This note discusses the use of the sample code (or test harnesses). There are four test harness project (with source code) optimized to exercise discrete function calls. The purpose of the test harnesses is to allow the user to become familiar with the various calls required to accomplish certain tasks. This note will show how to make the most basic measurements. And it specifically addresses the pulse profiling test harness. The other test harnesses are subsets of this project – except the high speed logger test harness. This test harness largely duplicates the logger application shipped with the product.

Note that some changes have been made since this document was first written. However, the changes are largely enhancements and haven't changed the essential elements demonstrated in this document. Also, all of the code for these projects is available in the sample code solution.

Specifically this document will cover:

- CW measurements (applies to all test harnesses and products)
- Pulse modulation measurements (applies to peak/pulse and pulse profiling products)
- Pulse profiling products only

We'll start by assuming:

- 1. You've installed the either or both of the power meter and pulse profiling applications
- 2. You have a sensor connected
- 3. You've been able to make measurements

The sample code may be shipped as a zipped file. The zip file contains a single solution containing four projects. Copy and extract the files to directory of your choosing. Load the application into Visual Studio 2010 or you can start the pulse profiling test harness directly. All of the code in the C# and VB.Net projects is available. You can start the pulse profiling project in Visual Studio 2010 or start the executable. The name of the debug version of the pulse profiling test harness executable is:

## "Csh\_PM\_PP\_TestHarness.exe"

There are also executable versions of the VB.Net test harness (power meter only), a similar C# power meter test harness and a high speed logger test harness. Each has source code a compile executable. They are named:

- "High Speed Logger.exe"
- "Csh\_PM\_TestHarness.exe"
- "VB\_PM\_TestHarness.exe"

When you start the pulse profiling test harness application you should see a window similar to the one that follows:

RgrTest_Tabbed LB_SensorCnt() Idx Adr LB_SensorList() V Auto fill on select	Sensor Coun SN Ser Ir Add	t No. dex 1 clear					
Index & Address Functions Init, CW & Pulse Mod	Meas Criteria & Offsets Limits	Triggering Record	ler Out PP - Setup	PP - Get Trace	PP - Markers	PP - Gates	Options
LB_GetAddress_Idx()	LB	BlinkLED_Idx( )					
LB_GetAddress_SN()		BlinkLED_Addr( )					
LB SetAddress Idx()	·	BlinkLED SN()	-				
LB SetAddress SN()		#FirmwareVersion()					
					-		
		SetCalDueDate()	10/2//08 💌	•	-		
LB_GetIndex_SN()	GetM	odelNumber_SN()					
LB_AddressConflictExists( not tested	GetM	odelNumber_Idx( )					
LB_WillAddressConflict() 1 🗢	not tested GetMo	delNumber_Addr()					
LB_GetSerNo_Idx()	[Let-up	Courseland CN(C)	Ser No.	Yes/No/Err	-		
LB_GetSerNo_Addr()	Issens	orConnected_SIN()	Address	Yes/No/Err			
	IsSens	orConnected_Addr(	1 📚	100/HO/Ell			
LB_DriverVersion							
							Close

We'll start by focusing on getting basic information. Click the sensor count button. You should get a count in the adjacent label. Sensor count is a very simple call. It has the following prototype:

## int LB\_SensorCnt()

Note in the above line that the return value is "int". It's important to remember that the C# "int" is the same as a C++ LONG or a 32 bit integer. The line of code that makes use of this call in is:

## lblLB\_SensorCount.Text = LB\_API2\_Declarations.LB\_SensorCnt().ToString();

This call does exactly it says it does. It returns a count of sensors. The "LB\_" indicates that this function is useful for CW, pulse modulation and pulse profiling measurements. You'll note that there are several tabs.

grTest_Tabbed
LB_SensorCnt()     1     Sensor List       Int+-     Adr       LB_SensorList()     1       1     122       086111     Index       Index     1       Address     1       Clear
Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Offsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
LB_GetAddress_Idx() · · LB_BlinkLED_Idx() ·
LB_GetAddress_SN() · LB_BlinkLED_Addr() ·
LB_SetAddress_ldx() 1 C LB_BlinkLED_SN()
LB_SetAddress_SN() 1 C . LB_GetFirmwareVersion()
LB_ChangeAddress 1 - LB_GetCalDueDate()
LB_GetIndex_Addr() · LB_SetCalDueDate() 10/27/08 ·
LB_GetIndex_SN() · GetModelNumber_SN()
LB_AddressConflictExists( not tested GetModelNumber_Idx( )
LB_WillAddressConflict() 1 I not tested GetModelNumber_Addr()
LB_GetSerNo_Idx() Ser No. Yes/No/Err
LB_GetSerNo_Addr() Address Yes/No/Err
IsSensorConnected_Addr( 1 📚
LB_DriverVersion
Close

Now click the sensor list button (as shown above). This will bring up a list of your sensors. You'll note that the USB index, address and serial number are listed. Now is the time to get acquainted with how the panel works. To start this process, select or click a sensor in the list box. I've selected the sensor with a USB index of 1, an address of 122 and a serial number of 086111. When this is done all the labels and the list box to the right are filled in. This is done through a series of calls.

