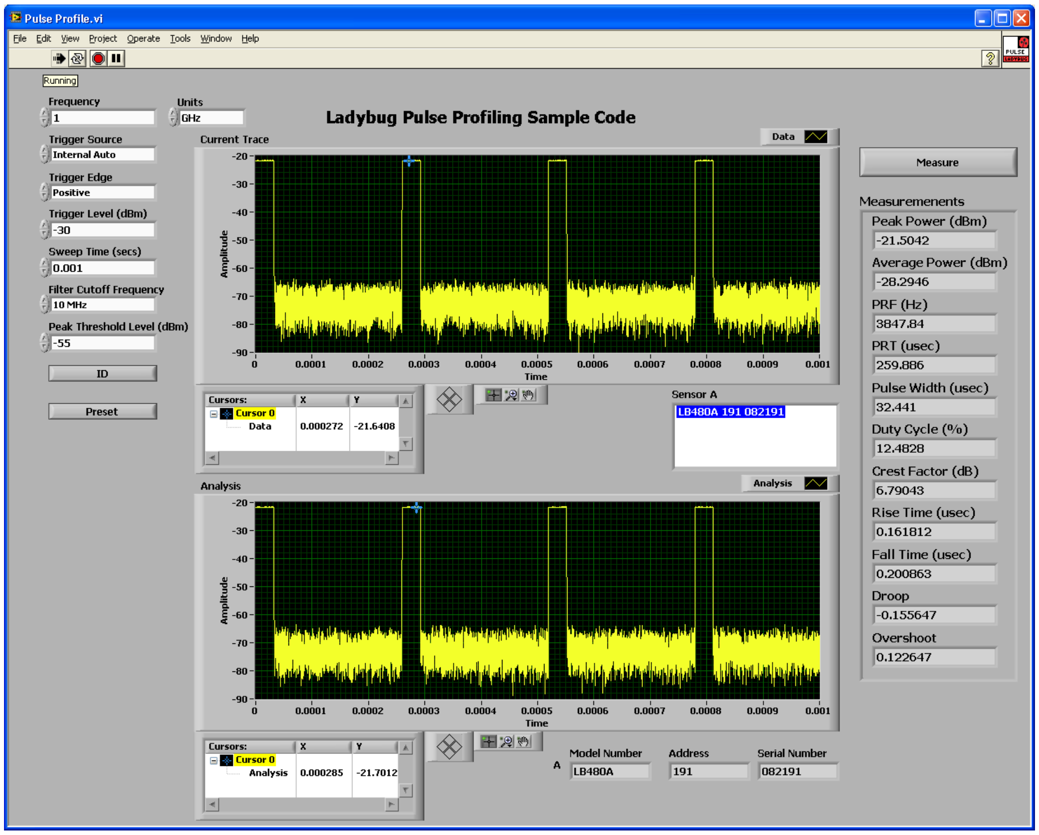
Ladybug LabVIEW Pulse Profiling Sample Code

Overview

This document introduces the user to using Ladybug power sensors for pulse profiling in a LabVIEW environment. It assumes familiarity with LabVIEW but does not require expertise. It demonstrates key functions including:

- Sensor Initialization
- Sensor Setup
- Pulse Profile Measurements
- Pulse Profile Calculations

An example test program (with source code) is included. The user interface is shown in figure 1. It will work with model 480 sensors only. . It is written as a queue-based state machine and is designed to be easy to understand. All LabVIEW code is written in LabVIEW version 8.5.1.



Frequency: 1
Units: GHz
Trigger Source: Internal Auto
Trigger Edge: Positive
Trigger Level (dBm): -30
Sweep Time (secs): 0.001
Filter Cutoff Frequency: 10 MHz
Peak Threshold Level (dBm): -55

Cursors:	X	Y
Cursor 0		
Data	0.000272	-21.6408

Sensor A: LB480A 191 082191

Cursors:	X	Y
Cursor 0		
Analysis	0.000285	-21.7012

	Model Number	Address	Serial Number
A	LB480A	191	082191

Measurements

Peak Power (dBm)	-21.5042
Average Power (dBm)	-28.2946
PRF (Hz)	3847.84
PRT (usec)	259.886
Pulse Width (usec)	32.441
Duty Cycle (%)	12.4828
Crest Factor (dB)	6.79043
Rise Time (usec)	0.161812
Fall Time (usec)	0.200863
Droop	-0.155647
Overshoot	0.122647

Figure 1: Run-time view of the Measurement Panel. Note that both the current trace and analysis trace are shown.

