	PointNo	Required. The point number; 1 for the first point in the sequence, 2 for the second, up to 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

3.1.3 (61) - 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 the hop sequence

Applies to SSG-6000 only.

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 commencing 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

3.1.3 (62) - 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 the sequence

Applies to SSG-6000 only.

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 commencing the 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

3.1.3 (63) - 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

Applies to SSG-6000 only.

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

3.1.3 (64) - 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.

Applies to SSG-6000 only.

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

3.1.3 (65) - Power Sweep – Get Maximum Dwell Time

Declaration

```
Short PSweep_GetMaxDwell()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (66) - Power Sweep – Get Minimum Dwell Time

Declaration

```
Short PSweep_GetMinDwell()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (67) - Power Sweep – Get Frequency

Declaration

```
Float PSweep_GetFreq()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (68) - Power Sweep – Get Start Power

Declaration

```
Double PSweep_GetStartPower()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (69) - Power Sweep – Get Stop Power

Declaration

```
Double PSweep_GetStopPower()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (70) - Power Sweep – Get Step Size

Declaration

```
Double PSweep_GetStepSize()
```

Description

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

Applies to SSG-6000 only.

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

3.1.3 (71) - 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 the power sweep

Applies to SSG-6000 only.

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 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

3.1.3 (72) - 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 the power sweep is initiated

Applies to SSG-6000 only.

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 the sweep is initialized

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

3.1.3 (73) - 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

Applies to SSG-6000 only.

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

3.1.3 (74) - 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.

Applies to SSG-6000 only.

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

3.1.3 (75) - 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.

Applies to SSG-6000 only.

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

3.1.3 (76) - Power Sweep – Set Frequency

Declaration

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

Description

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

Applies to SSG-6000 only.

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

3.1.3 (77) - Power Sweep – Set Start Power

Declaration

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

Description

This function sets the power in dBm at which the sweep will start.

Applies to SSG-6000 only.

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

3.1.3 (78) - Power Sweep – Set Stop Power

Declaration

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

Description

This function sets the power in dBm at which the sweep will stop.

Applies to SSG-6000 only.

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

3.1.3 (79) - 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.

Applies to SSG-6000 only.

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

3.1.3 (80) - 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 the power sweep

Applies to SSG-6000 only.

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 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

3.1.3 (81) - 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 the power sweep is initiated

Applies to SSG-6000 only.

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 the 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

3.2 - Operating in a Linux Environment