You must select a sensor to get many of the calls to be made without error!

	I selected this sensor to get the information
RgrTest_Tabbed  LB_SensorCnt()  Idx Adr SN  LB_SensorList()  Auto fill on select	Model No.         LB400A         Fit Opt = ST_ON Trg Opt = ST_ON Trg Opt = ST_ON Com type = STA_H           Index         1          Address         37_OFF           Address         122         00F Exp = 10/27/08         Cal Opt Exp = 10/27/08
Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Diffsets  LB_GetAddre LB_SetAddre LB_SetAddre LB_Changer LB_GetIndex	Links Triggering F corder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
LB_GetIndex_SN()       .         LB_GetIndex_SN()       .         LB_MiRAddressConflict()       1 Image: Conflict Conflite Conflict Conflict Conflict Conflict Conf	GetModeNumber_SN()       GetModeNumber_Idd()       GetModeNumber_Add()       Ser No.       IsSensorConnected_SN()       Address       Yes/No/Err

You can see options, cal date, serial number, model number etc are all acquired. Clearly this requires a number of calls. Fortunately each is fairly simple. Below is a snippet of the code in the IndexChanged event for the list box. You'll note that these calls (as do most) require you to identify the sensor. Sometimes you have an option of serial number, address or index. However, the most common identifier is the address.

```
rslt = LB_API2_Declarations.LB_GetModelNumber_Addr(currAddr, ref mn);
```

```
rslt = LB_API2_Declarations.LB_GetFilterOpt(currSN, ref optVal);
```

The first line gets the model number (as an enumeration) and the second one retrieves the value of the filter option. This is characteristic of most of the code. Some error checking is required. This is primarily needed to trap null pointers to serial number strings.

Now we'll examine how this panel works. The top portion we just filled in with the serial number, model number, address, etc is used to identify the sensor with which we want to communicate. <u>Again, if you</u> <u>don't select a sensor you'll get numerous errors.</u> We've select a sensor so let's make a few calls:

I've clicked four buttons (marked in light green). Next to each button is a value. On the two top buttons you'll note the address has been returned. On the bottom two indicated buttons you'll see the indexes have been returned.

Now we'll change the address of our selected device. There are three possible calls to change the address. They are LB\_SetAddress\_Idx, LB\_SetAddress\_SN and LB\_ChangeAddress. The first two use the index (\_Idx) and serial number (\_SN) to change the serial number. The last one passes the current address and new address. In each case the calls are obtaining the current serial number, address or index filled in when we clicked the sensor list box.

To complete the information we set the numeric up/down counter adjacent to the button to set the address we want to change to. The result of the call is then indicated by a "Success" or "Failed" message. After each address change you will need to click the LB\_SensorList button to refresh the list box. Then you'll need to select the device in the list box to update all the information to the right.

RgrTest_Tabbed			
LB_SensorCnit() LB_SensorLisit() V Auto fill on select	1 Idx Adr 1 6 00	SN 96111	Current Device           Model No.         LB480A           Ser No.         006111           Index         1           Address         6           Clear         P10 Opt = ST_OR           Cal 4 Wry Opt = ST_OFF           best Match Opt = ST_OFF           Cal 4 Wry Opt = ST_OFF           Cal 4 Opt = ST_OFF           Cal 4 Opt = ST_OFF           Cal 4 Opt = 10/27/08           Cal 4 Opt = 10/27/08
Index & Address Functions	Init, CW & Pulse Mod	Meas Criteria & Of	ffsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
LB_GetAddress_Idx( )	6		LB_BinkLED_Idx()
LB_GetAddress_SN()	6	Success	
LB_SetAddress_IdA()		-	
LB_ChangeAddress		-	LB_GetCaDueDate()
LB_GetIndex_Addr( )	] 1		LB_SetCaDueDate[] 10/27/08 💌
LB_GetIndex_SN()	1		GetModelNumber_SN()
LB_AddressConflictExists(	not tested		GetModelNumber_Idx()
LB_WillAddressConflict()	1 🗘	not tested	GetModelNumber_Addr()
LB_GetSerNo_Idx()	086111		Ser No. Yes/No/Err
LB_GetSerNo_Addr()	086111		Address Yes/No/Err
LB_DriverVersion			IsSensorConnected_Addi( )
			Liose

Now we're ready to make a simple CW measurement. To being click the next tab (Init, CW & Pulse Mod). You should get a screen like the one below.

