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

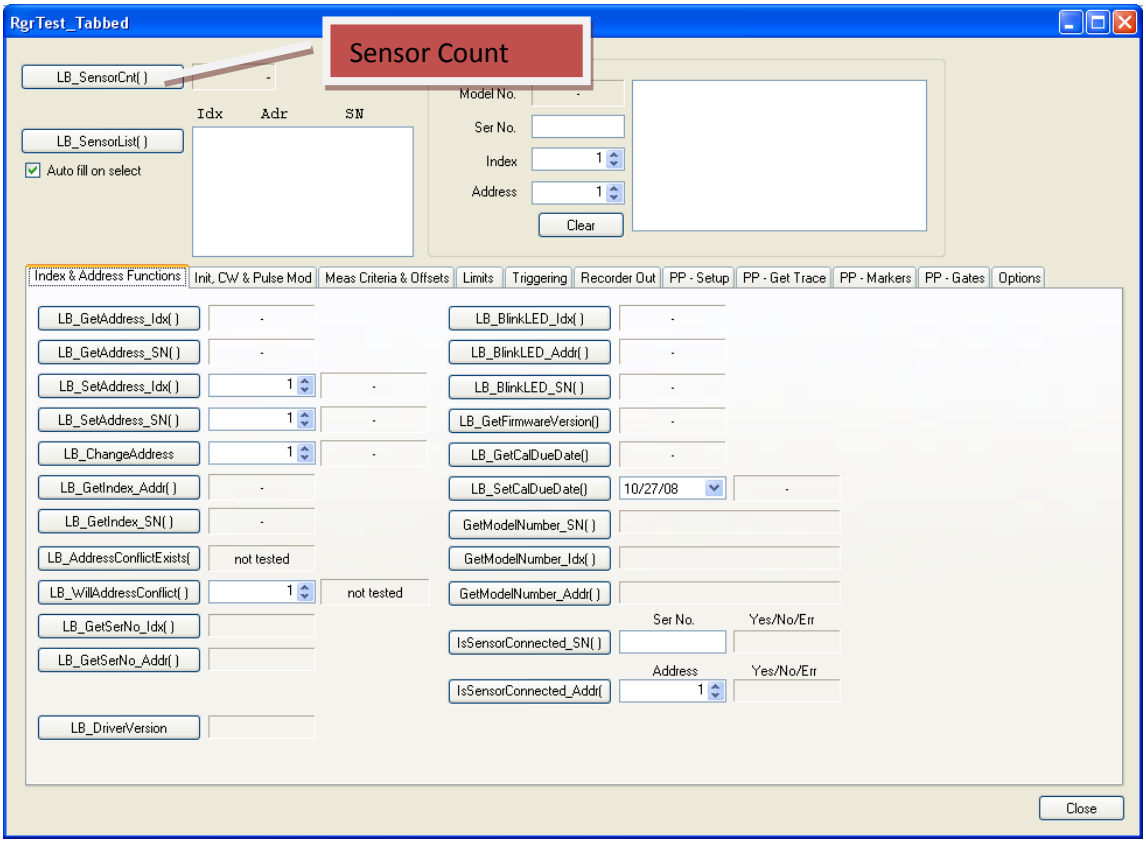
The sample code may be shipped as a zipped file. The zip file contains a single solution containing four projects. Copy and extract the files to directory of your choosing. Load the application into Visual Studio 2010 or you can start the pulse profiling test harness directly. All of the code in the 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:



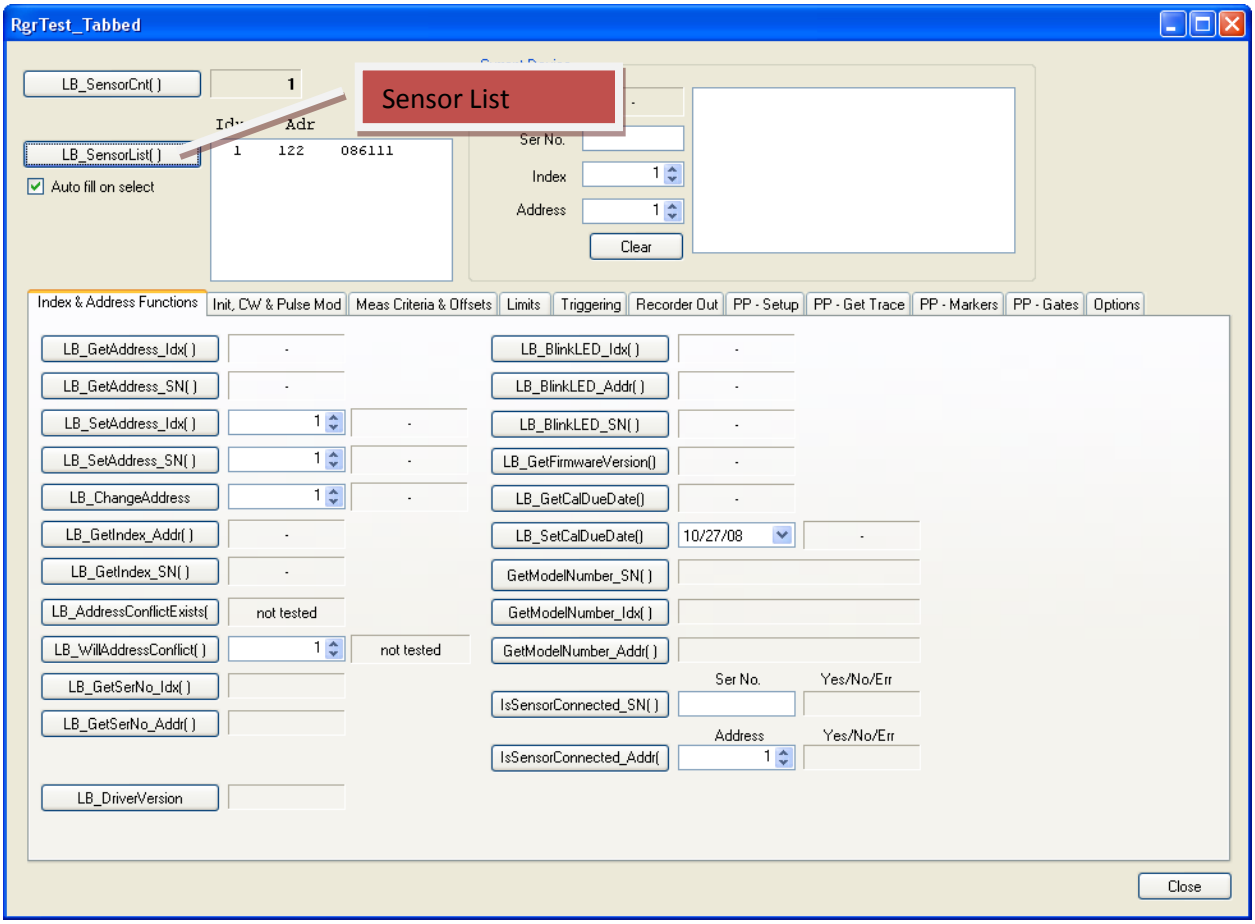
We'll start by focusing on getting basic information. Click the sensor count button. You should get a count in the adjacent label. Sensor count is a very simple call. It has the following prototype:

```
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:

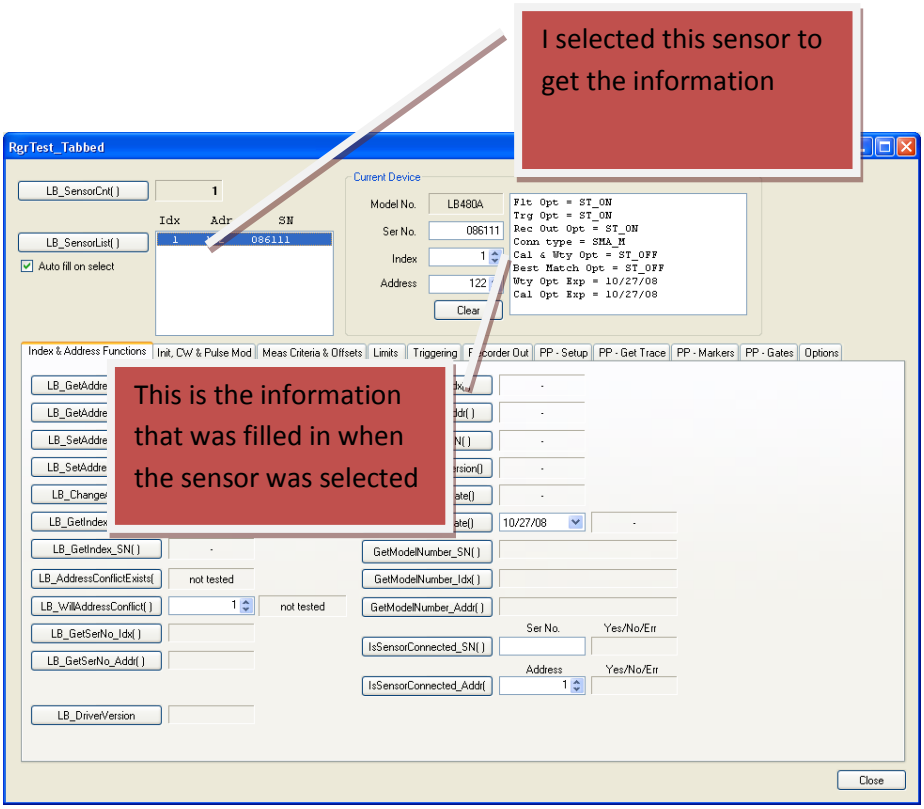
```
lblLB_SensorCount.Text = LB_API2_Declarations.LB_SensorCnt().ToString();
```

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.



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!



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);
```