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 Appendix C of this document. The example uses the libhid and libusb libraries to interface with the signal generator as a USB HID (Human Interface Device).

3.2.1 - 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.

#	Description	Command Code (Byte 0)
a	Get Device Model Name	40
b	Get Device Serial Number	41
c	Set Frequency & Power (SSG-4000 Series) Set Frequency & Power (SSG-6000)	103
d	Set Frequency (SSG-4000 Series) Set Frequency (SSG-6000)	101
e	Set Power	102
f	Set RF Power On/Off	104
g	Set Noise/Spur Mode	106
h	Get Generator Output Status (SSG-4000 Series) Get Generator Output Status (SSG-6000)	105
i	Get Generator Minimum Frequency	42
j	Get Generator Maximum Frequency (SSG-4000 Series) Get Generator Maximum Frequency (SSG-6000)	43
k	Get Generator Step Size Spec	44
l	Get Generator Minimum Power Spec	45
m	Get Generator Maximum Power Spec	46

3.2.2 - Detailed Description of Commands

3.2.2 (1) - Get Device Model Name

Description

This function determines the Mini-Circuits part number of the connected signal generator.

Send code 40 in BYTE0 of the transmit array. BYTE1 through to BYTE63 are don't care bytes and can be any value.

The model name is represented as a series of ASCII characters in the returned array, starting from BYTE1. The final ASCII character is contained in the byte immediately proceeding the first zero value byte. All subsequent bytes up to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	**Byte 0**
Description	Code
Value	40

Returned Array

Byte	**Byte 0**	**Byte 1**	**Byte 2**	**…**	**Byte (N-1)**	**Byte N**
Description	Code	First Char	Second Char	…	Last Char	End Marker
Value	40	ASCII	ASCII	…	ASCII	0

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

3.2.2 (2) - Get Device Serial Number

Description

This function determines the serial number of the connected signal generator.

Send code 41 in BYTE0 of the transmit array. BYTE1 through to BYTE63 are "don't care" bytes and can be any value.

The serial number is represented as a series of ASCII characters in the returned array, starting from BYTE1. The final ASCII character is contained in the byte immediately proceeding the first zero value byte. All subsequent bytes up to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	Byte 0
Description	Code
Value	41

Returned Array

Byte	Byte 0	Byte 1	Byte 2	…	Byte (N-1)	Byte N
Description	Code	First Char	Second Char	…	Last Char	End Marker
Value	41	ASCII	ASCII	…	ASCII	0

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

3.2.3 - Set Frequency and Power (SSG-4000 Series only)

Description

This function sets the RF ouput frequency and power level of the SSG-4000 Series signal generators and enables or disables the “trigger out” function.

The transmit array is made up of the following bytes:

- BYTE0
    - 103 (code for Set Frequency and Power)
- BYTE1 to BYTE4
    - Frequency in Hz broken up into 4 bytes, with MSB in BYTE1 and LSB in BYTE4
    - The value for each byte is calculated as:
        - BYTE1 = INTEGER VALUE (FREQUENCY / 256 ^ 3)
        - REMAINDER1 = FREQUENCY - BYTE1 * (256 ^ 3)
        - BYTE2 = INTEGER VALUE (REMAINDER1 / 256 ^ 2)
        - REMAINDER2 = REMAINDER1 - BYTE2 * (256 ^ 2)
        - BYTE3 = INTEGER VALUE (REMAINDER2 / 256)
        - BYTE4 = REMAINDER2 - BYTE3 * 256
- BYTE5
    - 1 (to set a negative power value) or 0 (to set a positive power value)
- BYTE6 to BYTE7
    - Absolute power in dBm multiplied by 100
    - The value is split into MSB (BYTE6) and LSB (BYTE7)
    - BYTE6 = INTEGER VALUE ((ABSOLUTE POWER * 100) / 256)
    - BYTE7 = (ABSOLUTE POWER * 100) - (BYTE6 * 256)
- BYTE8
    - 1 (to enable Trigger Out) or (0 to disable Trigger Out)
- BYTE9 to BYTE63
    - Can be any value (“don’t care” bytes)

The returned array contains 103 in BYTE0 (the code for “Set Frequency and Power”), BYTE1 to BYTE63 are “don’t care” bytes and could be any value.

Transmit Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq (LSB)	Power Sign
Value	103	Byte	Byte	Byte	Byte	0 or 1

Byte	Byte 6	Byte 7	Byte 8
Description	Power (MSB)	Power (LSB)	Trigger Out
Value	Byte	Byte	0 or 1

Returned Array

Byte	Byte 0
Description	Code
Value	103

Example

To set the generator output to 3501.56MHz with a power level of -5.5dBm and enable the Trigger Out:

1) Frequency = 3,501,560,000 Hz