RerTest Tabbed	
$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$	
Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Offsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates           LB_InitializeSensor_Idx()         Success           LB_InitializeSensor_Addr()         Entrace	Options
LB_InitializeSensor_SN()     Frequency (Hz)     Pass Fail Cont       LB_SetFrequency()     Frequency (Hz)     Frequency (Hz)	
LB_GetFrequency()     Power Units       SetMeasPowerUnits     Image: Comparison of the comparison o	
Averages       LB_SetAverages()       Averages       LB_GetAverages()	
	Close

Before making any measurements we must initialize the sensor. This takes about 5 seconds. This needs to be done once when the program starts...but before any measurements are taken or any measurement settings are made. It must be done once for each sensor...but multiple calls won'

The previous tab does not require initialization. Almost all remaining calls require initialization. To begin initialization click one of the three buttons highlighted in green above. The only difference in these initializations is the form of identification. Most people will choose to use address since it is the only form of identity under programmatic control.

When you click the button you'll see a message informing you that initialization is in process. When it is completed you'll see a success message. If you initialize again you'll see that the call returns very quickly. This is because the software has already initialized the sensor. Now we can make a CW measurement. This is done very easily by clicking the LB\_MeasureCW button highlighted in red.

To make a pulse measurement click the button hightlighted in blue-green

erTest Tabbed	
LB_SensorCnt() 1 Idx Adr SN LB_SensorList() 1 6 086111 V Auto fill on select	Current Device         Model No.       LB480A         Ser No.       086111         Index       1         Index       1         Best Match Opt = ST_OFF         Best Match Opt = ST_OFF         Best Match Opt = ST_OFF         Clear
Index & Address Functions     Init, CW & Pulse Mod     Meas Criteria & C       LB_InitializeSensor_Idx()     Success       LB_InitializeSensor_Add()     .       LB_InitializeSensor_SN()     .	Jffsets       Limits       Triggering       Recorder Out       PP - Setup       PP - Get Trace       PP - Markers       PP - Gates       Options         LB_MeasureCW()       Pass Fail       Cont         -28.21 dBm         LB_MeasurePulse()       Pass Fail       Cont
LB_SetFrequency()     Frequency (Hz)       LB_GetFrequency()     Power Units       SetMeasPowerUnits     V	Pulse=-20.00 dBm, Peak=-19.89 dBm, CW=-28.20 dBm, DC=14.97%,
GetMeasPowerUnits . Averages LB_SetAverages() Averages LB_GetAverages()	
	Dore

I've highlighted my test result in red and blue green above. To make relative CW measurement follow the process outlined below:

- 1. Select the "Meas Criteria & Offsets" tab
- 2. Set the CW offset by using the Set(Get)CWRefernce
- 3. Return to the Init, CW & Pulse measurements tab
- 4. Use Set(Get)MeasPowerUnits to DBREL (this enables relative measurements)
- 5. Now click the CW measurement. You should see measurements made relative to the current reference.

To make relative use a similar process but use the LB\_Set(Get)PluseReference button. You should see a dialog box similar to the one below (applies to peak/pulse and pulse profiling sensors):

SetPulseRef	
Peak Ref	0.0 🗢
Pulse Ref	0.0 🗢
Avg Ref	0.0 🗢
Duty Cycle Ref	0 🗘
(	OK Cancel

Fill dialog box with appropriate values. And then follow the CW procedure outlined above.

Now we'll proceed with getting a pulse profiling trace, displaying the trace and then making a few measurements with gates. Start by selecting the "PP-Setup" tab. You should see a window like the one shown below:

RgrTest_Tabbed				
LB_SensorCnt( ) LB_SensorList( ) LB_SensorList( ) Auto fill on select	Idx Adr	SN 086111	Current Device         Model No.       LB480A         Ser No.       086111         Index       1         Address       6         Clear       Clear    Fit Opt = ST_ON Trg Opt = ST_ON Conn type = STL_H Cal & Wty Opt = ST_OFF Best Match Opt = ST_OFF Wty Opt Exp = 10/27/08 Cal Opt Exp = 10/27/08	
Index & Address Functions	Init, CW & Pulse Mod	Meas Criteria & O	ffsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options	
InitializeSensor_Addr( )	]		PP_GetTraceLength()	
PP_SetSweepTime()	~		PP_GetSweepTime()	
PP_SetTriggerLevel()	0.0 🗘		PP_GetTriggerLevel()	
PP_SetTriggerSoure()	<b>~</b>		PP_GetTriggerSoure()	
PP_SetTriggerEdge	<b>`</b>		PP_GetTriggerEdge	
PP_SetTriggerOut	) <b>`</b>		PP_GetTriggerOut	
PP_SetFilter()	•		PP_GetFilter()	
PP_SetPoles()	•		PP_GetPoles()	
PP_SetTraceAvgs	1 🗘		PP_GetTraceAvgs	
PP_SetAvgMode()	·		PP_GetAvgMode()	
PP_SetSweepDelay()	1 🗘		PP_GetSweepDelay()	
SetSweepDelayMode	<b>`</b>		GetSweepDelayMode	
				lose