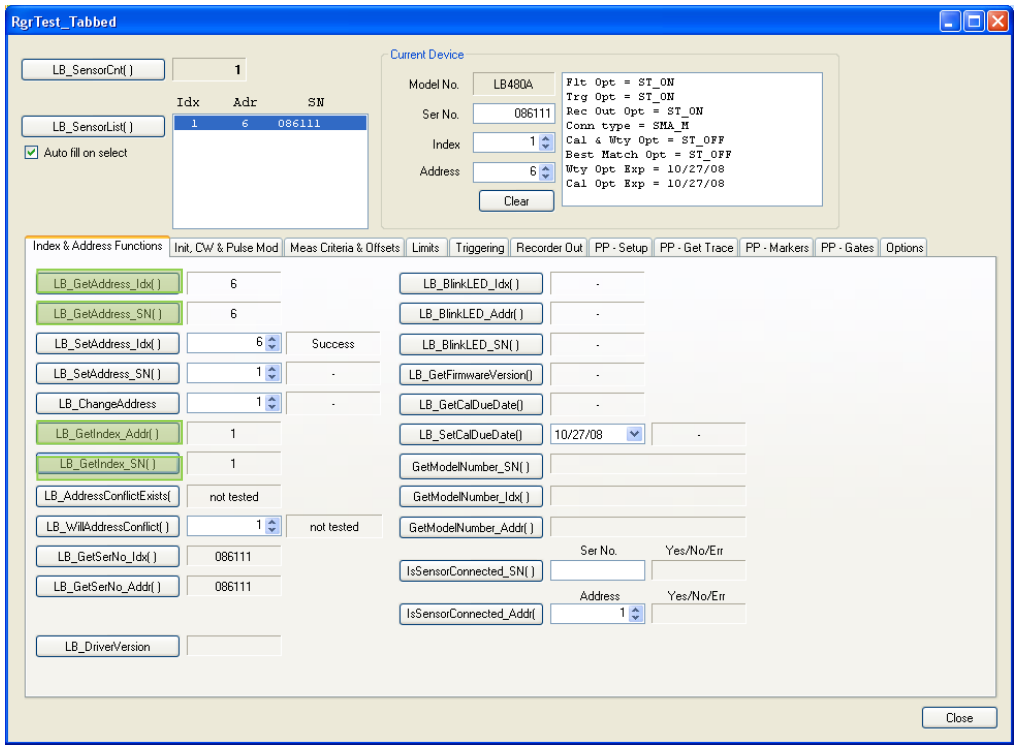
The first line gets the model number (as an enumeration) and the second one retrieves the value of the filter option. This is characteristic of most of the code. Some error checking is required. This is primarily needed to trap null pointers to serial number strings.

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. **Again, if you don't select a sensor you'll get numerous errors.** We've select a sensor so let's make a few calls:

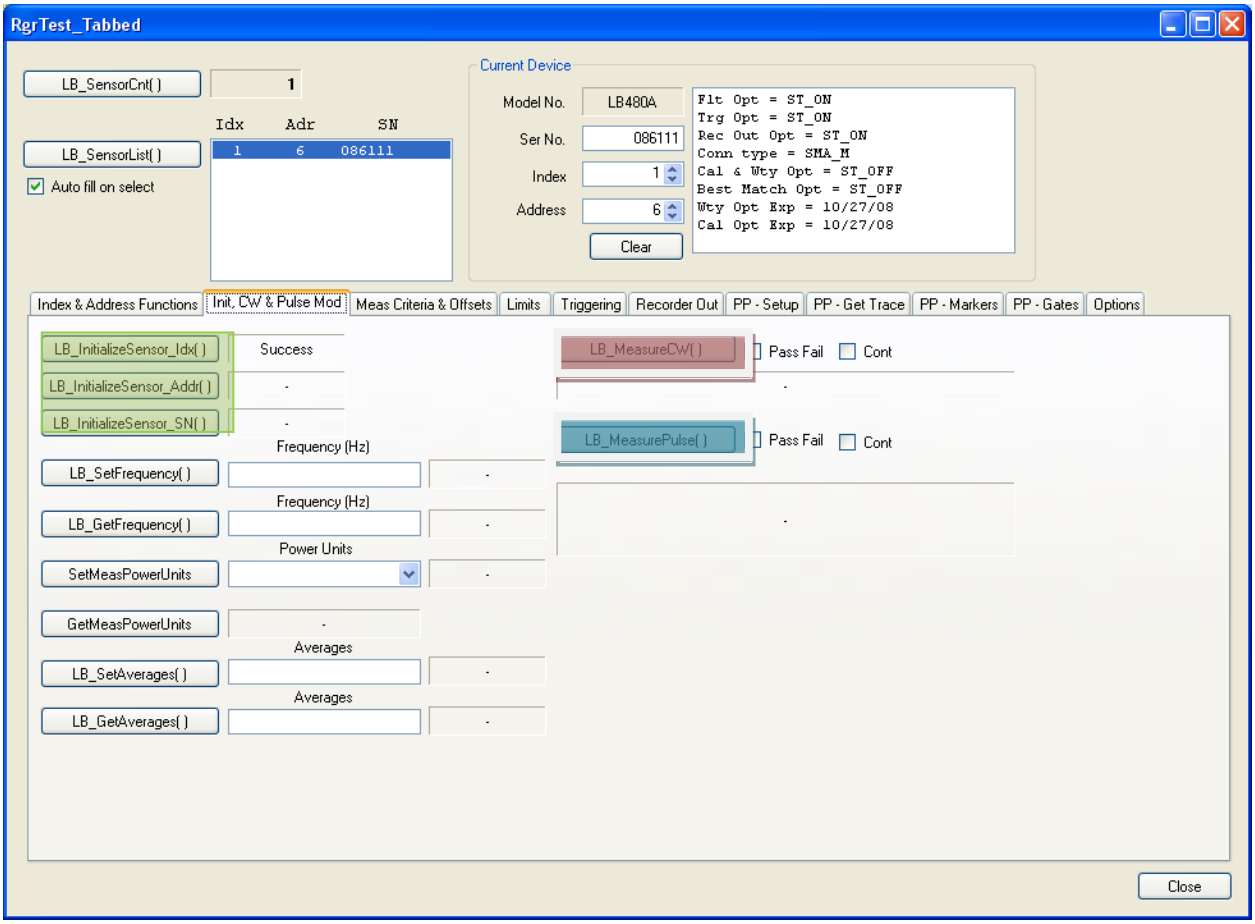
I've clicked four buttons (marked in light green). Next to each button is a value. On the two top buttons you'll note the address has been returned. On the bottom two indicated buttons you'll see the indexes have been returned.