```
BYTE1          = INTEGER (3,501,560,000 / 256 ^ 3)
               = INTEGER (208.71)
               = 208

REMAINDER1     = FREQUENCY - BYTE1 * (256 ^ 3)
               = 3,501,560,000 - 208 * (256 ^ 3)
               = 11,899,072

BYTE2          = INTEGER (REMAINDER1 / 256 ^ 2)
               = INTEGER (11,899,072 / 256 ^ 2)
               = 181

REMAINDER2     = REMAINDER1 - BYTE2 * (256 ^ 2)
               = 11,899,072 - 181 * (256 ^ 2)
               = 37,056

BYTE3          = INTEGER (REMAINDER2 / 256)
               = INTEGER (37,056 / 256)
               = 144

BYTE4          = REMAINDER2 - BYTE3 * 256
               = 37,056 - 144 * 256
               = 192
```

2) Power

```
BYTE5          = 1 (ignore the sign in the below calculations)

BYTE6          = INTEGER ((5.5 * 100) / 256)
               = INTEGER (2.15)
               = 2

BYTE7          = (5.5 *100) - (2 * 256)
               = 38
```

3) Trigger Out

```
BYTE8    = 1 to enable the trigger out
```

The complete transmit array is therefore:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq (LSB)	Power Sign
Value	103	208	181	144	192	1

Byte	Byte 6	Byte 7	Byte 8
Description	Power (MSB)	Power (LSB)	Trigger Out
Value	2	38	1

See Also

Set Frequency and Power (SSG-6000 only)
Set Frequency (SSG-4000 Series Only)
Set Power
Set RF Power On/Off
Get Generator Output Status

3.2.3 (1) - Set Frequency and Power (SSG-6000 only)

Description

This function sets the RF ouput frequency and power level of the SSG-6000 signal generator and enables or disables the "trigger out" function.

The transmit array is made up of the following bytes:

- BYTE0
  - 103 (code for Set Frequency and Power)
- BYTE1 to BYTE5
  - Frequency in Hz broken up into 5 bytes, with MSB in BYTE1 and LSB in BYTE5
  - The value for each byte is calculated as:
    - BYTE1 = INTEGER VALUE (FREQUENCY / 256 ^ 4)
    - REMAINDER1 = FREQUENCY - BYTE1 * (256 ^ 4)
    - BYTE2 = INTEGER VALUE (REMAINDER1 / 256 ^ 3)
    - REMAINDER2 = REMAINDER1 - BYTE2 * (256 ^ 3)
    - BYTE3 = INTEGER VALUE (REMAINDER2 / 256 ^ 2)
    - REMAINDER3 = REMAINDER2 - BYTE3 * (256 ^ 2)
    - BYTE4 = INTEGER VALUE (REMAINDER3 / 256)
    - BYTE5 = INTEGER VALUE (REMAINDER3 - BYTE4 * 256)
- BYTE6
  - 1 (to set a negative power value) or 0 (to set a positive power value)
- BYTE7 to BYTE8
  - Absolute power in dBm multiplied by 100
  - The value is split into MSB (BYTE7) and LSB (BYTE8)
  - BYTE7 = INTEGER VALUE ((ABSOLUTE POWER * 100) / 256)
  - BYTE8 = (ABSOLUTE POWER * 100) - (BYTE6 * 256)
- BYTE9
  - 1 (to enable Trigger Out) or (0 to disable Trigger Out)
- BYTE10 to BYTE63
  - Can be any value ("don't care" bytes)

The returned array contains 103 in BYTE0 (the code for "Set Frequency and Power"), BYTE1 to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)	Power Sign
Value	103	Byte	Byte	Byte	Byte	Byte	0 or 1

Byte	Byte 7	Byte 8	Byte 9
Description	Power (MSB)	Power (LSB)	Trigger Out
Value	Byte	Byte	0 or 1

Returned Array

Byte	Byte 0
Description	Code
Value	103

Example

To set the generator output to 5501.56MHz with a power level of +4.5dBm and enable the Trigger Out:

4) Frequency = 5,501,560,000 Hz

BYTE1	= INTEGER (5,501,560,000 / 256 ^ 4) = INTEGER (1.2809) = 1
REMAINDER1	= 5,501,560,000 - 1 * (256 ^ 4) = 1,206,592,704
BYTE2	= INTEGER (1,206,592,704 / 256 ^ 3) = 71
REMAINDER2	= 1,206,592,704 - 71 * (256 ^ 3) = 60,196
BYTE3	= INTEGER (60,196 / 256 ^ 2) = 235
REMAINDER3	= 1,206,592,704 - 71 * (256 ^ 2) = 9,408