Not the initialize sensor button (highlighted in red). We've already initialized so it is not necessary to do so at this time. This button is provided as a convenience. The defaults for pulse measurements will suffice for now so proceed to the next tab: "PP-GetTrace".

grTest_Tabbed	
LB_SensorCnt()     1       Idx     Adr       SensorList()     1       ✓     Auto fill on select	Current Device         Fit Opt = ST_ON           Model No.         LB480A         Fit Opt = ST_ON           Ser No.         065111         Rec Out Opt = ST_ON           Index         1          Comn type = SHA_H           Cal 4 Wty Opt = ST_OFF         Best Match Opt = ST_OFF           Address         6          Wty Opt Exp = 10/27/08           Clear         Clear         Elloyt Dyt Exp = 10/27/08
Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Offs	ets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
	Close

You should see a screen like the one above. Click PP\_GetTrace() button highlighted in red. Then click the draw trace button highlighted in blue green. I have a pulse signal going into my sensor. With this signal I see the trace shown below

LB_SensolCn()       1         Ldx       Addr         V       Auto fill on select         V       Auto fill on select         Image: SensolList()       Image: SensolList()	RgrTest_Tabbed	
Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Offsets Limits Triggering Recorder Dut PP - Setup PP - Get Trace PP - Markets PP - Gates Options           PP_GetTrace()         Sweep taken at addr 6 array sized to           Idex Adapt (Bar)         Copy           0         -20, 030           2         -20, 026           3         -20, 011           PP_GetTrace()         Draw Trace           PP_GetTraceLength()         Image: PP_GetTraceCorestFacto           PP_GetTracePL/sew         Image: PP_GetTraceDC	LB_SensorCnt()     1       Idx     Adr     SN       LB_SensorList()     1     6     086111       Iv     Auto fill on select     1     1	Current Device         Model No.       LB480A         Ser No.       0861111         Index       1 \$\operatornowcolor: 0 \$\operat
PP_GetTraceAvgPower         PP_GetTracePutsePow         PP_GetTracePow         PD_GetTracePow         PD_GetTracePow <td>Index &amp; Address Functions Init, CW &amp; Pulse Mod Meas Criteria &amp; Off PP_GetTrace() Sweep taken at addr 6 array sized to Idx Amp (dBm) 0 -20.012 1 -20.050 2 -20.025 3 -20.021 4 -20.138 PP_GetTraceLength() PP_CurrTraceZanalysis Intervenayas indeclengtin</td> <td>sets Limits Triggering Recorder Out PP-Setup PP-Get Trace PP-Markers PP-Gates Options</td>	Index & Address Functions Init, CW & Pulse Mod Meas Criteria & Off PP_GetTrace() Sweep taken at addr 6 array sized to Idx Amp (dBm) 0 -20.012 1 -20.050 2 -20.025 3 -20.021 4 -20.138 PP_GetTraceLength() PP_CurrTraceZanalysis Intervenayas indeclengtin	sets Limits Triggering Recorder Out PP-Setup PP-Get Trace PP-Markers PP-Gates Options
	PP_GetTraceAvgPower         PP_GetTracePulsePow         PP_GetTraceCrestFacto         PP_GetTracePkPwr         PP_GetTraceDC	

As you can see the list box below the PP\_GetTrace button has been filled with data (trace points) and the trace has been drawn.

## IMPORTANT:

The Ladybug sensor has two traces. It has the current trace and the analysis trace. All measurements are made on the analysis trace. The trace we are looking at is the current trace. To make measurements we must first tell the system to make the current trace the analysis trace.

While this may seem a bit odd a first...other functions will allow you to pass a trace (either one you've created by hand or one that you've acquired and saved) to the analysis trace and make measurements. In this way you can decouple the collection of data from the analysis of the data.

To make the current trace the analysis trace we click the CurrTrace2AnalysisTrace button highlighted above in red. Now you can click the buttons highlighted in red below to make trace based measurements. These measurements apply to the entire trace. You can measure:

- Average Power
- Peak power
- Pulse Power
- Crest Factor
- Duty Cycle

Again, it's important to note that these measurements differ from the Gate based measurements we'll make in a few minutes.