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.



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.

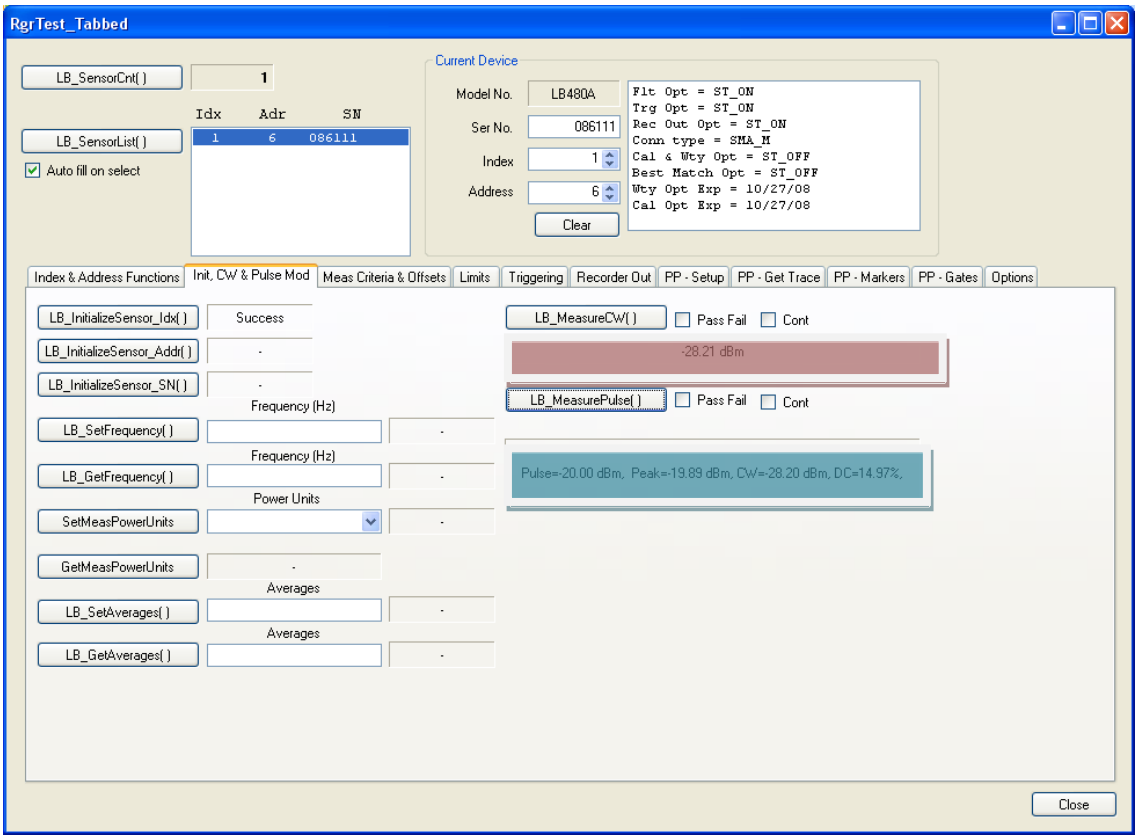


Before making any measurements we must initialize the sensor. This takes about 5 seconds. This needs to be done once when the program starts…but before any measurements are taken or any measurement settings are made. It must be done once for each sensor…but multiple calls won'

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.

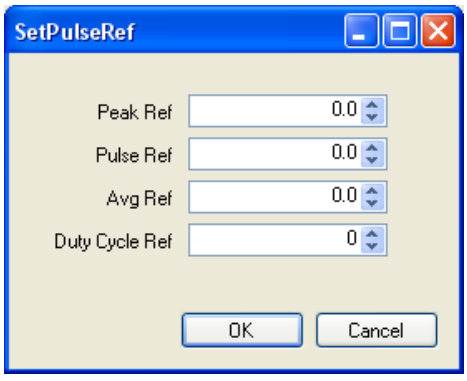
To make a pulse measurement click the button hightlighted in blue-green



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

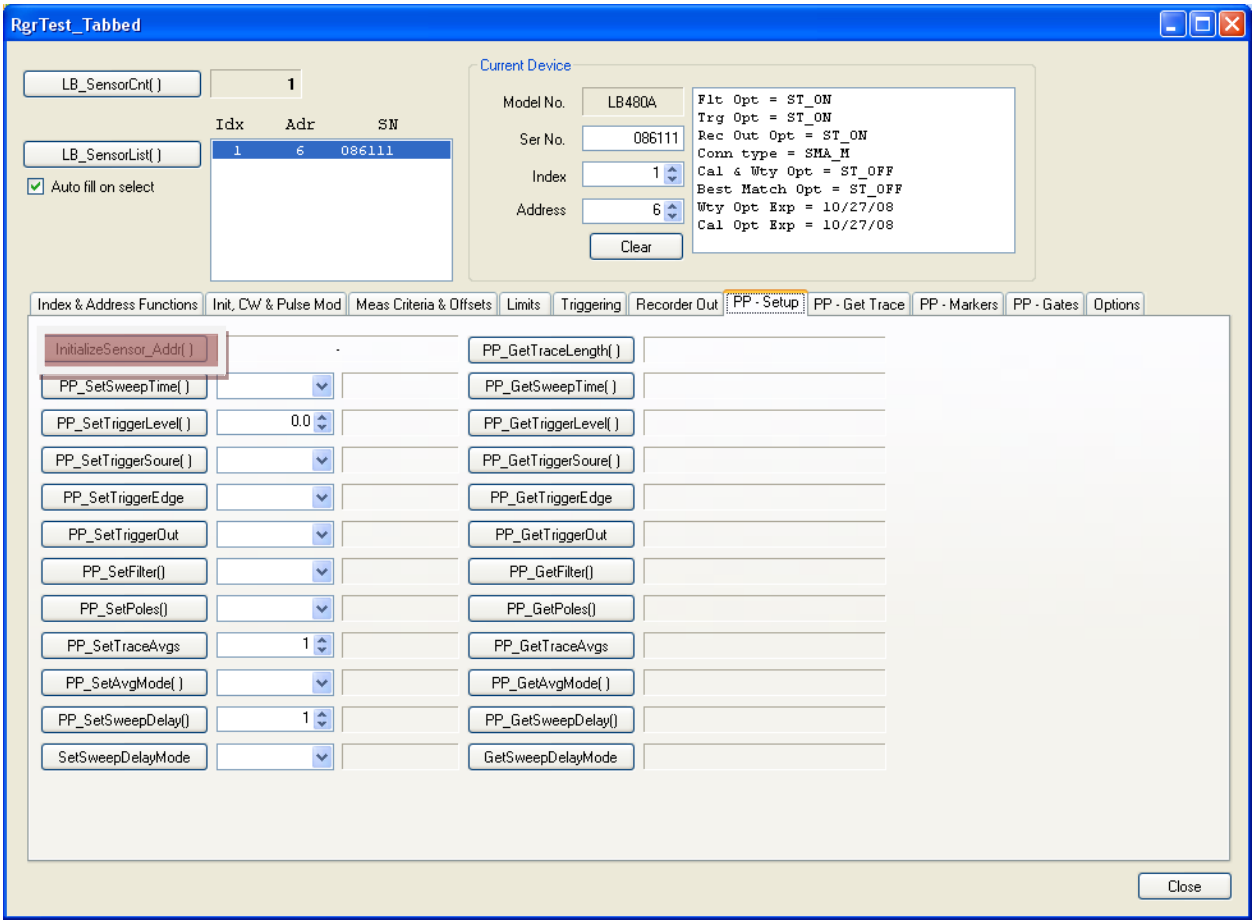
1. Select the "Meas Criteria & Offsets" tab
2. Set the CW offset by using the Set(Get)CWRefernce
3. Return to the Init, CW & Pulse measurements tab
4. Use Set(Get)MeasPowerUnits to DBREL (this enables relative measurements)
5. Now click the CW measurement. You should see measurements made relative to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