BYTE4	= INTEGER (9,408 / 256) = 36
BYTE5	= INTEGER (9,408 - (36 * 256)) = 192

5) Power

BYTE6	= 0 (ignore the sign in the below calculations)
BYTE7	= INTEGER ((POWER * 100) / 256) = INTEGER ((4.5 * 100) / 256) = INTEGER (1.76) = 1
BYTE8	= (POWER * 100) - (BYTE7 * 256) = (4.5 *100) - (1 * 256) = 194

6) Trigger Out

BYTE9 = 1 to enable the trigger out

The complete transmit array is therefore:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)
Value	103	1	71	235	36	192

Byte	Byte 6	Byte 7	Byte 8	Byte 9
Description	Power Sign	Power (MSB)	Power (LSB)	Trigger Out
Value	0	1	194	1

See Also

Set Frequency and Power (SSG-4000 Series only)
Set Frequency (SSG-6000 Only)
Set Power
Set RF Power On/Off
Get Generator Output Status

3.2.3 (2) - Set Frequency (SSG-4000 Series Only)

Description

This function sets the RF ouput frequency of the SSG-4000 series signal generators and enables or disables the "trigger out" function. It does not affect the current power setting.

The transmit array is made up of the following bytes:

- BYTE0
  - 101(code for Set Frequency)
- BYTE1 to BYTE4
  - Frequency in Hz broken up into 4 bytes, with MSB in BYTE1 and LSB in BYTE4
  - The value for each byte is calculated as:
    - BYTE1 = INTEGER VALUE (FREQUENCY / 256 ^ 3)
    - REMAINDER1 = FREQUENCY - BYTE1 * (256 ^ 3)
    - BYTE2 = INTEGER VALUE (REMAINDER1 / 256 ^ 2)
    - REMAINDER2 = REMAINDER1 - BYTE2 * (256 ^ 2)
    - BYTE3 = INTEGER VALUE (REMAINDER2 / 256)
    - BYTE4 = REMAINDER2 - BYTE3 * 256
- BYTE5
  - 1 (to enable Trigger Out) or (0 to disable Trigger Out)
- BYTE6 to BYTE63
  - Can be any value ("don't care" bytes)

The returned array contains 101 in BYTE0 (the code for "Set Frequency"), BYTE1 to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq (LSB)	Trigger Out
Value	101	Byte	Byte	Byte	Byte	0 or 1

Returned Array

Byte	Byte 0
Description	Code
Value	101

Example

To set the generator output to 1000.55MHz and enable the Trigger Out:

1) Frequency = 1,000,550,000 Hz

BYTE1 = INTEGER (1,000,550,000 / 256 ^ 3)
= INTEGER (59.64)
= 59

REMAINDER1 = FREQUENCY - BYTE1 * (256 ^ 3)
= 1,000,550,000 - 59 * (256 ^ 3)
= 10,694,256

BYTE2 = INTEGER (REMAINDER1 / 256 ^ 2)
= INTEGER (10,694,256 / 256 ^ 2)
= 163

REMAINDER2 = REMAINDER1 - BYTE2 * (256 ^ 2)
= 10,694,256 - 163 * (256 ^ 2)
= 11,888

BYTE3 = INTEGER (REMAINDER2 / 256)
= INTEGER (11,888 / 256)
= 46

BYTE4 = REMAINDER2 - BYTE3 * 256
= 11,888 - 46 * 256
= 112

2) Trigger Out

BYTE5 = 1 to enable the trigger out

3) The complete transmit array is therefore:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq (LSB)	Trigger Out
Value	101	59	163	46	112	1

See Also

Set Frequency and Power (SSG-4000 Series only)
Set Frequency (SSG-6000 Only)
Set Power
Set RF Power On/Off
Get Generator Output Status

3.2.3 (3) - Set Frequency (SSG-6000 Only)

Description

This function sets the ouput frequency of the SSG-6000 signal generator and enables or disables the "trigger out" function. It does not affect the current power setting.

The transmit array is made up of the following bytes:

- BYTE0
  - 101(code for Set Frequency)
- BYTE1 to BYTE5
  - Frequency in Hz broken up into 5 bytes, with MSB in BYTE1 and LSB in BYTE5
  - The value for each byte is calculated as:
    - BYTE1 = INTEGER VALUE (FREQUENCY / 256 ^ 4)
    - REMAINDER1 = FREQUENCY - BYTE1 * (256 ^ 4)
    - BYTE2 = INTEGER VALUE (REMAINDER1 / 256 ^ 3)
    - REMAINDER2 = REMAINDER1 - BYTE2 * (256 ^ 3)
