Mini-Circuits®

Programming Manual

For the

USB Smart Power Sensor PWR Series

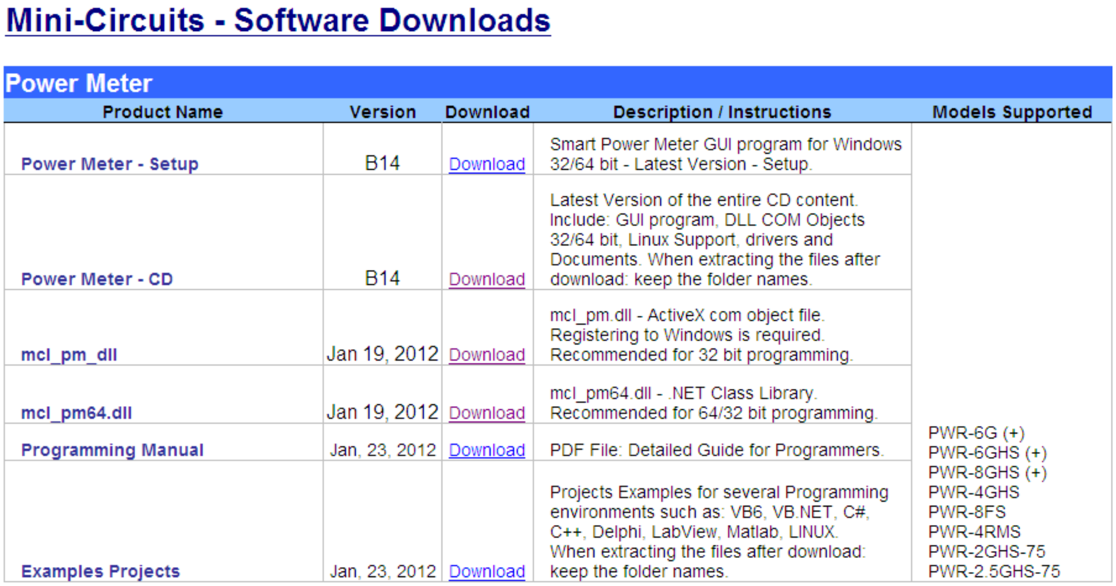
Contents

1- Overview

This programming Manual is intended for customers wishing to create their own interface for Mini-Circuits' USB **P**ortable **T**est **E**quipment (PTE).

Mini-Circuits offers support of the USB PTE for Windows® and Linux® Operating Systems, as well as for variety of programming environments including third-party applications such as LabVIEW® and MATLAB® through .NET assembly and ActiveX® Controls to write your own customized measurement 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.



Mini-Circuits - Software Downloads

Power Meter

Product Name	Version	Download	Description / Instructions	Models Supported
Power Meter - Setup	B14	Download	Smart Power Meter GUI program for Windows 32/64 bit - Latest Version - Setup.	PWR-6G (+) PWR-6GHS (+) PWR-8GHS (+) PWR-4GHS PWR-8FS PWR-4RMS PWR-2GHS-75 PWR-2.5GHS-75
Power Meter - CD	B14	Download	Latest Version of the entire CD content. Include: GUI program, DLL COM Objects 32/64 bit, Linux Support, drivers and Documents. When extracting the files after download: keep the folder names.	
mcl_pm_dll	Jan 19, 2012	Download	mcl_pm.dll - ActiveX com object file. Registering to Windows is required. Recommended for 32 bit programming.	
mcl_pm64.dll	Jan 19, 2012	Download	mcl_pm64.dll - .NET Class Library. Recommended for 64/32 bit programming.	
Programming Manual	Jan, 23, 2012	Download	PDF File: Detailed Guide for Programmers.	
Examples Projects	Jan, 23, 2012	Download	Projects Examples for several Programming environments such as: VB6, VB.NET, C#, C++, Delphi, LabView, Matlab, LINUX. When extracting the files after download: keep the folder names.	

