Mini-Circuits®

Programming Manual

For

USB Synthesized Signal Generator Series

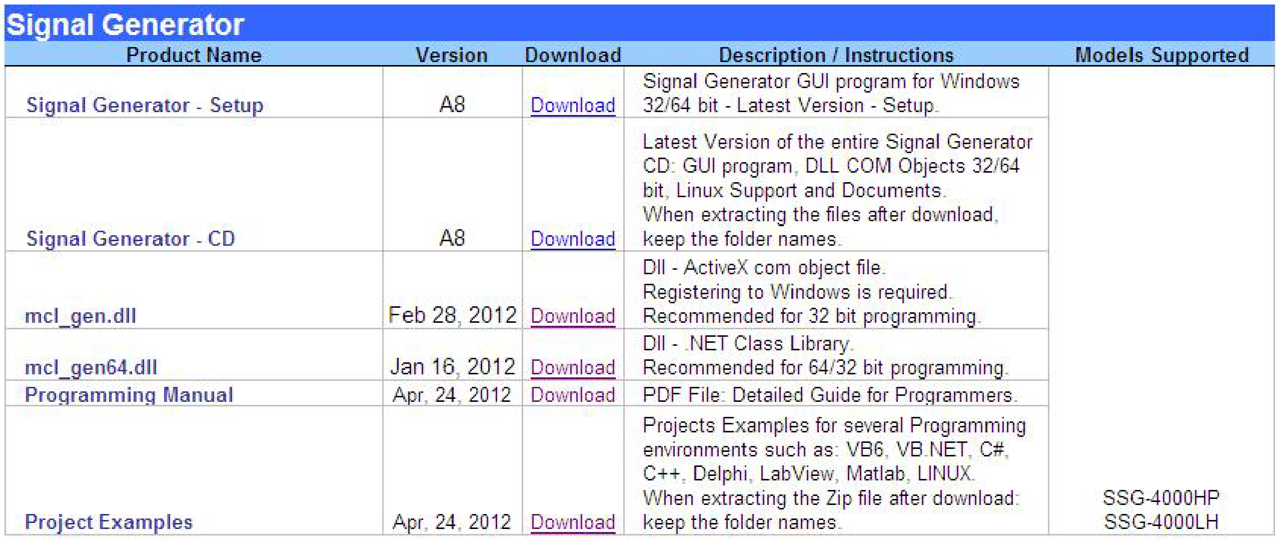
Contents

1- Overview

This programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' USB Synthesized Signal Generators.

Mini-Circuits offers support for USB Portable Test Equipment (PTE) in Windows® and Linux® Operating Systems, in a variety of programming environments including third-party applications such as LabVIEW® and MATLAB® through .NET assembly and ActiveX® Controls to write your own customized control applications.

Mini-Circuits' CD package Includes: GUI program installation, DLL Objects 32/64 bit, Linux Support, project examples for 3RD party software and Documents. The latest CD version is available for download at http://www.minicircuits.com/support/software_download.html , see Figure 1.



Signal Generator

Product Name	Version	Download	Description / Instructions	Models Supported
Signal Generator - Setup	A8	Download	Signal Generator GUI program for Windows 32/64 bit - Latest Version - Setup.	SSG-4000HP SSG-4000LH
Signal Generator - CD	A8	Download	Latest Version of the entire Signal Generator CD: GUI program, DLL COM Objects 32/64 bit, Linux Support and Documents. When extracting the files after download, keep the folder names.	
mcl_gen.dll	Feb 28, 2012	Download	Dll - ActiveX com object file. Registering to Windows is required. Recommended for 32 bit programming.	
mcl_gen64.dll	Jan 16, 2012	Download	Dll - .NET Class Library. Recommended for 64/32 bit programming.	
Programming Manual	Apr, 24, 2012	Download	PDF File: Detailed Guide for Programmers.	
Project Examples	Apr, 24, 2012	Download	Projects Examples for several Programming environments such as: VB6, VB.NET, C#, C++, Delphi, LabView, Matlab, LINUX. When extracting the Zip file after download: keep the folder names.	