    - BYTE3 = INTEGER VALUE (REMAINDER2 / 256 ^ 2)
    - REMAINDER3 = REMAINDER2 - BYTE3 * (256 ^ 2)
    - BYTE4 = INTEGER VALUE (REMAINDER3 / 256)
    - BYTE5 = INTEGER VALUE (REMAINDER3 - BYTE4 * 256)
- BYTE6
  - 1 (to enable Trigger Out) or (0 to disable Trigger Out)
- BYTE7 to BYTE63
  - Can be any value ("don't care" bytes)

The returned array contains 101 in BYTE0 (the code for "Set Frequency"), BYTE1 to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)	Trigger Out
Value	101	Byte	Byte	Byte	Byte	Byte	0 or 1

Returned Array

Byte	**Byte 0**
Description	Code
Value	101

Example

To set the generator output to 4100.55MHz and enable the Trigger Out:

4) Frequency = 4,100,550,000 Hz

BYTE1 = INTEGER (4,100,550,000 / 256 ^ 4)
= INTEGER (0.9547)
= 0

REMAINDER1 = 4,100,550,000 - 0 * (256 ^ 4)
= 4,100,550,000

BYTE2 = INTEGER (4,100,550,000 / 256 ^ 3)
= 244

REMAINDER2 = 4,100,550,000 - 244 * (256 ^ 3)
= 6,909,296

BYTE3 = INTEGER (6,909,296 / 256 ^ 2)
= 105

REMAINDER3 = 6,909,296 - 105 * (256 ^ 2)
= 28,016

BYTE4 = INTEGER (28,016 / 256)
= 109

BYTE5 = INTEGER (28,016 - (109 * 256))
= 112

5) Trigger Out

BYTE6 = 1 to enable the trigger out

6) The complete transmit array is therefore:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Description	Code	Freq 0 (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)	Trigger Out
Value	101	0	244	105	109	112	1

See Also

Set Frequency and Power (SSG-6000 only)
Set Frequency (SSG-4000 Series Only)
Set Power
Set RF Power On/Off
Get Generator Output Status

3.2.3 (4) - Set Power

Description

This function sets the RF ouput power of the signal generator and enables or disables the "trigger out" function. It does not affect the current frequency setting.

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)

The returned array contains 102 in BYTE0 (the code for "Set Frequency and Power"), BYTE1 to BYTE63 are "don't care" bytes and could be any value.

Transmit Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Power Sign	Power (MSB)	Power (LSB)	Trigger Out
Value	102	0 or 1	Byte	Byte	0 or 1

Returned Array

Byte	Byte 0
Description	Code
Value	102

Example

To set the generator output power to -5.5dBm and enable the Trigger Out:

1) Power

BYTE1 = 1 since the power is negative (ignore the sign in the below calculations)

BYTE2 = INTEGER ((POWER * 100) / 256)
= INTEGER ((5.5 * 100) / 256)
= INTEGER (2.15)
= 2

BYTE3 = (POWER * 100) - (BYTE2 * 256)
= (5.5 *100) - (2 * 256)
= 38

2) Trigger Out

BYTE4 = 1 to enable the trigger out

The complete transmit array is therefore:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Power Sign	Power (MSB)	Power (LSB)	Trigger Out
Value	102	1	2	38	1

See Also

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

3.2.3 (5) - 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	Byte 0	Byte 1
Description	Code	Power On/Off
Value	104	1 or 0

Returned Array

Byte	Byte 0
Description	Code
Value	104

Example

To enable the RF output, send the following transmit array:

Byte	Byte 0	Byte 1
Description	Code	Power On/Off
Value	104	1

To disable the RF output, send the following transmit array:

Byte	Byte 0	Byte 1
Description	Code	Power On/Off
Value	104	0

See Also

Set Frequency and Power (SSG-4000 Series only)
Set Frequency and Power (SSG-6000 only)
Set Power
Get Generator Output Status (SSG-4000 Series)
Get Generator Output Status (SSG-6000 Only)

3.2.3 (6) - Set Noise/Spur Mode

Description