RgrTest_Tabbed		
LB_SensorCni()     1       Idx     Adr     SN       LB_SensorLisi()     1     6     066111       V     Auto fill on select     1     1	Current Device           Model No.         LB480A           Set No.         066111           Index         1           Index         1           Address         6           Clear         Flt Opt = ST_ON           Try Opt = ST_ON           Conv. type = STAL, H           Conv. type = STAL, H           Cal 4 Wey Opt = ST_OFF           Best March Opt = ST_OFF           Wey Opt Exp = 10/27708	
Index & Address Functions   Init, CW & Pulse Mod   Meas Criteria & C	fsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options	
PP GetTracel Sweep taken at addr 6 array sized to		
Idx         Aup (dBa)         Copy           0         -20,012         0         0           1         -20,050         0         0           2         -20,026         0         0           3         -20,021         0         0           4         -20,138         0         0		
PP_GetTraceLength[]		
PP_CurrTrace2Analysis Success		
GetAnalvsi TraceLenath         -28:182 dBm           (P_GetTracePulsePow         -20:046 dBm           (P_GetTracePulsePow         -20:046 dBm           (P_GetTracePulsePow         -19:934 dBm           (PP_GetTracePicPulsePow         -19:934 dBm           (PP_GetTracePicPulsePow         -15:300 %		
		Close

RgrTest_Tabbed LB_SensorCnt() 1 Idx Adr LB_SensorList() 1 6 V Auto fill on select	Current Device Model No. 086111 Index Address	LB480A Trg Opt = ST Trg Opt = ST Rec Out Opt = Cal & Wy Opt Exp = Cal Opt Exp = Cal Opt Exp = Cal Opt Exp = Cal Opt Exp =	0N 0N ST_0N MA_H = ST_0FF t = ST_0FF t = ST_0FF 10/27/08 10/27/08	
Index & Address Functions       Init, CW & Pulse         PP_CurrTrace2Analysis       Peak Ex. Cnt       3.000 ♀         Peak Ex. Cnt       3.000 ♀       Thresh         PP_GetPeaks_Idx()       613       12.48         Count = 16       2437       2431         2431       2931       1	Mod Meas Criteria & Offsets Limits Trigg Mkr Index PP_MarkerToP -19.95 dBm -19.95 dBm -19.95 dBm -19.95 dBm -19.95 dBm -19.94 dBm -19.94 dBm	ering Recorder Out PP - Setup	PP - Get Trace PP - Markers PP PP_MarkerToFirstPk() PP_MarkerToLastPk() PP_MarkerPrevPk() PP_MarkerNextPk()	- Gates Options
PP_GetPeaks_Val()         4784           Count = 16         2991           613         1240           PP_SetMarkerMode()         1           PP_GetMarkerMode()         1	-19.93 dBn -19.94 dBn -19.94 dBn -19.95 dBn Set[Get]MarkerPo Set[Get]MarkerPo Set[Get]DeltaMarker PP_MarkerPosls	erTim	GetMarkerDeltaAmp	
				Close

Now we'll do a little work with markers then close with gate measurements.

Click the PP-Markers tab. It should look like the one above. You'll note the large "Mkr Index" in the top center of the page. This indicates the index of the current marker we're working with. Assuming you've taken a trace, we must do is ensure that we have an analysis trace. Click the PP\_GurrTrace2Analysis now (highlighted in blue).

Now we can continue by getting a list of ordered peaks, first by index, then by amplitude. Click the two buttons highlighted in red. Then examine the adjacent list boxes highlighted in yellow. These are lists of peaks. You'll see a label indicating the number of peaks in each list. Each entry in the list box has the index in the trace and the amplitude.

To use a marker select a marker index (I've used 0) highlighted in green. Then proceed by selecting the marker mode and clicking PP\_SetMarkerMode button (purple). You can test the marker mode by clicking PP\_GetMarkerMode (gray).