Figure 1 – Download Screen

2 - Operating in a Windows® Environment 32/64Bits OS with USB HID Support

The DLL Object (Dynamic Link Library) - Concept:

Dynamic **L**ink **L**ibrary is Microsoft's implementation of the shared library concept in the Microsoft Windows® environment.

DLLs provide a mechanism for shared code and data, allowing a developer of shared code/data to upgrade functionality without requiring applications to be re-linked or recompiled.

Mini-Circuits' CD package provides DLL Objects in order to allow your own Software Application to interface with the functions of the Mini-Circuits' USB Portable Test Equipment hardware, see Figure 2.

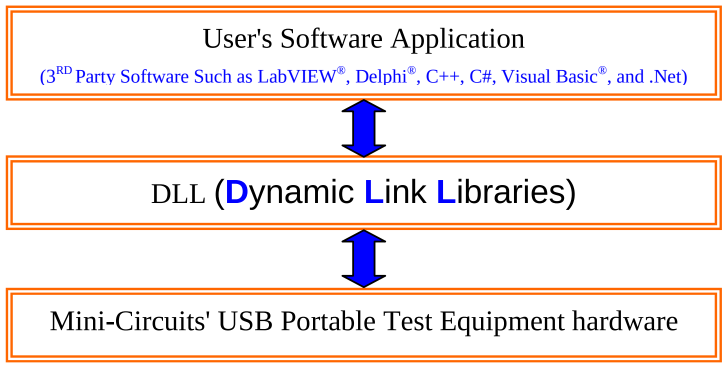

User's Software Application
(3RD Party Software Such as LabVIEW®, Delphi®, C++, C#, Visual Basic®, and .Net)
DLL (Dynamic Link Libraries)
Mini-Circuits' USB Portable Test Equipment hardware

Figure 2 – DLL Interface

<u>Mini-Circuits' provides two DLLs files:</u>

1. ActiveX® com object - *mcl_Gen.dll* → Click to download http://www.minicircuits.com/support/software_download.html
   ActiveX® com object can be used in any programming environment that supports ActiveX® objects - third party COM (Component Object Model) compliant application. The ActiveX® DLL should be registered using RegSvr32 (see pages 5 and 6 - Register an ActiveX® DLL).

2. .NET Class Library - *mcl_Gen64.dll*→ Click to download http://www.minicircuits.com/support/software_download.html
   .NET object – a logical unit of functionality that runs under the control of the .NET