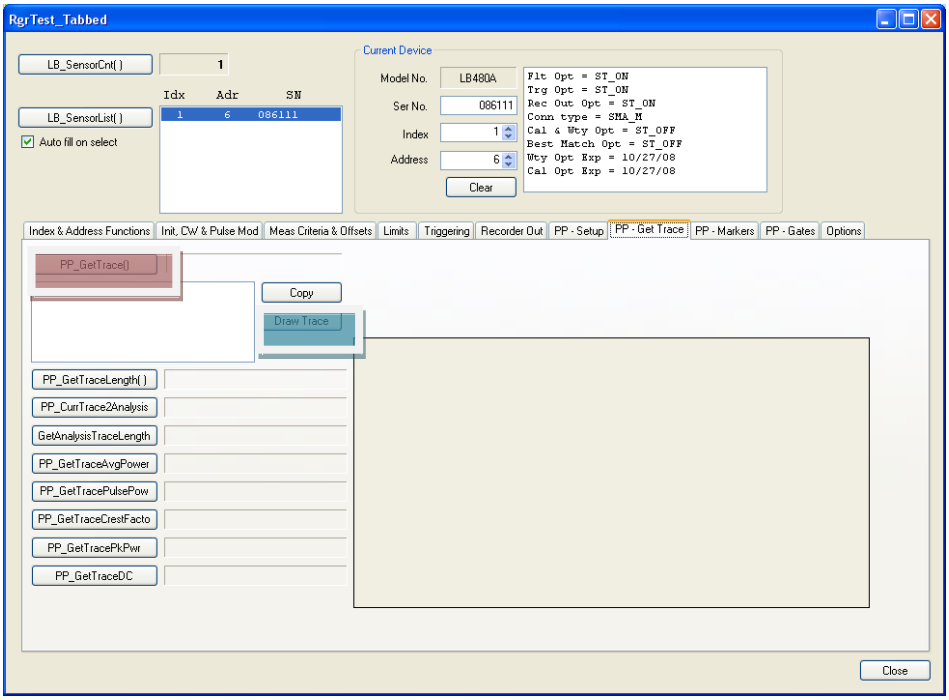


Fill dialog box with appropriate values. And then follow the CW procedure outlined above.

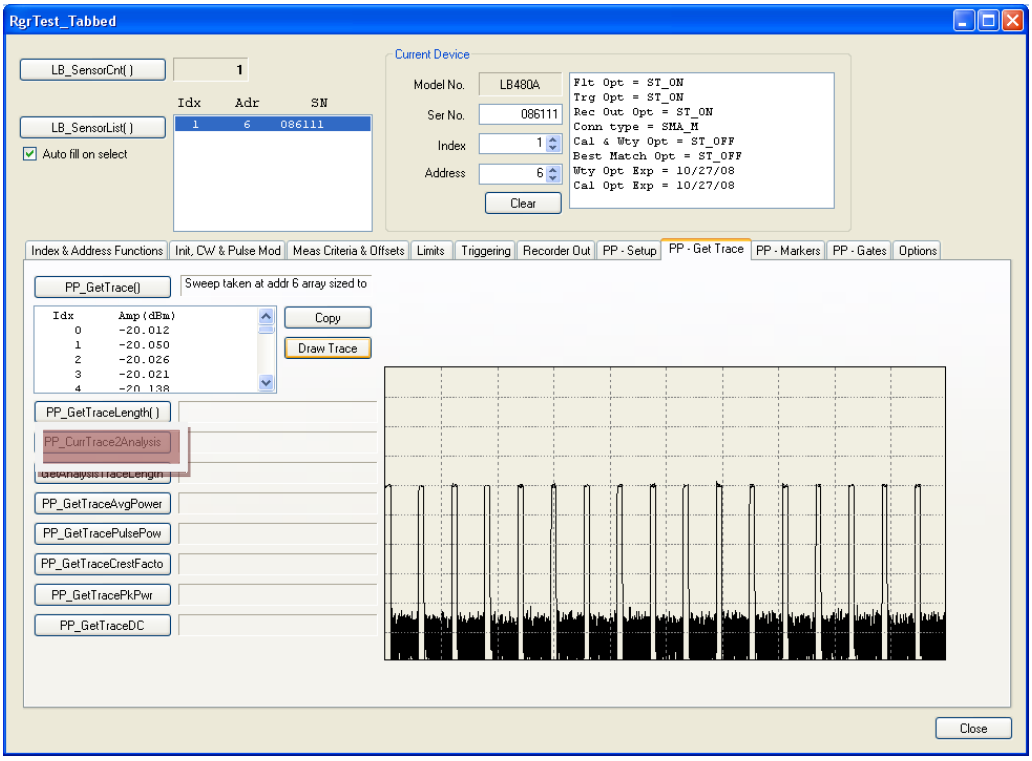
Now we'll proceed with getting a pulse profiling trace, displaying the trace and then making a few measurements with gates. Start by selecting the "PP-Setup" tab. You should see a window like the one shown below:



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".



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



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:

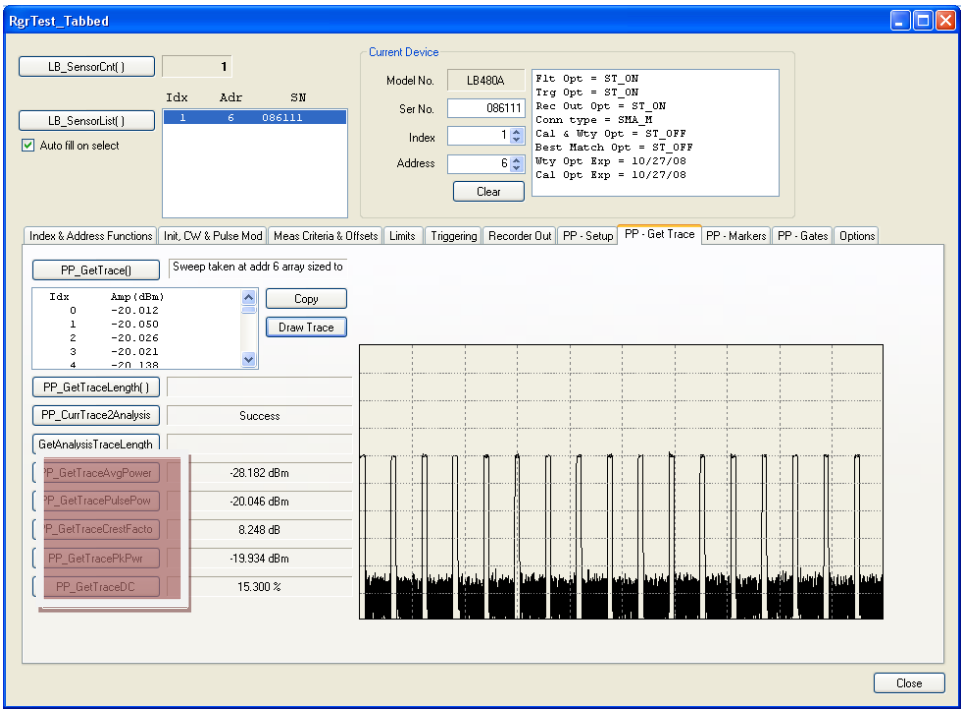
The Ladybug sensor has two traces. It has the current trace and the analysis trace. All measurements are made on the analysis trace. The trace we are looking at is the current trace. To make measurements we must first tell the system to make the current trace the analysis trace.

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.