Figure 1 – Download Screen

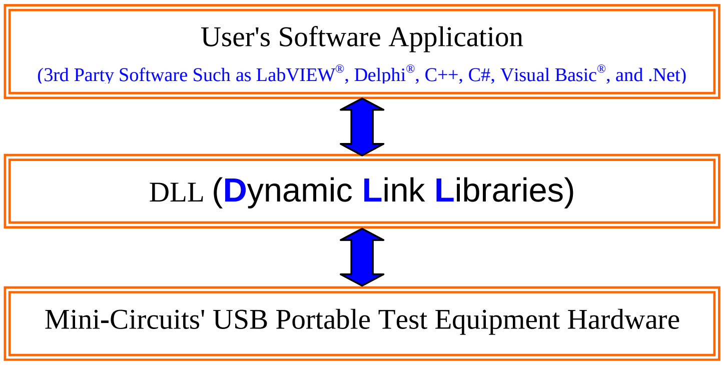
2 - Operating in a Windows® Environment 32/64Bits OS with USB HID Support

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 (**D**ynamic **L**ink **L**ibraries)

Mini-Circuits' USB Portable Test Equipment Hardware

Figure 2 – DLL Interface

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

1. ActiveX com object - *mcl_pm.dll* → Click to Download http://yoni.minicircuits.com/downloads/mcl_pm_dll.zip
   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_pm64.dll* → Click to Download http://yoni.minicircuits.com/downloads/mcl_pm64_DLL.zip
   .NET object – a logical unit of functionality that runs under the control of the .NET

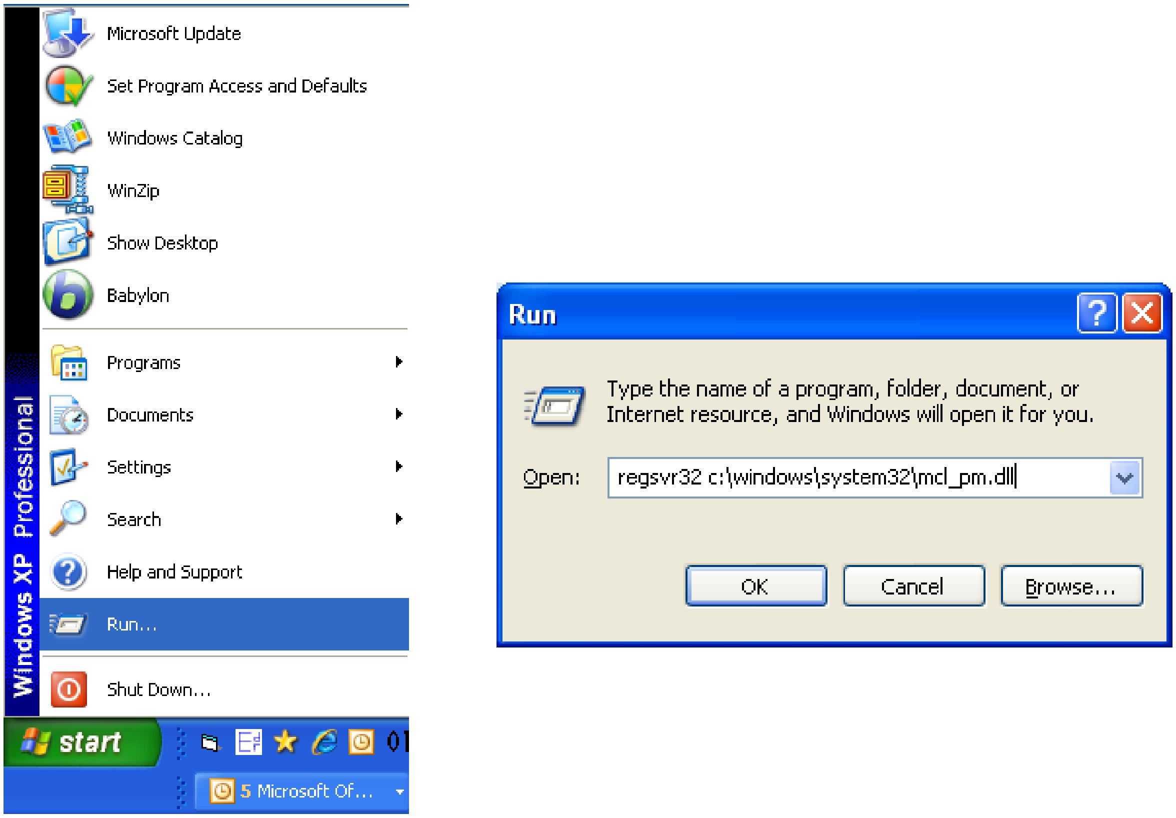
2.1 - Software supported by ActiveX® and .NET Class Library