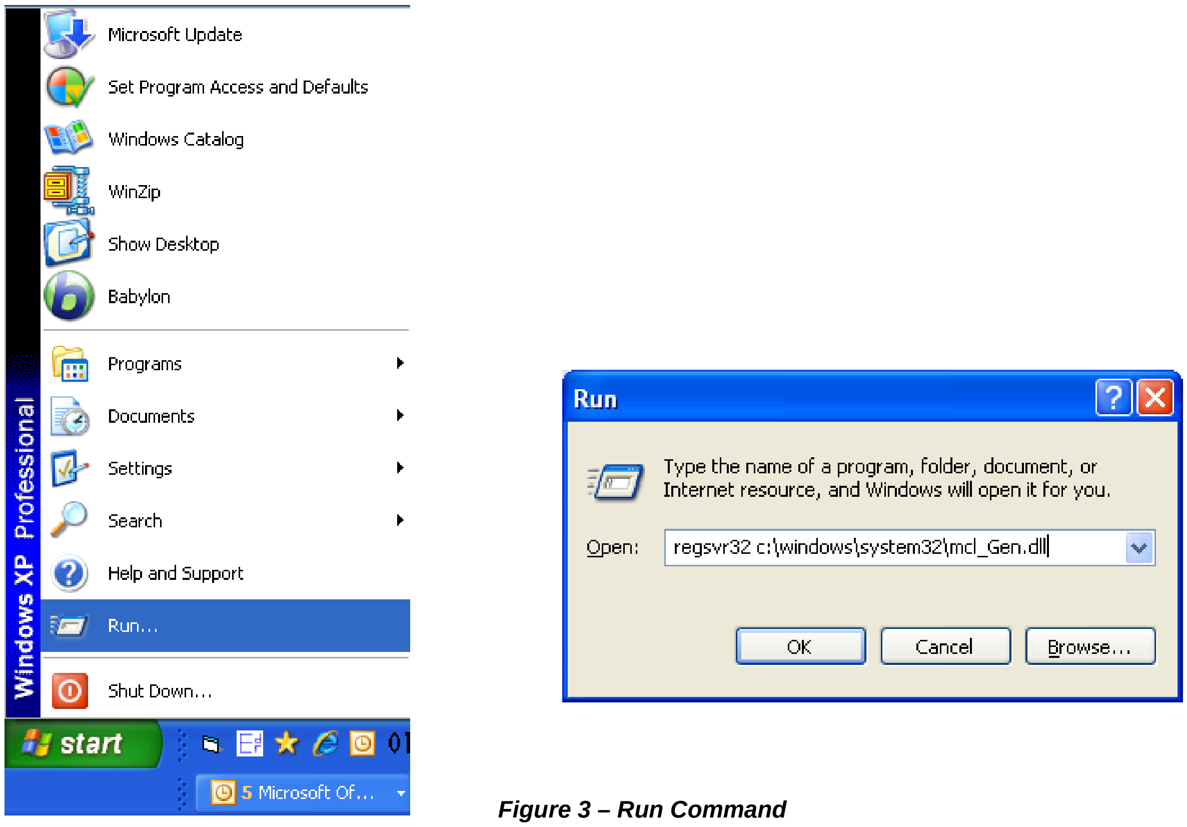
2.1 - Software supported by ActiveX® and .NET Class Library

mcl_Gen.dll - ActiveX® com object	mcl_Gen.dll - .NET Class Library
Instructions ▪ For **32bit** Windows OS, copy mcl_pm.dll to windows\system32 folder ▪ For **64bit** Windows OS, copy mcl_pm.dll to windows\SysWOW64 folder ▪ Register the DLL, see instructions below	**Instructions** ▪ For **32bit** Windows OS copy mcl_pm64.dll to windows\system32 folder ▪ For **64bit** Windows OS copy mcl_pm64.dll to windows\SysWOW64 folder ▪ DLL Registry is not required
Visual Studio 6 (VC++,VB®) NI LabVIEW® 8.0 or newer MATLAB® 7 or newer Delphi® Borland C++ Agilent VEE® Python	NI CVI NET (VC++, VB.net, C# 2003,2005,2008,2010) NI LabVIEW®_2009 or newer MATLAB® 2008 or newer Delphi® Borland C++

* Additional 3RD party software are supported, contact Mini-Circuits for details.

How to register mcl_Gen.dll, 32-bit DLL, on a 32-bit Windows operating system?

Open the Run Command from the Start Menu and type regsvr32 c:\windows\system32\mcl_Gen.dll



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: regsvr32 c:\windows\system32\mcl_Gen.dll

Figure 3 – Run Command

How to register mcl_Gen.dll, 32-bit DLL on a 64-bit Windows operating system?