Exercising the Pulse Profiling Code

The pulse profiling code is an interactive application designed as an introduction to interfacing the Ladybug DLL with LabVIEW. To start it, simply click on the ‘Play’ button.

The software has many capabilities and is easy to use. A few of the settings and techniques are shown here to assist you in getting started.

The overall code is shown in the following figure (note that not all cases of the case structures are shown).

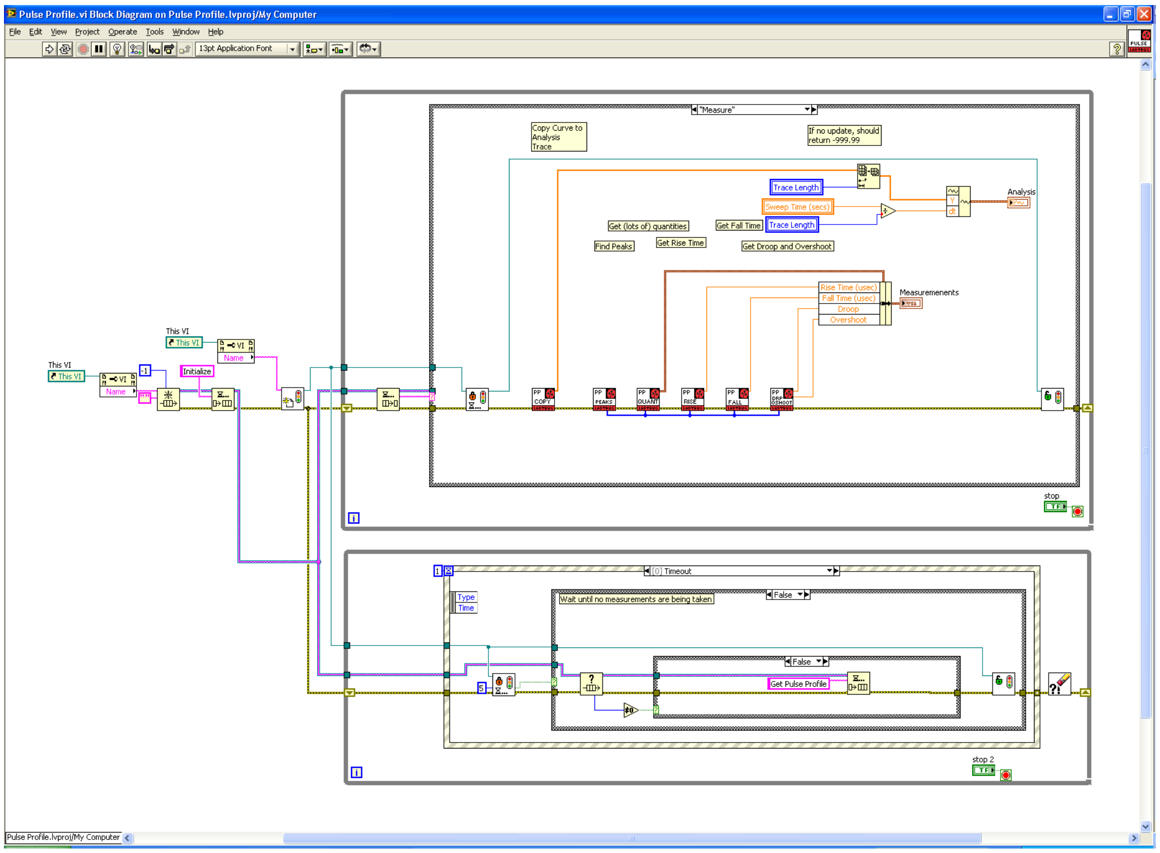


Figure 2: Detail view of the pulse profiling code. The upper case statement is executed when the “Measure” button is pressed.