RgrTest_Tabbed	
LB_SensorCnt() 1 Idx Adr SN LB_SensorList() 1 6 086111 ✓ Auto fill on select Address 0	480A 086111 1 ↓ 6 ↓ Clear Flt Opt = ST_ON Trg Opt = ST_ON Conn type = ST_ON Conn type = ST_OFF Best Match Opt = ST_OFF Wty Opt Exp = 10/27/08 Cal Opt Exp = 10/27/08
Index & Address Functions   Init, CW & Pulse Mod   Meas Criteria & Offsets   Limits   Triggering	Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
PP_CurrTrace2Analysis Mkr Index 0	
Peak Ex. Crit 3.00 C Threshold -50.00 C (PP_MarkerToPk()	Idx = 4784 -19.934 d PP_MarkerToFirstPk()   Idx = 613 -19.946 dB
PP_GetPeaks_Idx()         613         -19.95         dBm         (PP_MarkerToLowestPl)           Count = 16         1248         -19.95         dBm         (PP_MarkerToLowestPl)           2437         -19.95         dBm         (PP_MarkerToLowestPl)           2991         -19.95         dBm         (PP_MarkerToLowestPl)           Prove the state of	Idx = 8886         :19.977 d         PP_MarkerToLastPk()         Idx = 9484         :19.97 d           Idx = 8886         :19.977 d         PP_MarkerPrevPk()         Idx = 8886         :19.977 d           Idx = 1794         :19.973 d         PP_MarkerNextPk()         Idx = 9484         :19.97 dB
PP_GetPeaks_Val() 4784 -19.93 dBm GetMarkerAmp	GetMarkerDeltaAmp
Count = 16 2991 -19.94 dBm 613 -19.95 dBm 1248 -19.95 dBm Set(Get)MarkerPosition	
PP_SetMarkerMode() NORMAL_MK V Success Set(Get)DeltaMarkerTir	
PP_GetMarkerMode() NORMAL_MKR = 1, (PP_MarkerPostsValid	
	Close

Now you can test the marker functionality by clicking the buttons highlighted in yellow. To make delta marker measurements you first have to set the mode to delta marker (blue green) then use the marker measurements. To make delta marker measurements use the two buttons highlighted in red. Finally, markers can be positioned by index or time. The buttons highlighted in orange are for this purpose.

Now we can move to gated measurements. Select the "PP-Gates" tab. You should see a window like the one shown below:

RgrTest_Tabbed		
LB_SensorCnt() LB_SensorList() V Auto fill on select	1 Idx Adr SN 1 6 086111	Current Device Model No. LB480A Ser No. 086111 Index 1 Address 6 Clear Hot Opt = ST_ON Conn type = SMA_H Cont Opt = ST_OFF Best Match Opt = ST_OFF Wty Opt Exp = 10/27/08 Clear
Index & Address Functions PP_CurrTrace2Analysis Peak E.X. Unt 3.90	Init, CW & Pulse Mod Meas Criteria & Offs Gate Index Threshold 50.00 C	ets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
PP_Get(Set)GateMode(		PP_GetGatePeakPower
LiatePositionIsValid		PP_GetGateAveragePo
SetGateEndPosition		PP_GetGateCrestFactor
GetGateEndPosition		PP_GetGateRiseTime
SetGateStartPosition	0 🗢	PP_GetGateFallTime
GetGateStartPosition		PP_GetGatePulseWidth
SetGateEndTime	0 🗘	PP_GetGatePRT
GetGateEndTime		PP_GetGatePRF
SetGateStartTime	0 🗘	PP_GetGateDutyCycle
GetGateStartTime		PP_GetGateDroop
		PP_GetGateOverShoot
		Close

This window is very much like the marker window. You'll You have a large up/down button labeled Gate Index. This is used to set the gate index. It applies to all buttons on this tab. Like the markers, if you haven't set the analysis trace do so now by clicking the button highlighted in blue.

Like the markers we must first turn on the gate we are interested in. Use the PP\_Set(Get)GateMode button (highlighted in red). Check the gate position. It should be invalid since we haven't yet set the position (yellow).

I'm going to start by just positioning my gate at the 1000 and 4000 index points as shown below (green). Then I'll make a number of measurements by clicking the buttons highlighted in red. Some of the measurements are meaningful (peak power, pulse power, pulse width, PRF, PRT and duty cycle. Others have numbers but are fairly meaningless because of the gates position (droop, overshoot, etc). And finally, you'll note that the rise time and fall time failed. They failed because of a poorly positioned gate.