mcl_pm.dll - ActiveX® com object	mcl_pm64.dll - .NET Class Library
Instructions ▪ Register the DLL, see instructions below ▪ For **32bit** Windows OS, Copy mcl_pm.dll to windows\system32 folder ▪ For **64bit** Windows OS, Copy mcl_pm.dll to windows\SysWOW64 folder	**Instructions** ▪ DLL Registry is not required ▪ For **32bit** Windows OS Copy mcl_pm64.dll to windows\system32 folder ▪ For **64bit** Windows OS Copy mcl_pm64.dll to windows\SysWOW64 folder
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_pm.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_pm.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_pm.dll

Figure 3 – Run Command

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

- Run the command Prompt as Administrator, see figure 4

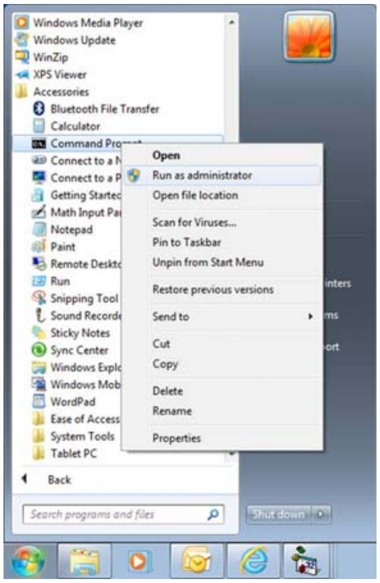
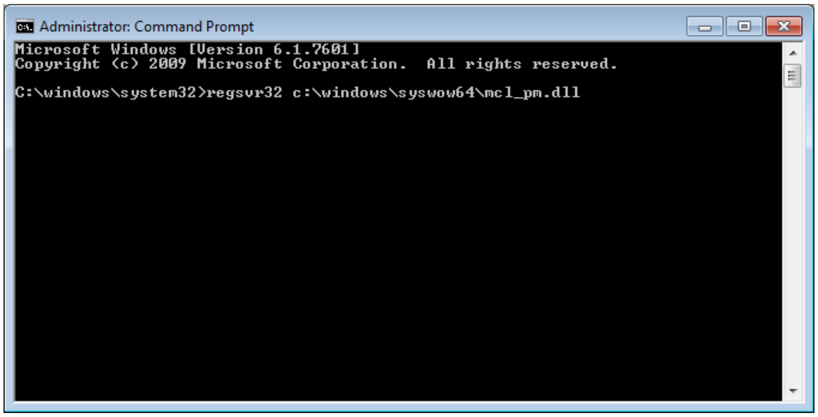


Figure 4 – Command Prompt

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

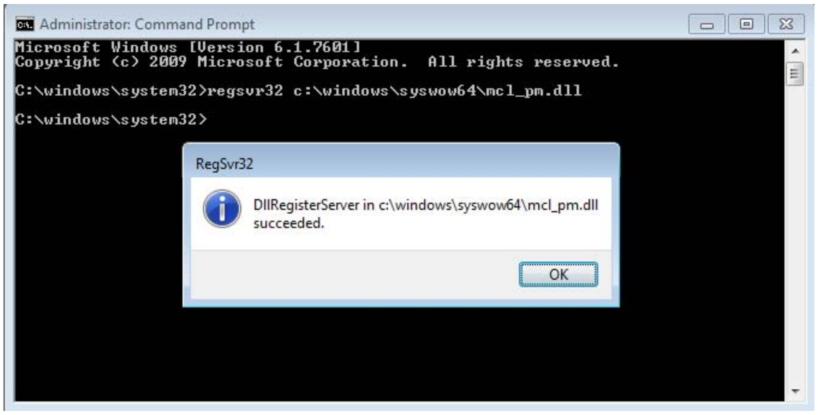


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\windows\system32>regsvr32 c:\windows\syswow64\mcl_pm.dll
```

Figure 5 – Type command

- Click Enter, see figure 6.



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

C:\windows\system32>regsvr32 c:\windows\syswow64\mcl_pm.dll

C:\windows\system32>
```