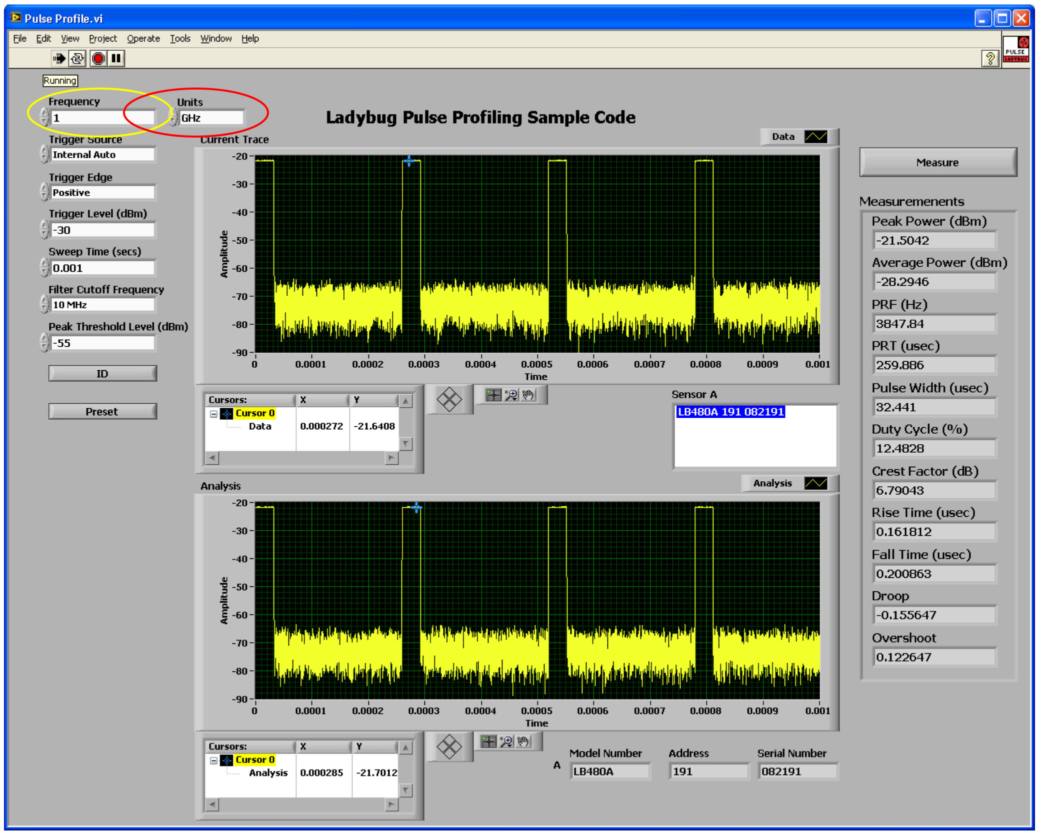
Software Installation

To install the software, simply run the installer and it will copy all files to the appropriate locations.

Setting the Frequency

To set the frequency, change the value in the frequency control (shown below in yellow). You may also change the units (shown in red) as necessary. These changes will take effect immediately.

Note that this will only change the frequency for the active sensor. If more than one sensor is connected, the others will not change.



Pulse Profile.vi
File Edit View Project Operate Tools Window Help
Running
Ladybug Pulse Profiling Sample Code
Current Trace
Data
Amplitude
Time
ID
Preset
Measure
Analysis

Frequency: 1
Units: GHz
Trigger Source: Internal Auto
Trigger Edge: Positive
Trigger Level (dBm): -30
Sweep Time (secs): 0.001
Filter Cutoff Frequency: 10 MHz
Peak Threshold Level (dBm): -55

Cursors:	X	Y
Cursor 0 Data	0.000272	-21.6408

Sensor A: LB480A 191 082191

Cursors:	X	Y
Cursor 0 Analysis	0.000285	-21.7012

	Model Number	Address	Serial Number
A	LB480A	191	082191

Measurements
Peak Power (dBm): -21.5042
Average Power (dBm): -28.2946
PRF (Hz): 3847.84
PRT (usec): 259.886
Pulse Width (usec): 32.441
Duty Cycle (%): 12.4828
Crest Factor (dB): 6.79043
Rise Time (usec): 0.161812
Fall Time (usec): 0.200863
Droop: -0.155647
Overshoot: 0.122647

Figure 3: Setting the frequency is easily done from the user interface.