RgrTest_Tabbed				
LB_SensorCnt() LB_SensorList() ✓ Auto fill on select	1 Idx Adr SN 1 6 086111	Current Device Model No. LB480A Ser No. 086111 Index 1 Address 6 Clear	Fit Opt = ST_ON Trg Opt = ST_ON Rec Out Opt = ST_ON Conn type = SHA_H Cal 4 Wey Opt = ST_OFF Best Match Opt = ST_OFF Wey Opt Exp = 10/27/08 Cal Opt Exp = 10/27/08	
Index & Address Functions	Init, CW & Pulse Mod Meas Criteria & C	Offsets Limits Triggering Recorder	Dut PP · Setup PP · Get Trace PP · Markers PP ·	Gates Options
PP_CurrTrace2Analysis	Gate Index	0 🗘 GateStartEndPosition	1000 🗢 4000 🗘 Succ	ess: 1000 4000
Peak Ex. Crit 3.00	Threshold -50.00 🗘	GateStartEndTime	0 🗘 0 🗘	
PP. Cot(Cot)CotoMode(	Mode/Position/Time Result/M	essage		
		ATE_ON PP_LietLiatePeakPower	Success: -19.936 dBm	
GatePositionIsValid		PP_GetGateAveragePo	Success: -28.268 dBm	
SetGateEndPosition	0 🗢	PP_GetGateCrestFactor	Success: 8.332 dB	
GetGateEndPosition		PP_GetGateRiseTime	Failed	
SetGateStartPosition	0 🗘	PP_GetGateFallTime	Failed	
GetGateStartPosition		PP_GetGatePulseWidth	Success: 9.064404 usec	
SetGateEndTime	0 🗢	PP_GetGatePRT	Success: 58,906 usec	
GetGateEndTime		PP_GetGatePRF	Success: 16.976 kz	
SetGateStartTime	0 🗢	PP_GetGateDutyCycle	Success: 15.388 %	
GetGateStartTime		PP_GetGateDroop	Success: 0.000 dB	
		PP_GetGateOverShoot	Success: 30.064 dB	
			d	
				Close

Now I'll demonstrate a rise time measurement. First I'll set the sweep time so that I can easily place the start and stop of the gate. I'm going to use 100usec. This is done by selecting the "PP-Setup" tab. Finding the PP\_SetSweepTime button, selecting a sweep time using the drop down box, then clicking the PP\_SetSweepTime button. You can check the sweep time by clicking the PP\_GetSweepTime button.

Then return the to PP-GetTrace tab and click the PP\_GetTrace button (green). Next click Draw Trace (yellow). If you've used 100usec as I have you should be able to click PP\_GetTraceLength (purple) and see a result of 4800 points. Finally, click the PP\_CurrTrace2Analysis (red). After this I'm left with a window that appears as the one shown below:

RgrTest_Tabbed			
LB_SensorCnt() LB_SensorList() ✓ Auto fill on select	1 Idx Adr SN 1 6 086111	Current Device         Model No.       LB480A         Ser No.       086111         Index       1          Address       6          Clear       Clear	
Index & Address Functions	Init, CW & Pulse Mod Meas Criteria & OI	Iffsets Limits Triggering Recorder Out PP · Setup PP · Get Trace PP · Markers PP · Gates Options	
Idx         Amp (dBm)           0         -20.127           1         -20.078           2         -20.039           3         -20.070	Copy Draw Trace		
PP_GetTraceLength() PP_CurrTrace2Analysis	4800 : Success Success		
GetAnalysisTraceLength	-28.182 dBm		
PP_GetTracePulsePow PP_GetTraceCrestFacto	-20.046 dBm 8.248 dB		
PP_GetTraceDC	15.300 %		
		Close	

Now we can return to the gates tab. I'm going to set my gate of using time instead of indexes. So I'll set my gate to (remember it's a 100usec trace) start with 50usec and end at 65usec (yellow in the picture below). Then I measure my rise time (about 60nsec). Then I reposition my gate (or you can use a different gate) for 65usec and 80usec and measure fall time. I get a fall time of about 30nsec.

Now I'll make an over shoot measurement. For this to be successful I need to position the end of the gate as starting or reference point. I'll use the back part of my pulse. I cheated and looked at the list of trace data (on the PP-GetTrace tab) and determined that my pulse ends at about index 3240. And it starts about 2800 (yellow) and then measure overshoot (red).

Positioning the start of the gate at about 2875 lets me measure droop fairly well.

RgrTest_Tabbed					
LB_SensorCnt() LB_SensorList() V Auto fill on select	1 Idx Adr SN 1 6 086111	Current Mo	t Device del No. LB480A Ser No. 086111 Index 1 0 Address 6 0 Clear	<pre>% ST_ON % rg Opt = ST_OFF % rg Opt = ST_OFF % rg Opt Exp = 10/27/08 % rg Opt Exp = 10/27/08 % rg Opt Exp = 10/27/08</pre>	
Index & Address Functions	Init, CW & Pulse Mod Meas Crite	ia & Offsets Lin	nits Triggering Recorder (	lut PP - Setup PP - Get Trace PP - Markers F	PP - Gates Options
PP_CurrTrace2Analysis	Success Gate Inde	x 0 🛟	GateStartEndPosition	2800 🗘 3240 🗘 S	uccess: 2800 3240
Peak Ex. Crit 3.00	🗘 Threshold -50.00 😂		GateStartEndTime	69 👽 🛛 89 👽 Su	ccess: 65.000 85.000
PP_Get(Set)GateMode(	Mode/Position/Time Re GATE_ON = 1 Succe	ult/Message ss: GATE_ON	PP_GetGatePeakPower	Failed	
GatePositionIsValid			PP_GetGateAveragePo	Success: Infinity dBm	
SetGateEndPosition	0 🗢		PP_GetGateCrestFactor	Failed	
GetGateEndPosition	Success: Pos = 31	9	PP_GetGateRiseTime	Success: 0.059997 use	90
SetGateStartPosition	0 🗘		PP_GetGateFallTime	Success: 0.033140 use	ec
GetGateStartPosition	Success: Pos = 23	9	PP_GetGatePulseWidth	Failed	
SetGateEndTime	0 🗢		PP_GetGatePRT	Failed	
GetGateEndTime	Success: 65.000us	c	PP_GetGatePRF	Failed	
SetGateStartTime	0 🗘		PP_GetGateDutyCycle	Failed	
GetGateStartTime	Success: 50.000 us	ec	PP_GetGateDroop	Failed	
			PP_GetGateOverShoot	Success: 0.117 dB	
					Close