- Run the Command Prompt as Administrator, see figure 4

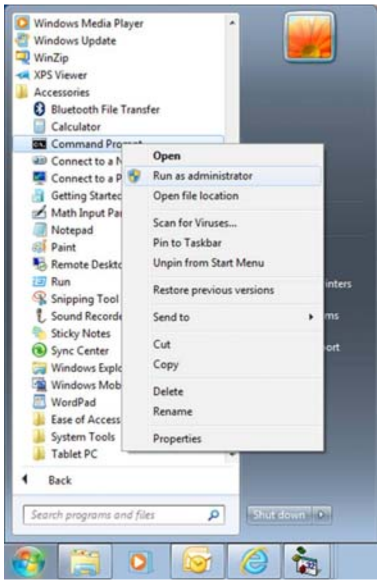
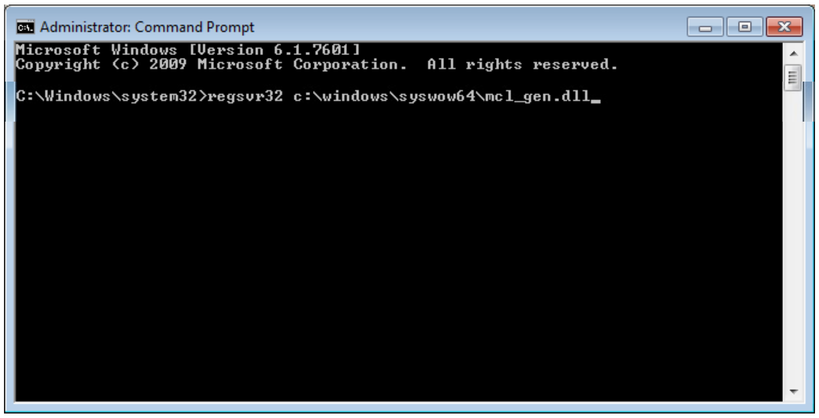


Figure 4 – Command Prompt

- Type regsvr32 c:\windows\syswow64\mcl_Gen.dll, see figure 5

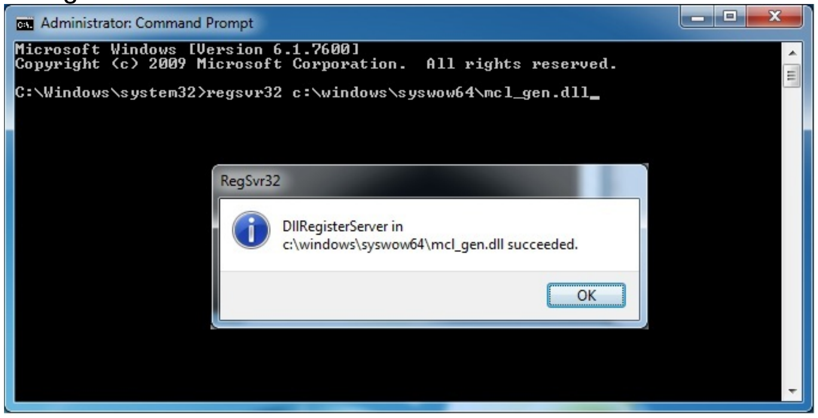


```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>regsvr32 c:\windows\syswow64\mcl_gen.dll_
```