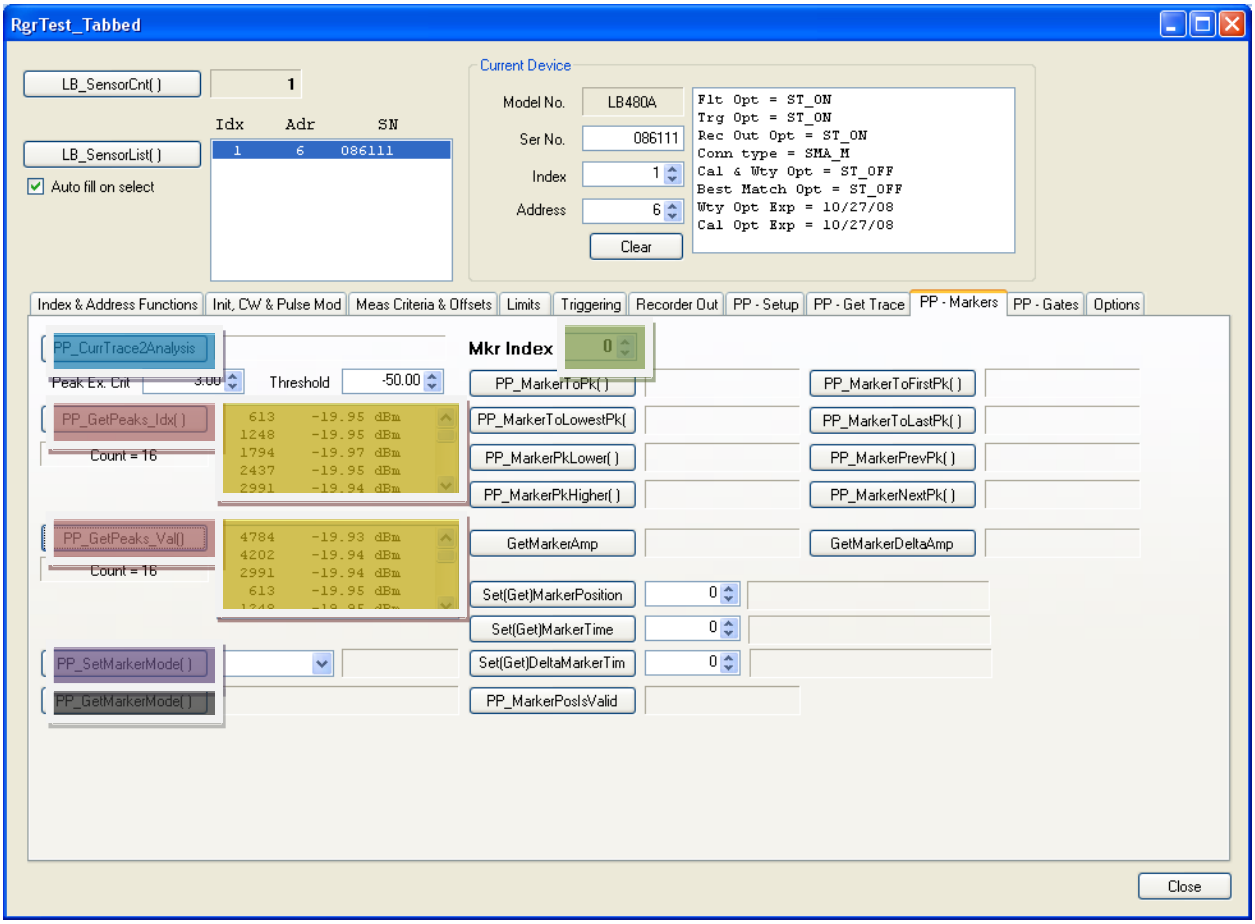
To make the current trace the analysis trace we click the CurrTrace2AnalysisTrace button highlighted above in red. Now you can click the buttons highlighted in red below to make trace based measurements. These measurements apply to the entire trace. You can measure:

- 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.



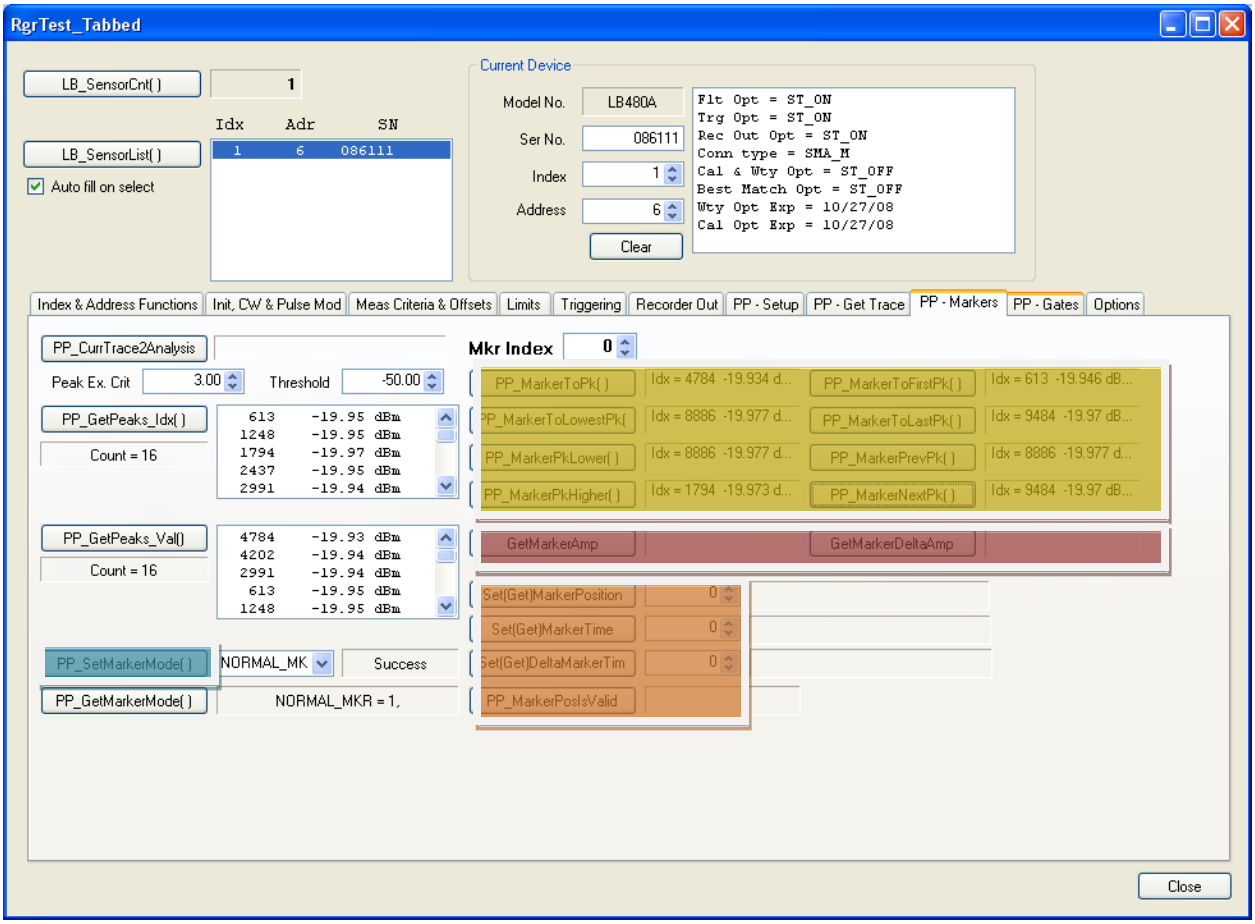
Now we'll do a little work with markers then close with gate measurements.



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).