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

Figure 6 – Registration was succeeded

2.2 - DLL Structure (both *mcl_pm.dll* and *mcl_pm64.dll*)

DLLs Functions mcl_pm.dll / mcl_pm64.dll

1. Long Open_Sensor(Optional *string SN)
2. Long Open_AnySensor()
3. Float ReadPower()
4. Int GetStatus()
5. Void Close_Sensor()
6. String GetSensorSN()
7. String GetSensorModelName()
8. Void SetFasterMode(Short)
9. Float GetDeviceTemperature(Optional TemeratureFormat As String)
10. Long Get_Avaliable_SN_List(*string SN_List)

DLLs Properties mcl_pm.dll / mcl_pm64.dll

1. Freq
2. AVG
3. AvgCount
4. Format_mW

Functions Description:

1. Long Open_Sensor(Optional *string SN)

   SN parameter is needed when more than one Power Sensor is connected to the computer.
   SN is the Serial Number of the Power Sensor.

   The function returns a Value as follows:
   0=Failed to open Sensor
   1=Success
   2=Object is already open
   3=SN is not Available

2. Long Open_AnySensor() function can be used when only one Power Sensor is connected to the computer.

   The function returns a value as follows:
   0=Failed to open sensor
   1=Success
   2=Object is already open
   3=SN is not available

3. Float ReadPower()
   - Returns the power Value

4. Int GetStatus()

   The Function returns a value as follows:
   1=Sensor is connected
   0=Sensor is disconnected

5. Void Close_Sensor()
   - Close connection to the sensor

6. String GetSensorSN()
   - Returns the Sensor's Serial Numbers

7. String GetSensorModelName()
   - Returns the Sensor's Model Name

8. Void SetFasterMode(Short Mode)

   Mode=0, Low Noise Mode
   Mode=1, Faster Mode (Not available for the PWR-6G)
   Mode=2, Fastest Mode (Available only for the PWR-8FS)

9. Float GetDeviceTemperature(Optional TemeratureFormat As String)
   - TemperatureFormat - optional parameter: "F" for Fahrenheit "C" (Default) for Celsius

10. Long Get_Avaliable_SN_List(*string SN_List)
    - SN_List returns a List of all available Serial Numbers (separated by a single space character):
    [SN1] [SN2] [SN3] [SN4]...

Properties Description:

1. Freq: the CAL Freq in MHz
2. AVG: 1=ON 0=OFF
3. AvgCount: the No of Avg Count
4. Format_mW: if true, the ReadPower result will be in mW

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#, Visual 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://yoni.minicircuits.com/downloads/mcl_pm_Examples.zip

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

To open a connection to the power sensor, Vendor ID and Product ID are required:

- Mini-Circuits Vendor ID is: 0x20CE
- Power Sensor Product ID is: 0x11

The communication with the sensor is done by USB Interrupt.
The Transmit and receive buffer size is 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 the Sensor power value	102	Byte[1] - Freq MHz (MSB) Byte 2] - Freq MHz (LSB)
2	Set the Sensor to Low Noise Mode	15	Byte[1] - 0
3	Set the Sensor to Faster Mode **NOT Available for the PWR-6G**	15	Byte[1] - 1
4	Set the Sensor to Fastest Mode **Available ONLY for the PWR-8FS**	15	Byte[1] - 2
5	Get the Sensor Model Name	104	--
6	Get the Sensor Serial Number	105	--
7	Get the Sensor Temperature	103	--

* See detailed description on pages 10 - 13

1. Read Power Value:

To get the power value from the sensor, send to the sensor code number 102 and also the Frequency value for the correction cal factors.
The Frequency is in MHz and should be Integer Value split into 2 Bytes (MSB and LSB)