In terms of the code, changing the frequency triggers the event structure which places a "Set Frequency" message in the queue. This results in the "Set Frequency" case being executed. These sections of the code are shown in the following figure.

The initial value of the frequency is read from the sensor when the software starts to execute using the "Read Setup" case.

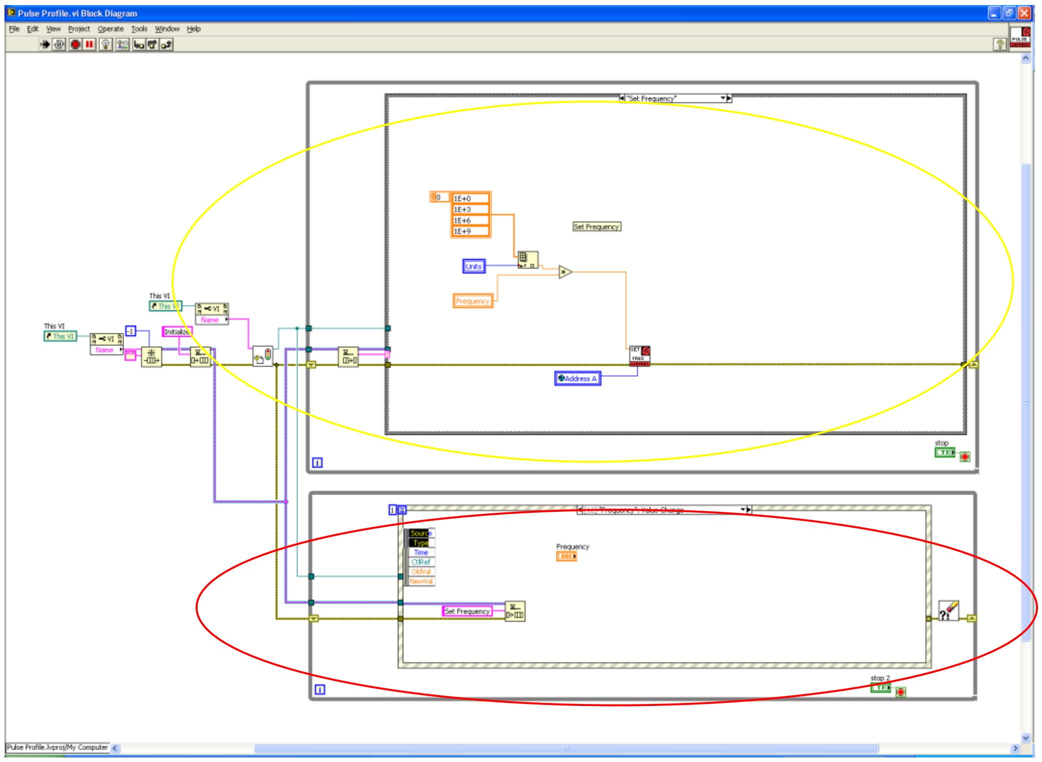

Pulse Profile.vi Block Diagram
File Edit View Project Operate Tools Window Help
"Set Frequency"
Set Frequency
Units
Frequency
Address A
This VI
Initialize
"Frequency": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Set Frequency
stop
stop 2
Pulse Profile.lvproj/My Computer