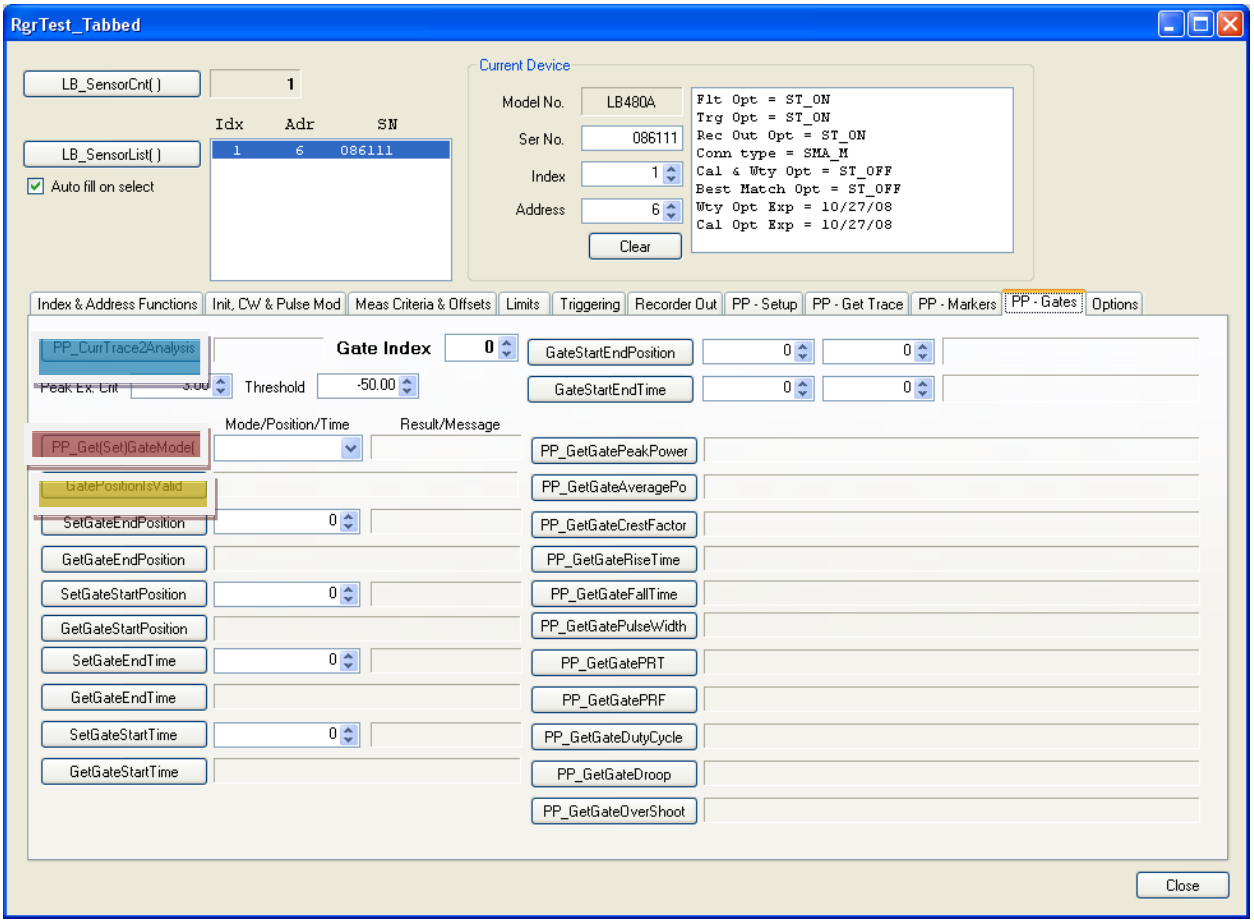
Now we can continue by getting a list of ordered peaks, first by index, then by amplitude. Click the two buttons highlighted in red. Then examine the adjacent list boxes highlighted in yellow. These are lists of peaks. You'll see a label indicating the number of peaks in each list. Each entry in the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



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.

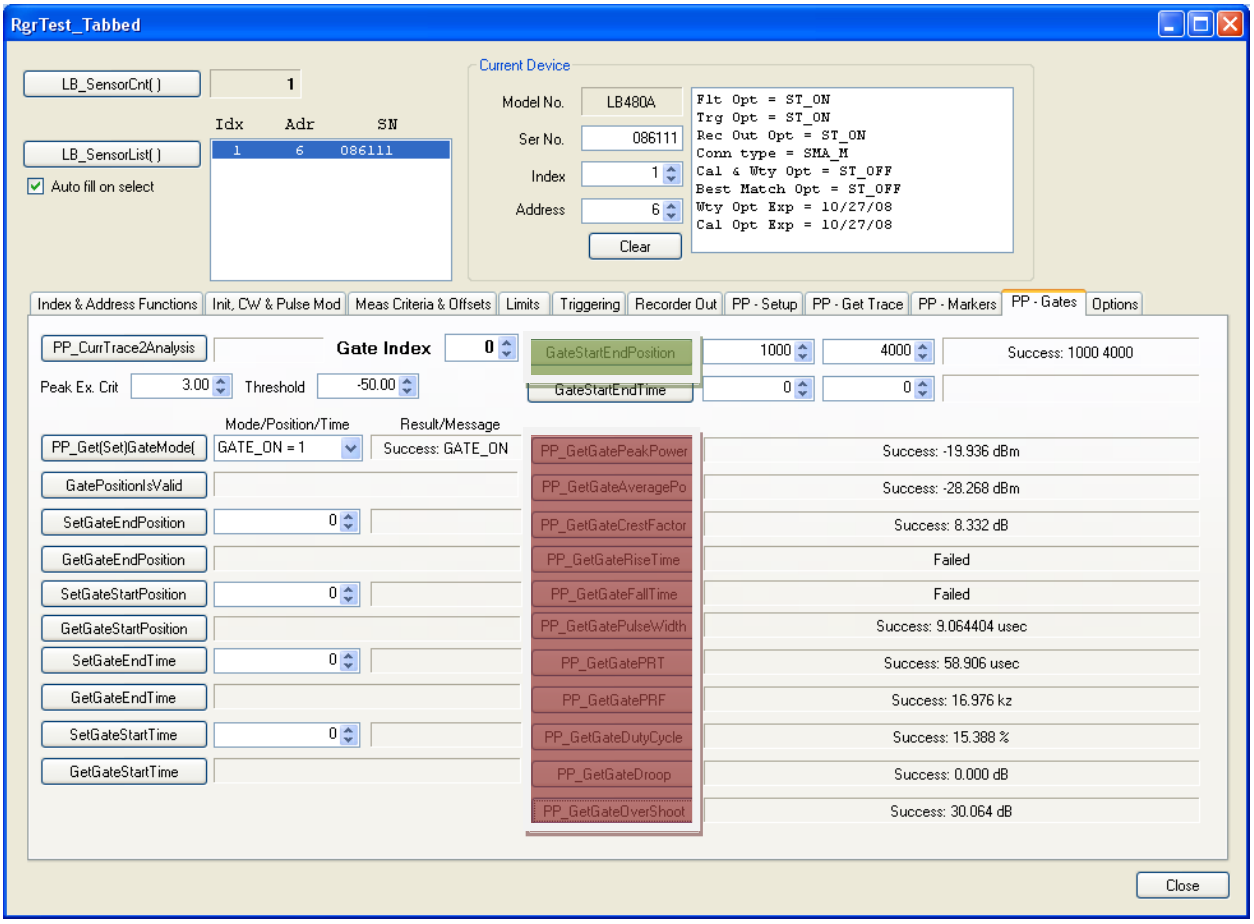
Now we can move to gated measurements. Select the “PP-Gates” tab. You should see a window like the one shown below:



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.