- Byte[0] =102
- Byte[1] = Frequency - MSB
- Byte[2] = Frequency – LSB

Byte [0]	Byte [1]	Byte [2]	▯□ ⇨	Byte [63]
102	Frequency - MSB	Frequency – LSB		

Example: to read power at frequency of 3000 MHz, then send

- Byte[0] =102
- Byte[1] =11
- Byte[2] =184
- Bytes[3] through [63] are NC - Not Care

(3000=11*256+184)

Transmit Array

Byte [0]	Byte [1]	Byte [2]	▯□ ⇨	Byte [63]
102	11*256	184		

The Power will be returned in 8 Bytes array in this way:

- Byte[0]=102 - the sensor return the same code number.
- Byte[1] up to Byte[6] - the reading power(dBm) in ASCII characters
- Byte[7]= 0
- Bytes[8] through [63] contain random values

Received Array

Byte [0]	Byte [1]	Byte [2]	Byte [3]	Byte [4]	Byte [5]	Byte [6]	Byte [7]	▯▯ ⇨	Byte [63]
102									
102	Sign	Number	Number	Dot	Number	Number	0		

ASCII characters represent the Power

2. **Set Low Noise Mode:** (this is the sensor's default mode)

To set the sensor to Low Noise Mode, code number 15 should be sent and then the second byte should be 0

Transmit Array

- Byte[0]=15
- Byte[1]=0
- Bytes[2] through 63 are NC - Not Care

Received Array

- Byte[0]=15
- Bytes[1] through [63] contain random values

3. **Set Faster Mode:** **NOT Available for the PWR-6G**

To set the sensor to Faster Mode, code number 15 should be sent and the second byte should be 1

Transmit Array

- Byte[0]=15
- Byte[1]=1
- Bytes[2] through [63] are NC - Not Care

Received Array

- Byte[0]=15
- Bytes[1] through [63] contain random values

4. **Set Fastest Mode:** **Available ONLY for the PWR-8FS**

To set the sensor to Fastest Mode, code number 15 should be sent and the second byte should be 1.

Transmit Array

- Byte[0]=15
- Byte[1]=2
- Bytes[2] through [63] are NC - Not Care

Received Array

- Byte[0]=15
- Bytes[1] through [63] contain random values

5. Get Power Sensor Model Name:

To get the sensor Model Name, code number 104 should be sent

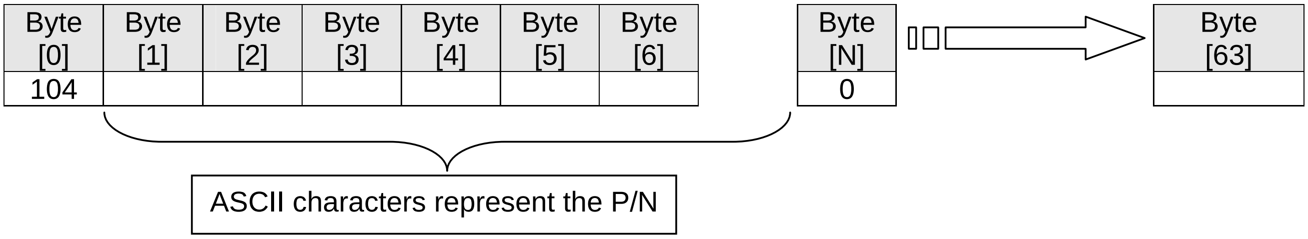
Transmit Array

- Byte[0]=104
- Bytes[1] through [63] are NC - Not Care

Received Array

The model name will be return in the receive array in ASCII characters ending with 0.