This function applies to SSG-4000LH and SSG-4000HP only. It sets the generator in either “Low Noise” mode (for best phase noise performance) or “Low Spur” mode (for best spurious performance). The generator defaults to “Low Noise” mode.

Send code 106 in BYTE0 of the transmit array with BYTE1 as 1 to enable “Low Spur” mode or 0 to enable “Low Noise” mode. BYTE2 to BYTE63 are “don’t care” bytes and can be any value.

The returned array contains 106 in BYTE0. BYTE1 to BYTE63 are “don’t care” bytes and could be any value.

Transmit Array

Byte	Byte 0	Byte 1
Description	Code	Mode
Value	106	1 or 0

Returned Array

Byte	Byte 0
Description	Code
Value	106

Example

To set “Low Spur” mode, send the following transmit array:

Byte	Byte 0	Byte 1
Description	Code	Mode
Value	106	1

To set “Low Noise” mode, send the following transmit array:

Byte	Byte 0	Byte 1
Description	Code	Mode
Value	106	0

3.2.3 (7) - Get Generator Output Status (SSG-4000 Series)

Description

This function returns the current output status of the SSG-4000 series signal generators. 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

Send code 105 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 105 (code for Get Generator Status)
- BYTE1
  - RF output status (1 if enabled or 0 if disabled)
- BYTE2
  - Lock status (1 if frequency is locked or 0 if not locked)
- BYTE3 to BYTE6
  - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
  - FREQUENCY = (256 ^ 3) * BYTE3 + (256 ^ 2) * BYTE4 + 256 * BYTE5 + BYTE6
- BYTE7
  - Output power sign (1 if power value is negative or 0 if positive)
- BYTE8 to BYTE9
  - Power setting in dBm (absolute value), split into MSB (BYTE8) and LSB (BYTE9)
  - Absolute power is calculated as:
    P = (256 * BYTE8 + BYTE9) / 100
- BYTE10
  - High power request warning (UnLevel High)
  - 1 if the user requested a higher power than the generator can achieve or 0 if the output power is within the correct range
- BYTE11
  - Low power request warning (UnLevel Low)
  - 1 if the user requested a lower power than the generator can achieve or 0 if the output power is within the correct range
- BYTE12 to BYTE63
  - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	105

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	RF On/Off	Lock Status	Freq (MSB)	Freq 1	Freq 2
Value	105	1 or 0	1 or 0	Byte	Byte	Byte

Byte	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
Description	Freq (LSB)	Power + or -	Power (MSB)	Power (LSB)	UnLevel High	UnLevel Low
Value	Byte	0 or 1	Byte	Byte	0 or 1	0 or 1

Example

The following array would be returned if the generator was set with the output enabled at 751.25MHz, +5.5dBm):

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	RF On/Off	Lock Status	Freq (MSB)	Freq 1	Freq 2
Value	105	1	1	44	199	42

Byte	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
Description	Freq (LSB)	Power + or -	Power (MSB)	Power (LSB)	UnLevel High	UnLevel Low
Value	80	0	2	38	0	0

The returned array is broken down as follows:

- RF output is enabled (BYTE1 = 1)
- Generator is locked (BYTE2 = 1)
- Frequency
  = (256 ^ 3) * BYTE3 + (256 ^ 2) * BYTE4 + 256 * BYTE5 + BYTE6
  = (256 ^ 3) * 44 + (256 ^ 2) * 199 + 256 * 42 + 80
  = 751,250,000 Hz
  = 751.25 MHz
- Absolute power
  = (256 * BYTE2 + BYTE3) / 100
  = (256 * 2 + 38) / 100
  = 5.5dBm
- Real power
  = +5.5dBm (since BYTE7 = 1)
- Power has settled to user defined level (BYTE10 = 0 and BYTE11 = 0)

See Also

Set Frequency and Power (SSG-4000 Series only)
Set Frequency (SSG-4000 Series Only)
Set Power
Set RF Power On/Off