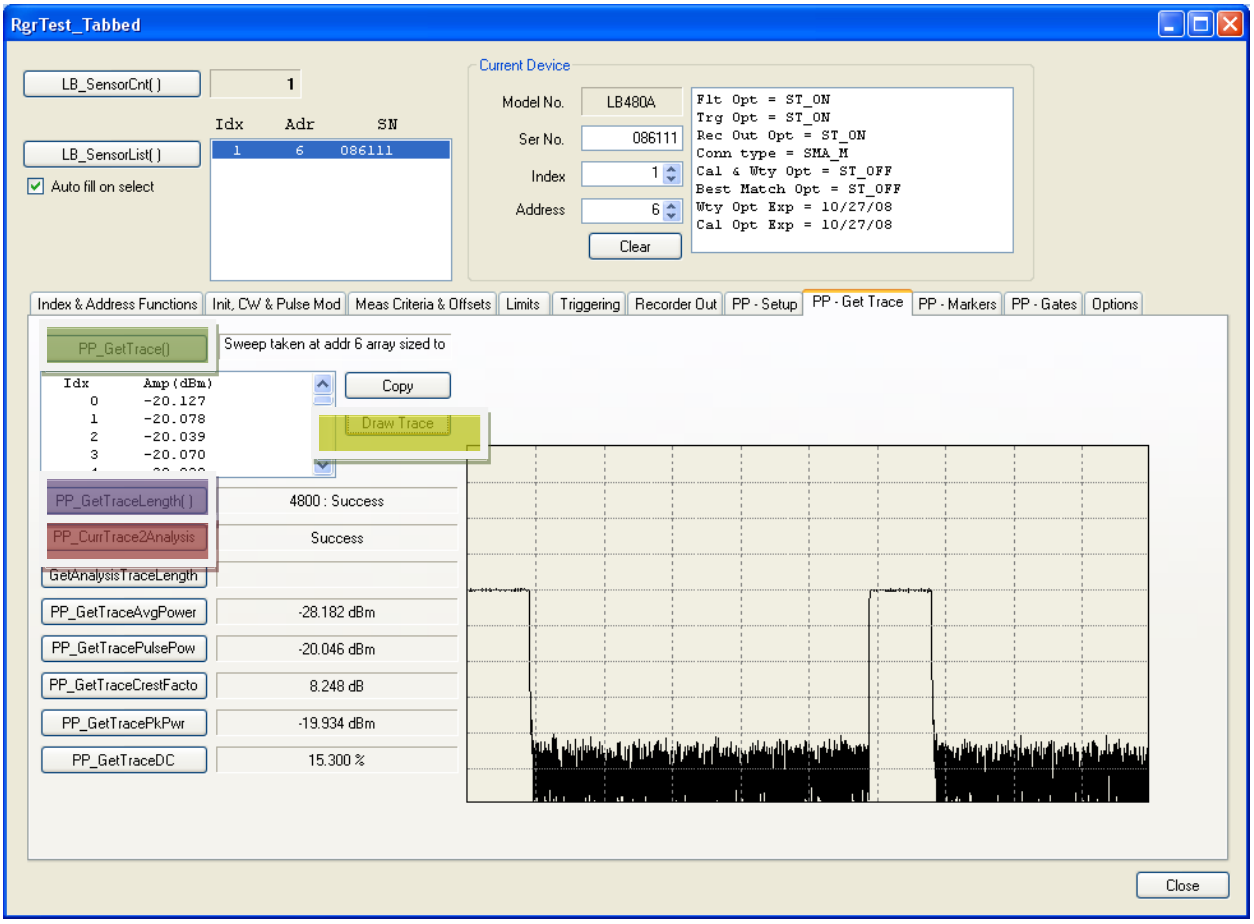
Like the markers we must first turn on the gate we are interested in. Use the PP_Set(Get)GateMode button (highlighted in red). Check the gate position. It should be invalid since we haven't yet set the position (yellow).

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.



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.

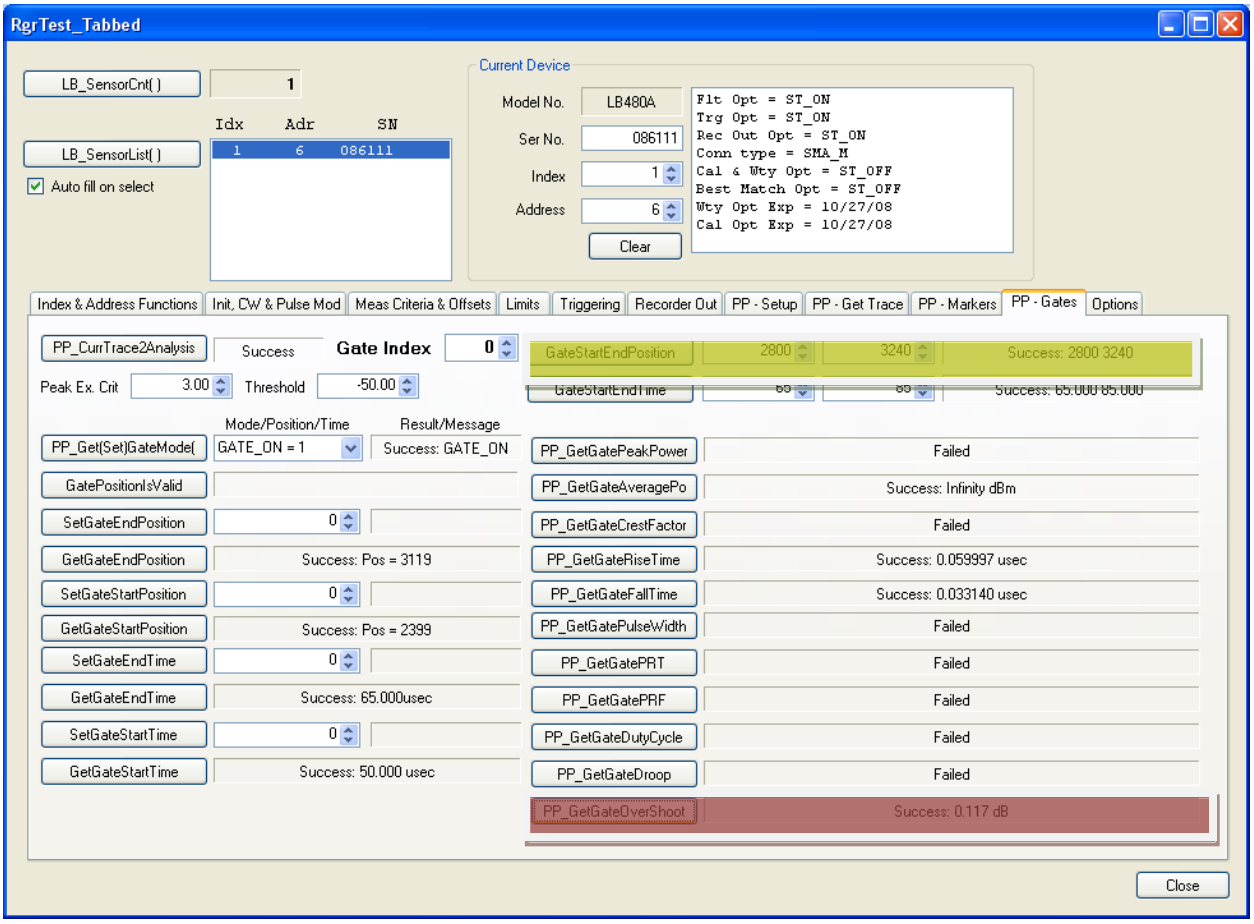
Then return the to PP-GetTrace tab and click the PP_GetTrace button (green). Next click Draw Trace (yellow). If you've used 100usec as I have you should be able to click PP_GetTraceLength (purple) and see a result of 4800 points. Finally, click the PP_CurrTrace2Analysis (red). After this I'm left with a window that appears as the one shown below:



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.