Figure 5 – Type command

- Click Enter, see figure 6.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>regsvr32 c:\windows\syswow64\mcl_gen.dll_
```

RegSvr32

DllRegisterServer in c:\windows\syswow64\mcl_gen.dll succeeded.

OK

Figure 6 – Registration succeeded

2.2 - DLL Structure (both *mcl_Gen.dll* and *mcl_Gen64.dll*)

DLLs Functions mcl_Gen.dll / mcl_Gen64.dll

1. Int Connect(Optional *string SN)
2. Int ConnectByAddress(Optional Short Address)
3. Void Disconnect()
4. Int SetPowerON()
5. Int SetPowerOFF()
6. Byte SetFreqAndPower(Double Fr, Float Pr, Int TriggerOut)
7. Byte SetFreq(Double Fr, Int TriggerOut)
8. Byte SetPower(Float Pr, Int TriggerOut)
9. Byte Set_Noise_Spur_Mode(Int nsm)
10. Int ExtRefDetected()
11. Int Read_ModelName(String ModelName)
12. Int Read_SN(String SN)
13. Int GetTriggerIn_Status()
14. Float GetGenMaxFreq()
15. Float GetGenMinFreq()
16. Float GetGenMaxPower()
17. Float GetGenMinPower()
18. Float GetGenStepFreq()
19. Byte Get_Noise_Spur_Mode(Int nsm)
20. Int GetGenStatus(Byte Locked, Int PowerIsOn, Double Fr , Float pr, UNLEVELHigh, UNLEVELLow)
21. Int GetStatus()
22. Int Set_Address(Int Address)
23. Int Get_Address()
24. Int Get_Available_SN_List(String[] SN_List)
25. Int Get_Available_Address_List(String[] Add_List)

Functions Description:

1. Int Connect(Optional *String SN)
   This function creates a USB connection to the device.

   SN is the Generator Serial Number.
   SN parameter is needed when more than one Generator is connected to the computer.

   The function returns a value as follows:
   0=Failed to connect to device
   1=Success
   2=Device is already connected
   3=SN is not available

2. Int ConnectByAddress(Optional Short Address)

   Address parameter is required in case more than 1 Generator is connected to the PC and you want to connect the unit by Address instead of SN.
   This is an alternative to function 1 (connect by SN)

   Address parameter is the Address of the Generator.
   Address value can be any integer number between 1 to 255 and can be changed by software .

3. Void Disconnect()
   Close connection to the Generator.
   Shutting down the program without disconnecting the device may result in connection problem to the device. If you experience problems, shut down the program, then unplug the Generator from the computer and plug it back in before starting the program again.

4. Int SetPowerON() - Turn RF Power ON
   Function returns nun zero number upon success.

5. Int SetPowerOFF() - Turn RF Power OFF
   Function returns nun zero number upon success.

6. Byte SetFreqAndPower(Double Fr, Float Pr, Int TriggerOut)

   Sets the Generator Frequency and Power value and enables or disables Trigger-Out function.

- Fr is the requested Frequency in MHz.
- Pr is the requested Power in dBm.
- TriggerOut=1 to enable Trigger Out or =0 to disable Trigger Out.

7. Byte SetFreq(Double Fr, Int TriggerOut)

   Sets the Generator Frequency value without changing power values and enables or disables Trigger-Out function

- Fr is the requested Frequency in MHz.
- TriggerOut=1 to enable Trigger Out or =0 to disable Trigger Out .

8. Byte SetPower(Float Pr, Int TriggerOut)

   Sets the Generator Power value without changing frequency values and enables or disables Trigger-Out function.

- Pr is the requested Power in dBm.
- TriggerOut=1 to enable Trigger Out or =0 to disable Trigger Out.

9. Byte Set_Noise_Spur_Mode(Int nsm)

   Set the Low Spur or Low Noise Mode.
- nsm=1 for Low Spur mode or nsm=0 for Low Noise Mode (Default).

10. Int ExtRefDetected ()

    The function returns 1, if 10 MHz External Reference is detected.

11. Int Read_ModelName(String ModelName)
    Read the Model Name of the Generator device. Returns 1 upon success.

12. Int Read_SN(String SN)
    Read the Serial Number of the Generator device. Returns 1 upon success.

13. Int GetTriggerIn_Status()
    Function returns the Trigger Status.

14. Float GetGenMaxFreq()
    Function returns the Generator's maximum frequency in MHz.

15. Float GetGenMinFreq()
    Function returns the Generator's minimum fFrequency in MHz.

16. Float GetGenMaxPower()
    Function returns the Generator's maximum power in dBm.

17. Float GetGenMinPower()