Well, I think that does it. We started off by getting basic information, initializing, then making CW and pulse modulation measurements. Then we setup to get a trace, make some trace based measurements (similar to the pulse modulation measurements then went on to peaks, markers and gates. Now we need to deal with one other issue. That is offsets and response adjustments to the measured values.

The offset applies to the CW, pulse modulation and the pulse profiling measurements. I'll make some simple adjustments then we'll be done. So return to the Meas Criteria & Offsets tab. You should see a tab similar to the one below:

RgrTest_Tabbed		
LB_SensorCnt() I LB_SensorList()	1 :dx Adr SN 1 6 086111	Current Device         Model No.       LB480A         Ser No.       0861111         Index       1          Address       6          Clear       Vex p = 10/27/08
Index & Address Functions In	it, CW & Pulse Mod Meas Criteria & O	Offsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
LB_SetAntiAliasing	·	LB_SetOffsetEnabled Failed
LB_GetAntiAliasing	· ·	LB_GetOffsetEnabled ST_OFF: Success
SetAutoPulseEnabled	· ·	LB_SetOffset 30.0 🗢 Success
GetAutoPulseEnabled		LB_GetOffset 30.000 dB :Success
LB_SetPulseCriteria	0.0 🗘	LB_SetCWReference 0.0 🗢 Success
LB_GetPulseCriteria		LB_GetCWReference 0.000 DBM: Success
SetDutyCycleEnabled	· ·	LB_Get(Set)PulseReference Cancelled
GetAutoPulseEnabled		LB_SetResponseEnabled · ·
SetDutyCyclePerCent	0 🗘	LB_GetResponseEnabled
GetDutyCyclePerCent		LB_Get(Set)Response
		Close

The set and get offsets are highlighted in red. The response is highlighted in yellow. If you enter an offset (in dB) then click LB\_SetOffset followed by LB\_GetOffset you should get a sense for how offset operates. I've entered a value of 30dB. Finally, you have to enable the offset (blue). When this is done and you make a CW, pulse modulation measurement you'll see a 30dB offset.

If you want the offset to be a function of frequency you must use the response command. When you click this button (yellow) you'll see a small dialog like the one shown below:

🛃 RgrTst_SetGetR	kesponse 📃 🗖 🔀
Frequency (MHz)	Correction (dB)
5000 😂	30.0 🛟
1000	0
5000	30
Add	Delete
	OK Cancel

Enter your correct and select OK. Then enable the response (organge) and your measurements will be adjusted. I've acquired a trace after setting up for a 30dB offset. You can see the result below.

RgrTest_Tabbed		
LB_SensorCnt( ) LB_SensorList( ) ✓ Auto fill on select	1 Idx Adr SN 1 6 086111	Current Device         Model No.       LB430A         Ser No.       086111         Index       1          Address       6          Clear       Clear
Index & Address Functions	Init, CW & Pulse Mod Meas Criteria & C	ffsets Limits Triggering Recorder Out PP - Setup PP - Get Trace PP - Markers PP - Gates Options
Idx         Amp (dBn)           0         9.945           1         10.008           2         10.021           3         9.956           4         9.943	Copy Draw Trace	
PP_GetTraceLength()	4800 : Success	
PP_CurrTrace2Analysis GetAnalysisTraceLength	Success	
PP_GetTraceAvgPower	-28.182 dBm	
PP_GetTracePulsePow	-20.046 dBm	fire and antipieree all and the plants for an an and the fire alled an a fire an and an and a start and a
PP_GetTraceCrestFacto	8.248 dB	hale of the line and are not the second s
PP_GetTracePkPwr	-19.934 dBm	in a the design of the second seco
PP_GetTraceDC	15.300 %	

I hope this helps. The code for the test harness has been supplied along with the driver.