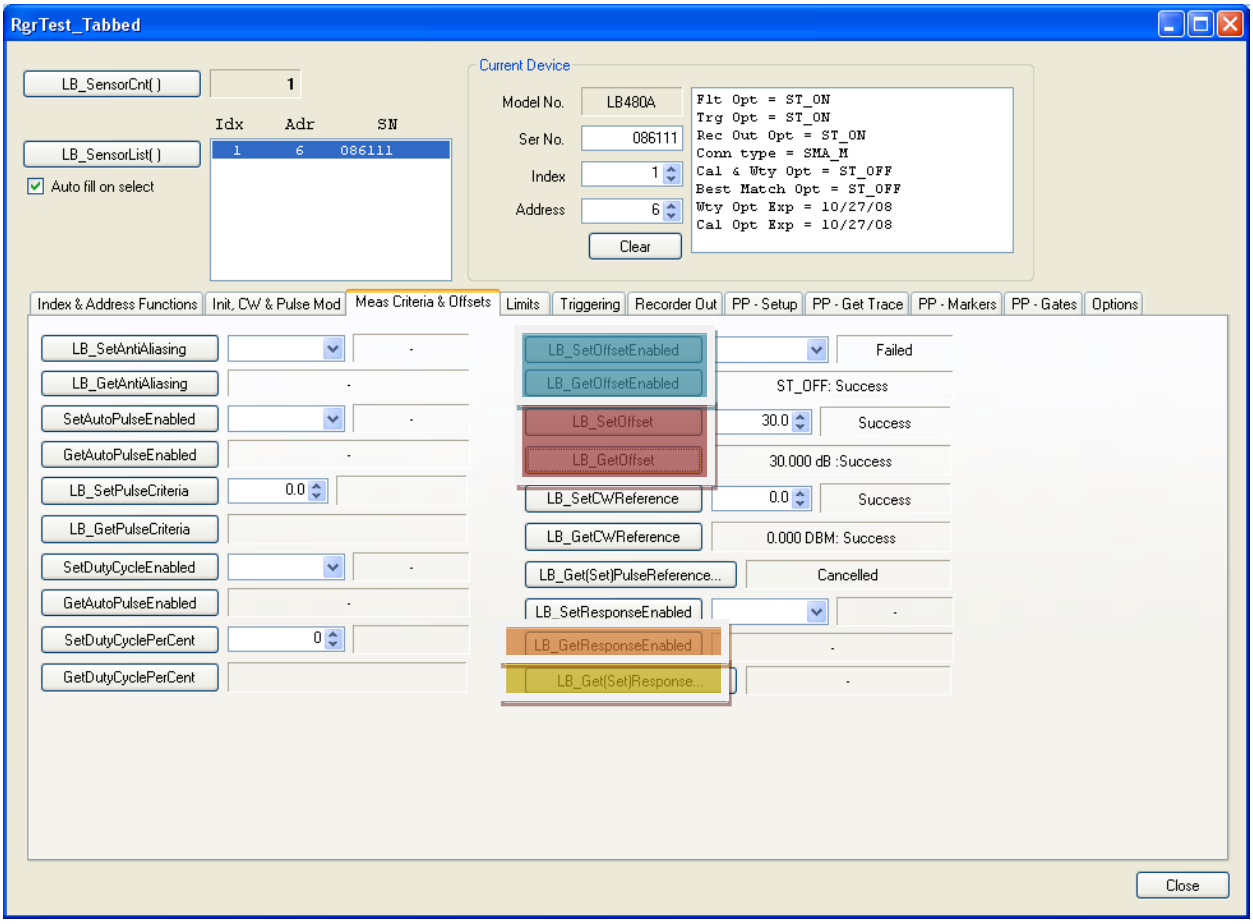
Now I'll make an over shoot measurement. For this to be successful I need to position the end of the gate as starting or reference point. I'll use the back part of my pulse. I cheated and looked at the list of trace data (on the PP-GetTrace tab) and determined that my pulse ends at about index 3240.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



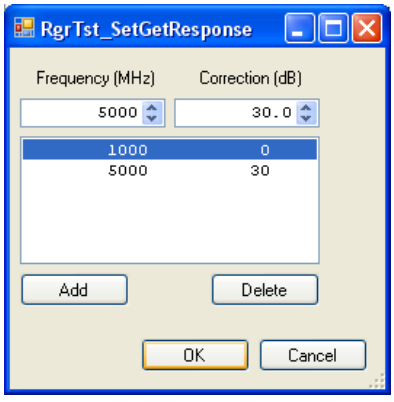
Well, I think that does it. We started off by getting basic information, initializing, then making CW and pulse modulation measurements. Then we setup to get a trace, make some trace based measurements (similar to the pulse modulation measurements then went on to peaks, markers and gates. Now we need 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:

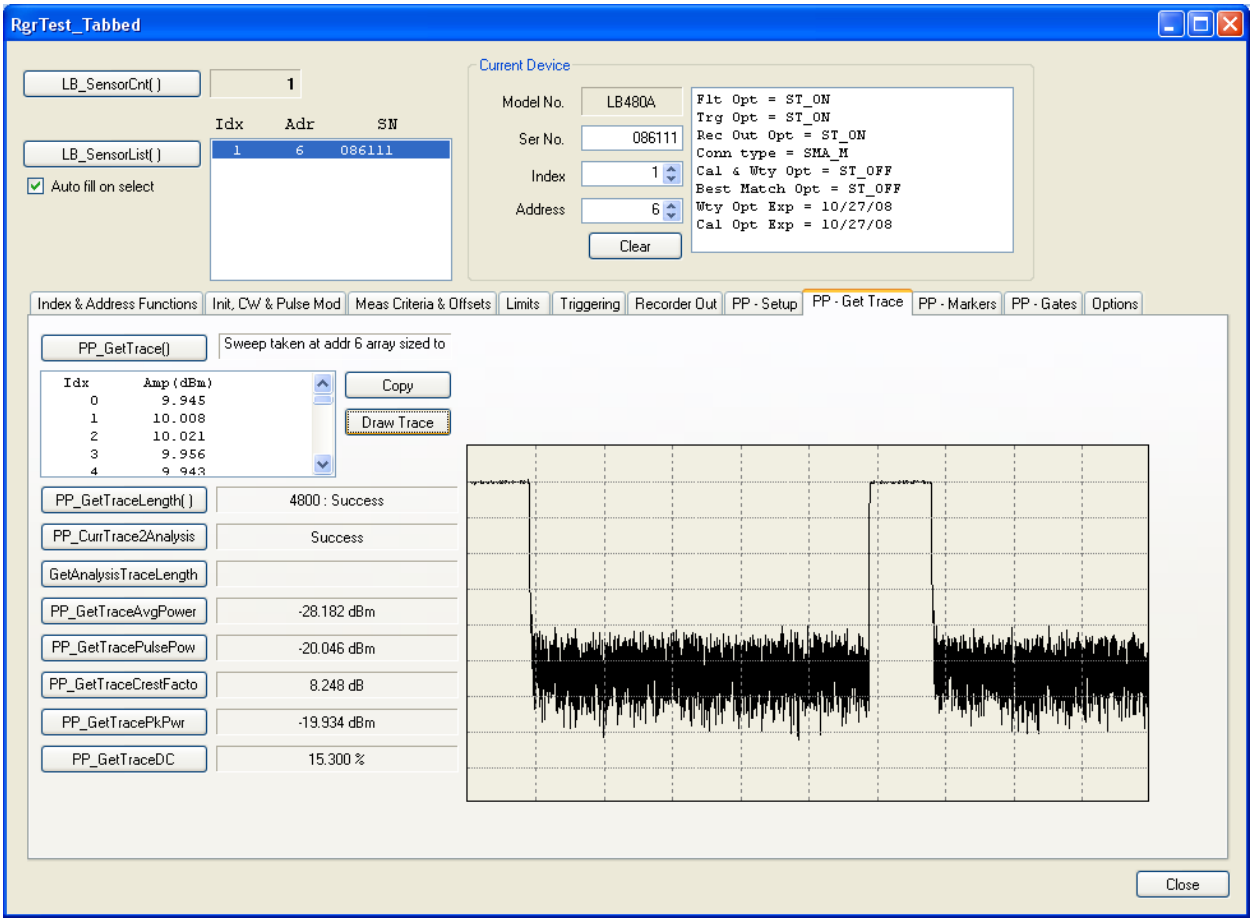


The set and get offsets are highlighted in red. The response is highlighted in yellow. If you enter an offset (in dB) then click LB_SetOffset followed by LB_GetOffset you should get a sense for how offset operates. I've entered a value of 30dB. Finally, you have to enable the offset (blue). When this 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:



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.



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