    Function returns the Generator's minimum power in dBm.

18. Float GetGenStepFreq()
    Function returns the Generator's Step Freq in kHz.

19. Byte Get_Noise_Spur_Mode(Int nsm)
    Get the status of Noise/Spur Mode.

- nsm is set to 1 for Low Spur mode or 0 for Low Noise Mode.

20. Int GetGenStatus(Byte Locked, Int PowerIsOn, Double Fr, Float pr, UNLEVELHigh, UNLEVELLow)

    Use this function to get the status of the Generator.

- Locked=1 if the Frequency is Locked.
- PowerIsOn=1 if the RF Power is ON.
- Fr will set to the Generator frequency in MHz.
- pr will set to the Generator Power in dBm.
- UNLEVELHigh will be set to 1 if the Generator Power is too high and the Generator cannot reach the Requested Power
- UNLEVELLow will be set to 1 if the Generator Power is too low and the Generator cannot reach the Requested Power

21. Int GetStatus()

    Get the status of the device connection.

22. Int Set_Address(Int Address)

    Sets the address of the unit. The address can be any integer number between 1 to 255.
    The function returns a non-zero value upon success.

23. Int Get_Address ()

    Returns the device address.
    The function returns a non-zero value upon success.

24. Int Get_Available_SN_List(String[] SN_List)

    String SN_List variable contains the SN of all available Generators connected to the computer.
    The function returns the Number of Generators.

25. Int Get_Available_Address_List(String[] Add_List)

    String Add_List variable contains the Addresses all available Generators connected to the computer.
    The function returns the Number of Generators.

2.3 - Sample code

The CD package also includes a number of sample programs developed to show you how to write your own programs. The sample programs were developed in Visual C++®, Visual Basic®, C# and LabVIEW®. The sample programs provide an excellent starting point to write your own applications.

The complete project examples are available for download at:
http://www.minicircuits.com/support/software_download.html

3 - Operating in a Linux® Environment 32/64Bits OS with USB HID Support

To open a connection to a Synthesized Signal Generators, Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID is: 0x20CE
- USB Synthesized Signal Generators Product ID is: 0x12

The communication with the Generator is done by USB Interrupt.
The transmitted and received buffer sizes are 64 Bytes.

Transmit Array should be 64 bytes	[Byte 0][Byte1][Byte2]………[Byte 63]
Receive Array contains 64 bytes	[Byte 0][Byte1][Byte2]………[Byte 63]

Commands List:

#	Description	Command Code – Byte[0]	Additional Transmitted Bytes
1	Get device Model Name	40	--
2	Get device Serial Number	41	--
3	Set both frequency & power	103	Byte[1] to Byte[4] – Freq in Hz Byte[5] '1' if Power <0 else '0' Byte[6] to Byte [7] Absolute Power in dBm Byte[8] '1' if TriggerOut is required,else '0'
4	Set the frequency	101	Byte[1] to Byte[4] – Freq in Hz Byte[5] '1' if TriggerOut is required,else '0'
5	Set the power	102	Byte[1] '1' if Power <0 else '0' Byte[2] to Byte [3] Absolute Power in dBm Byte[4] '1' if TriggerOut is required,else '0'
6	Set RF Power ON/OFF	104	Byte[1] '1' for Power ON, '0' for OFF
7	Set Noise_Spur_Mode	106	Byte[1] '1' Low Spur Mode, '0' for low Noise Mode
8	Get Generator status	105	--
9	Get Generator minimum frequency	42	--
10	Get Generator maximum frequency	43	--
11	Get Generator step frequency	44	--
12	Get Generator minimum power	45	--
13	Get Generator minimum power	46	--

* See detailed description on pages 12 - 15

1. Get Device Model Name:

To get the device Model Name, code number 40 should be sent

Transmit Array

- Byte[0]=40
- Bytes[1] through [63] are NC - Not Care

Received Array

The Model Name will be returned in the receive array of ASCII characters. End of Model Name is signified by a 0 value.

- Byte[0]=40
- Byte[1] to the byte before the 0 value = Model Name
- All bytes after the 0 value up to byte [63] contain random values

2. Get Device Serial Number

To get the device serial number, code number 40 should be sent

Transmit Array

- Byte[0]=41
- Bytes[1] through [63] are NC - Not Care

Received Array

The Serial Number will be returned in the receive array of ASCII characters. End of Serial Number is signified by a 0 value.