- Byte[0]=104
- Byte[1] to the byte with value=0 = Part Number
- All bytes after the 0 character up to byte [63] contain random values



Byte [0]	Byte [1]	Byte [2]	Byte [3]	Byte [4]	Byte [5]	Byte [6]	Byte [N]	Byte [63]
104							0	

ASCII characters represent the P/N

6. Get Power Sensor Serial Number:

To get the sensor Serial Number, code number 105 should be send

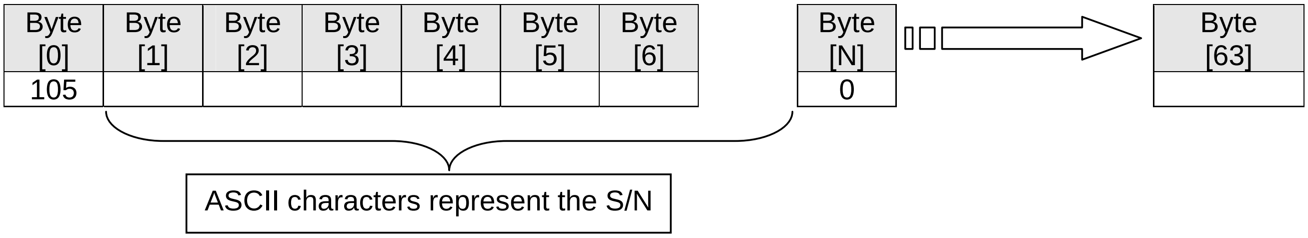
Transmit Array

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

Received Array

The Serial Number will be return in the receive array in ASCII characters ending with 0.

- Byte[0]=105
- Byte[1] to the byte with value=0 = Serial Number
- All bytes after the 0 character up to byte [63] contain random values



Byte [0]	Byte [1]	Byte [2]	Byte [3]	Byte [4]	Byte [5]	Byte [6]	Byte [N]	Byte [63]
105							0	

ASCII characters represent the S/N

7. Read Device Temperature:

To get the sensor Temperature, code number 103 should be send

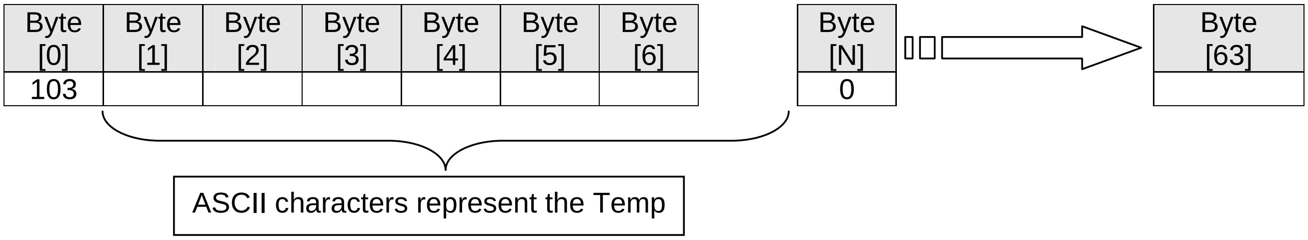
Transmit Array

- Byte[0]=103
- Bytes[1] through [63] are NC - Not Care

Received Array

The Temperature will be return in the receive array in ASCII characters ending with 0.

- Byte[0]=103
- Byte[1] to the byte with value=0 = Device Temperature in Celsius
- All bytes after the 0 character up to byte [63] contain random values



Byte [0]	Byte [1]	Byte [2]	Byte [3]	Byte [4]	Byte [5]	Byte [6]	Byte [N]	Byte [63]
103							0	

ASCII characters represent the Temp

3.1 – Sample code

The Linux Folder in the CD package contains the following:

Name

PowerMeter.tar.gz

PowerMeterUSB_LINUX

ReadMe1st.txt

Figure 7 – Linux Folder

1. PowerMeter.tar.gz is a tar (gzip) archive which contains:

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

- MCL_PM an executable to be run under LINUX terminal,
- The command line: sudo ./MCL_PM [frequency in MHz] for example for 10MHz the command is sudo ./MCL_PM 10
  (The frequency must be within the range of the power sensor used).

2. PowerMeterUSB_LINUX is a GUI sample program under Linux using GTK.

- You will need administrator/Root permissions to run it.
- Command line from terminal: Sudo. /PowerMeterUSB_LINUX
- After the GUI launch, enter the frequency in the frequency text box.
- It will return the measured Power and Temperature.

Linux Project Examples are also available on the CD package and can be downloaded at:
http://yoni.minicircuits.com/downloads/mcl_pm_Examples.zip