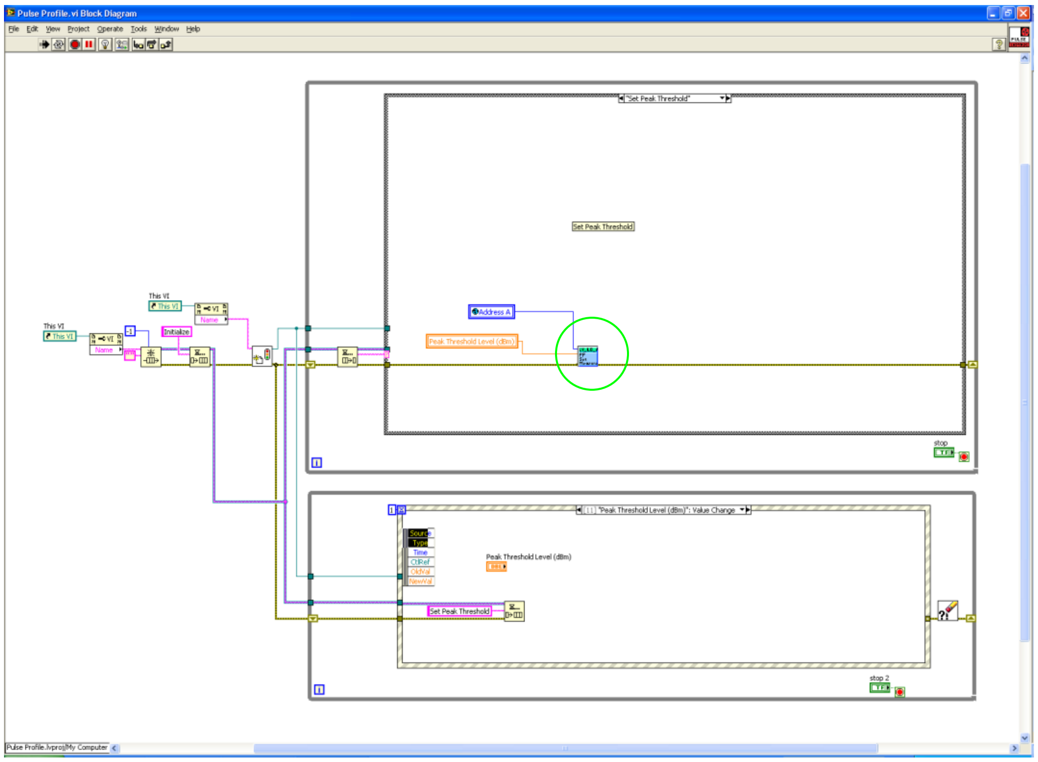
Figure 4: The set frequency operation starts in the event structure (in red) and finishes with a call to the Set Frequency VI (in yellow). Note that units must be considered.

Similar Controls

Many of the other controls execute nearly identically, including:

- Trigger Source
- Trigger Edge
- Trigger Level
- Sweep Time
- Filter Cutoff Frequency
- Peak Threshold Level
- ID
- Preset

Each of these triggers the event structure which puts a message in the queue. The message is decoded in the upper case statement and executed. The primary difference for each of these is the subVI's that they call. Some use library routines such as the Set Frequency VI shown above. Most, however, use the low-level VI's described in the Ladybug LabVIEW Sample Code documentation. For example, changing the peak threshold on the front panel will execute code as shown in the following figure.



Pulse Profile.vi Block Diagram
File Edit View Project Operate Tools Window Help
"Set Peak Threshold"
Set Peak Threshold
This VI
Name
Initialize
Address A
Peak Threshold Level (dBm)
stop
"Peak Threshold Level (dBm)": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Set Peak Threshold
stop 2
Pulse Profile.lvproj/My Computer

Figure 5: The peak threshold is set using a low-level DLL call (shown in green). These low-level calls are always blue with a green stripe at the top.

Measurements

When the “Measure” button is pressed, several subVI’s are called to determine all of the values on the right-hand side of the screen. This section assumes that the reader is familiar with edge detection and the gating abilities of the power sensor. The code is shown in the figure below.