- Byte[0]=41
- Byte[1] to the byte before the 0 value = Serial Number
- All bytes after the 0 value up to byte [63] contain random values

3. Set both Freq And Power

Transmit Array

- Byte[0]=103
- Byte[1] to Byte[4]= requested frequency in Hz. (Byte[1] is the MSB).
- Byte[5] = '1' if the requested Power lower than Zero, else '0'
- Byte[6] to Byte[7]= The absolute value of the power in dBm (Byte[7] is the MSB)
- Byte[8]= '1' if Trigger Out is required, else '0'
- Bytes[9] through [63] are NC - Not Care

Received Array

- Byte[0]=103
- Bytes[1] through [63] contain random values

4. **Set Freq**:

   Transmit Array

   - Byte[0]=101
   - Byte[1] to Byte[4]= requested frequency in Hz. (Byte[1] is the MSB).
   - Byte[5] = '1' if Trigger Out is required, else '0'
   - Bytes[6] through [63] are NC - Not Care

   Received Array

   - Byte[0]=101
   - Bytes[1] through [63] contain random values

5. **Set Power:**

   Transmit Array

   - Byte[0]=102
   - Byte[1]= '1' if the requested Power lower than Zero, else '0'
   - Byte[2] to Byte[3]= The absolute value of the power in dBm (Byte[3] is the MSB)
   - Byte[5] = '1' if Trigger Out is required, else '0'
   - Byte[6]= '1' if Trigger Out is required, else '0'
   - Bytes[7] through [63] are NC - Not Care

   Received Array

   - Byte[0]=102
   - Bytes[1] through [63] contain random values

6. **Set RF Power ON/OFF:**

   Transmit Array

   - Byte[0]=104
   - Byte[1]= '1' for Power ON, '0' for OFF
   - Bytes[2] through [63] are NC - Not Care

   Received Array

   - Byte[0]=104
   - Bytes[1] through [63] contain random values

7. **Set Set_Noise_Spur_Mode**:

   Transmit Array

   - Byte[0]=106
   - Byte[1]= '1' Low Spur Mode, '0' Low Noise Mode (Default)
   - Bytes[2] through [63] are NC - Not Care

   Received Array

   - Byte[0]=106
   - Bytes[1] through [63] contain random values

8. **GetGenStatus** - Get the Generator status (current freq , power etc)

Transmit Array

- Byte[0]=105
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=105
- Byte[1]= 1 RF ON, 0 if RF OFF
- Byte[2]= 1 if frequency is Locked, 0 if frequency is Unlocked
- Byte[3] through Byte[6] represent the Generator current frequency in Hz. (Byte4 is the MSB).
- Byte[7]= 1 if the Generator current Generator setting power is below 0Bm.
- Byte[8] through Byte[9] Absolute of the current Generator power(dBm).Byte9 is the MSB
- Byte[10]= 1 if the requested Power is too High - Unlevel High
- Byte[11]= if the requested Power is too Low - Unlevel Low.
- Bytes[12] through [63] contain random values

9. **GetGenMinFreq**

Transmit Array

- Byte[0]=42
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=42
- Byte[1] through Byte[4]= Generator's minimum frequency in Hz. Byte[1] is the MSB
- Bytes[5] through [63] contain random values

10. **GetGenMaxFreq**

Transmit Array

- Byte[0]=43
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=43
- Byte[1] through Byte[4]= Generator's maximum frequency in Hz. Byte[1] is the MSB
- Bytes[5] through [63] contain random values

11. **GetGenStepFreq**

Transmit Array

- Byte[0]=44
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=44
- Byte[1] through Byte[4]= Generator's step frequency in Hz. Byte[1] is the MSB
- Bytes[5] through [63] contain random values

12. GetGenMinPower

Transmit Array

- Byte[0]=45
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=42
- Byte[1]= '1' if the minimum power is negative, else '0'
- Byte[2] through Byte[3]=Generator's minimum power in dBm absolute value. Byte[2] is the MSB
- Bytes[4] through [63] contain random values

13. GetGenMaxPower

Transmit Array

- Byte[0]=46
- Bytes[1] through [63] are NC - Not Care

Received Array

- Byte[0]=46
- Byte[1]= '1' in case the maximum power is negative, else '0'
- Byte[2] through Byte[3]=Generator's maximum power in dBm absolute value. Byte[2] is the MSB
- Bytes[4] through [63] contain random values

3.1 – Sample code

The Linux Folder in the CD package contains the following:

The Linux Folder in the CD package contains the following:

- Generator.c example source code using the libhid & libusb libraries to open the USB HID device.

The complete project samples are available on the CD or at:
http://www.minicircuits.com/support/software_download.html