3.2.3 (8) - Get Generator Output Status (SSG-6000 Only)

Description

This function returns the current output status of the SSG-6000 signal generator. 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

Send code 105 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 105 (code for Get Generator Status)
- BYTE1
  - RF output status (1 if enabled or 0 if disabled)
- BYTE2
  - Lock status (1 if frequency is locked or 0 if not locked)
- BYTE3 to BYTE7
  - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
  - FREQUENCY = (256 ^ 4) * BYTE3 + (256 ^ 3) * BYTE4 + (256 ^ 2) * BYTE5 + 256 * BYTE6 + BYTE7
- BYTE8
  - Output power sign (1 if power value is negative or 0 if positive)
- BYTE9 to BYTE10
  - Power setting in dBm (absolute value), split into MSB (BYTE8) and LSB (BYTE9)
  - Absolute power is calculated as:
    P = (256 * BYTE9 + BYTE10) / 100
- BYTE11
  - High power request warning (UnLevel High)
  - 1 if the user requested a higher power than the generator can achieve or 0 if the output power is within the correct range
- BYTE12
  - Low power request warning (UnLevel Low)
  - 1 if the user requested a lower power than the generator can achieve or 0 if the output power is within the correct range
- BYTE13 to BYTE63
  - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	105

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Description	Code	RF On/Off	Lock Status	Freq (MSB)	Freq 1	Freq 2	Freq 3
Value	105	1 or 0	1 or 0	Byte	Byte	Byte	Byte

Byte	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
Description	Freq (LSB)	Power + or -	Power (MSB)	Power (LSB)	UnLevel High	UnLevel Low
Value	Byte	0 or 1	Byte	Byte	0 or 1	0 or 1

Example

The following array would be returned if the generator was set with the output enabled at 4980.50MHz, +5.5dBm):

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Description	Code	RF On/Off	Lock Status	Freq (MSB)	Freq 1	Freq 2	Freq 3
Value	105	1	1	1	40	220	102

Byte	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
Description	Freq (LSB)	Power + or -	Power (MSB)	Power (LSB)	UnLevel High	UnLevel Low
Value	32	0	2	38	0	0

The returned array is broken down as follows:

- RF output is enabled (BYTE1 = 1)
- Generator is locked (BYTE2 = 1)
- Frequency
  = (256 ^ 4) * BYTE3 + (256 ^ 3) * BYTE4 + (256 ^ 2) * BYTE5
  + 256 * BYTE6 + BYTE7
  = (256 ^ 4) * 1 + (256 ^ 3) * 40 + (256 ^ 2) * 220 + 256 * 102 + 32
  = 751,250,000 Hz
  = 751.25 MHz
- Absolute power
  = (256 * BYTE2 + BYTE3) / 100
  = (256 * 2 + 38) / 100
  = 5.5dBm
- Real power
  = +5.5dBm (since BYTE7 = 1)
- Power has settled to user defined level (BYTE10 = 0 and BYTE11 = 0)

See Also

Set Frequency and Power (SSG-6000 only)
Set Frequency (SSG-6000 Only)
Set Power
Set RF Power On/Off

3.2.3 (9) - Get Generator Minimum Frequency

Description

This function reports the signal generator minimum frequency specification in Hz.

Send code 42 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 42 (code for Get Generator Minimum Frequency)
- BYTE1 to BYTE4
  - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
  - FREQUENCY = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
- BYTE5 to BYTE63
  - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	42

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	42	Byte	Byte	Byte	Byte

Example

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

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	42	14	230	178	128

The minimum frequency spec can be calculated as follows:

```
Min Frequency  = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
               = (256 ^ 3) * 14 + (256 ^ 2) * 230 + 256 * 178 + 128
               = 250,000,000 Hz
               = 250 MHz
```

See Also

Get Generator Maximum Frequency (SSG-4000 Series Only)
Get Generator Maximum Frequency (SSG-6000 Only)
Get Generator Step Size
Get Generator Minimum Power
Get Generator Maximum Power

3.2.3 (10) - Get Generator Maximum Frequency (SSG-4000 Series Only)

Description

This function reports the maximum frequency specification in Hz for the SSG-4000 series signal generators.

Send code 43 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
    - 43 (code for Get Generator Maximum Frequency)
- BYTE1 to BYTE4
    - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
    - FREQUENCY = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
- BYTE5 to BYTE63
    - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	43

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	43	Byte	Byte	Byte	Byte

Example

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

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	43	238	107	40	0

The maximum frequency spec can be calculated as follows:

Max Frequency = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
= (256 ^ 3) * 238 + (256 ^ 2) * 107 + 256 * 40 + 0
= 4,000,000,000 Hz
= 4,000 MHz

See Also

Get Generator Maximum Frequency (SSG-6000 Only)
Get Generator Minimum Frequency
Get Generator Step Size
Get Generator Minimum Power
Get Generator Maximum Power

3.2.3 (11) - Get Generator Maximum Frequency (SSG-6000 Only)

Description

This function reports the maximum frequency specification in Hz for the SSG-6000 signal generators.

Send code 43 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 43 (code for Get Generator Maximum Frequency)
- BYTE1 to BYTE5
  - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
  - FREQUENCY = (256 ^ 4) * BYTE1 + (256 ^ 3) * BYTE2 + (256 ^ 2) * BYTE3 + 256 * BYTE4 + BYTE5
- BYTE6 to BYTE63
  - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	43

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)
Value	43	Byte	Byte	Byte	Byte	Byte

Example

The following array would be returned for SSG-6000:

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq 3	Freq (LSB)
Value	43	1	101	160	188	0

The maximum frequency spec can be calculated as follows:

Max Frequency = (256 ^ 4) * BYTE1 + (256 ^ 3) * BYTE2 + (256 ^ 2) * BYTE3
+ 256 * BYTE4 + BYTE5
= (256 ^ 4) * 1 + (256 ^ 3) * 101 + (256 ^ 2) * 160 + 256 * 188 + 0
= 6,000,000,000 Hz
= 6,000 MHz

See Also

Get Generator Maximum Frequency (SSG-4000 Series Only)
Get Generator Minimum Frequency
Get Generator Step Size
Get Generator Minimum Power
Get Generator Maximum Power

3.2.3 (12) - Get Generator Step Size

Description

This function reports the signal generator's step size in Hz.

Send code 44 in BYTE0 of the transmit array. BYTE2 to BYTE63 are "don't care" bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 44 (code for Get Generator Step Size)
- BYTE1 to BYTE4
  - Frequency in Hz, broken up into 4 bytes with MSB in BYTE3 and LSB in BYTE6
  - FREQUENCY = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
- BYTE5 to BYTE63
  - Could be any value ("don't care" bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	44

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	44	Byte	Byte	Byte	Byte

Example

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

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Description	Code	Freq (MSB)	Freq 1	Freq 2	Freq (LSB)
Value	43	0	0	19	136

The step size frequency spec can be calculated as follows:

Step Size = (256 ^ 3) * BYTE1 + (256 ^ 2) * BYTE2 + 256 * BYTE3 + BYTE4
= (256 ^ 3) * 0 + (256 ^ 2) * 0 + 256 * 19 + 136
= 5,000 Hz
= 5 KHz

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency (SSG-4000 Series Only)
Get Generator Maximum Frequency (SSG-6000 Only)
Get Generator Minimum Power
Get Generator Maximum Power

3.2.3 (13) - Get Generator Minimum Power

Description

This function reports the minimum output power in dBm that the generator is capable of providing.

Send code 45 in BYTE0 of the transmit array. BYTE2 to BYTE63 are “don’t care” bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 45 (code for Get Generator Minimum Power)
- BYTE1
  - Output power sign (1 if power value is negative or 0 if positive)
- BYTE2 to BYTE3
  - Absolute power in dBm, split into MSB (BYTE2) and LSB (BYTE3)
  - Absolute power is calculated as:
    P = (256 * BYTE2 + BYTE3) / 100
- BYTE4 to BYTE63
  - Could be any value (“don’t care” bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	45

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3
Description	Code	Power + or -	Power (MSB)	Power (LSB)
Value	45	0 or 1	Byte	Byte

Example

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

Byte	Byte 0	Byte 1	Byte 2	Byte 3
Description	Code	Power + or -	Power (MSB)	Power (LSB)
Value	45	1	19	136

The minimum power spec can be calculated as follows:

Absolute Power = (256 * BYTE2 + BYTE3) / 100
= (256 * 19 + 136) / 100
= 50 dBm

Real power = -50 dBm (since BYTE1 = 1)

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency (SSG-4000 Series Only)
Get Generator Maximum Frequency (SSG-6000 Only)
Get Generator Step Size
Get Generator Maximum Power

3.2.3 (14) - Get Generator Maximum Power

Description

This function reports the maximum output power in dBm that the generator is capable of providing.

Send code 46 in BYTE0 of the transmit array. BYTE2 to BYTE63 are “don’t care” bytes and can be any value.

The returned array is made up of the following bytes:

- BYTE0
  - 46 (code for Get Generator Maximum Power)
- BYTE1
  - Output power sign (1 if power value is negative or 0 if positive)
- BYTE2 to BYTE3
  - Absolute power in dBm, split into MSB (BYTE2) and LSB (BYTE3)
  - Absolute power is calculated as:
    P = (256 * BYTE2 + BYTE3) / 100
- BYTE4 to BYTE63
  - Could be any value (“don’t care” bytes)

Transmit Array

Byte	Byte 0
Description	Code
Value	46

Returned Array

Byte	Byte 0	Byte 1	Byte 2	Byte 3
Description	Code	Power + or -	Power (MSB)	Power (LSB)
Value	46	0 or 1	Byte	Byte

Example

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

Byte	Byte 0	Byte 1	Byte 2	Byte 3
Description	Code	Power + or -	Power (MSB)	Power (LSB)
Value	46	0	7	208

The maximum power spec can be calculated as follows:

Absolute Power = (256 * BYTE2 + BYTE3) / 100
= (256 * 7 + 208) / 100
= 20 dBm

Max Power = +20dBm (since BYTE1 = 1)

See Also

Get Generator Minimum Frequency
Get Generator Maximum Frequency (SSG-4000 Series Only)
Get Generator Maximum Frequency (SSG-6000 Only)
Get Generator Step Size
Get Generator Minimum Power