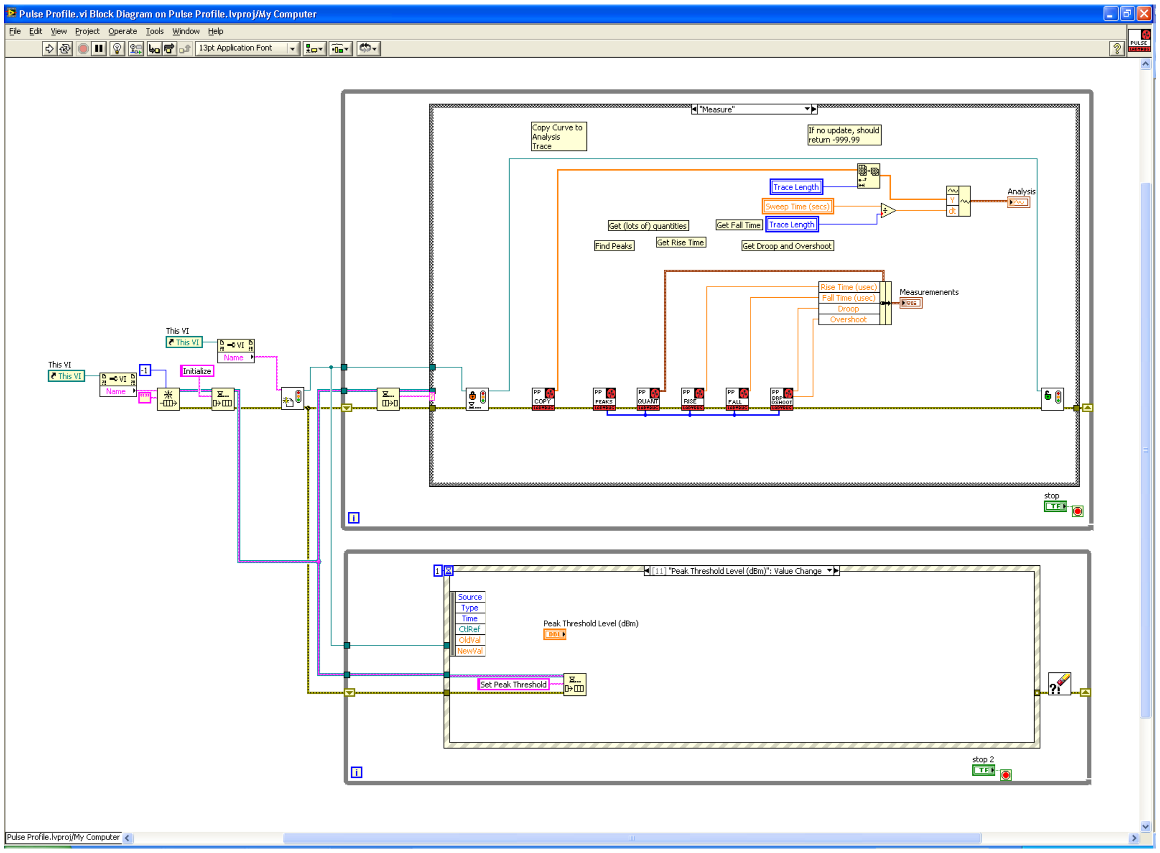


Figure 6: These measurement routines use the gate library routines to perform all of the math. The important settings for the user are the start and stop indices of the gate.

First, the instrument is reserved by acquiring the semaphore. This will ensure that no other operations will be performed on the instrument while the measurements are made.

Next, the current trace (the top trace on the user interface) is copied to the analysis trace (the bottom trace on the user interface). This step is crucial as all calculations are performed on the analysis trace.

Following this, the locations of the peaks are determined. The peaks should be roughly in the center of each pulse. The indices of the peaks are fed to each of the calculating VI’s. The array that is returned by the library function is larger than necessary, but the peaks have positive indices. Therefore, any array element less than 0 is ignored.

Next, the peak power, average power, PRF, PRT, pulse width, duty cycle, and crest factor are calculated in the Measure PP Quantities VI. All of these are gated measurements and use the same gate settings. Note that, in this example, we will always use gate 1 only. This is somewhat complicated.

First, the peak edges for the first two peaks are found using the library function Get_Pulse_Edges_Position. Each pulse edge can be thought of as a left or right edge, and the peaks are numbered 0 and 1. Therefore, the pulse width would be the index of the right side of the pulse minus the index of the left side of the pulse. To simplify matters, the formulas below refer to the indices, so left[0] would be the index of the left side of pulse 0.

To completely contain two pulses (with a little to spare), we will put the gate positions at the following locations. The start of the gate will be at:

left[0]-0.05*(right[0]-left[0])

This is the index of the starting edge of the first pulse minus 5% of the pulse width (plus one). The end of the gate will be at:

right[1]+0.05*(right[1]-left[1])+1

This is the index of the ending edge of the second pulse plus 5% of the pulse width.

Once gate 1 is set to this position, all of the quantities can be read directly using the appropriate library functions.
Similarly, the rise time is also calculated in the Get Rise Time VI as a gated measurement. In this case, we only need the first half of a pulse, so the gate is set to a starting index of:

left[0] - 5% of pulse width - 10

and an ending index of:

left[0]+(right[0]-left[0])/2

Next, the fall time is calculated using the second half of the pulse with the starting index of the gate set to:

left[0]+(right[0]-left[0])/2

and the ending index set to:

right[0] + 5% of pulse width + 10 (just to be sure)

Finally, the droop and overshoot are calculated by setting the gate to left[0]+2 and right[0]-2 – essentially, only the pulse.

Selecting the Power Sensor

The next two userobjects (Get Sensor List and Select Sensor) retrieve the list of all available sensors, filter out those that will not work (non-480 sensors), and allows the user to select the sensor.

The only item worth noting here that may be other wise confusing is that 480 type sensors have a ModelNumber of 3. In reality, this is the enumerated value of the model number.

Selecting the power sensor will set the variable Address_A.

Once a power sensor is selected, the LED on the power sensor will blink several times. This is controlled by the Blink userobject.

Initializing the Power Sensor

Prior to use, the power sensor must be initialized. This is performed in the "Initialize Sensor" UserObject, as shown in the following figure.

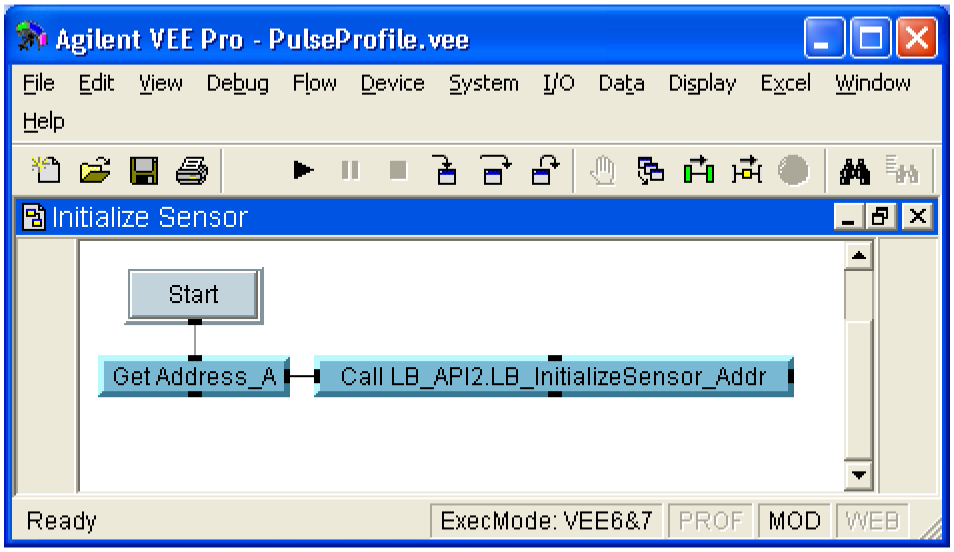

Agilent VEE Pro - PulseProfile.vee
File Edit View Debug Flow Device System I/O Data Display Excel Window Help
Initialize Sensor
Start
Get Address_A
Call LB_API2.LB_InitializeSensor_Addr
Ready
ExecMode: VEE6&7
PROF
MOD
WEB

Figure 7: Sensor initialization using the address. Note that this is the typical format of the library calls, although a formula call could also be used.

Setting Up a Pulse Profile Measurement

Before taking measurements, certain default values are set and the relevant settings of the sensor are read. All of this occurs in the Setup Measurement UserObject, shown below.

First, the sweep time is set to 1 millisecond, the trigger source is set to internal continuous (0), and the gate mode is turned on.

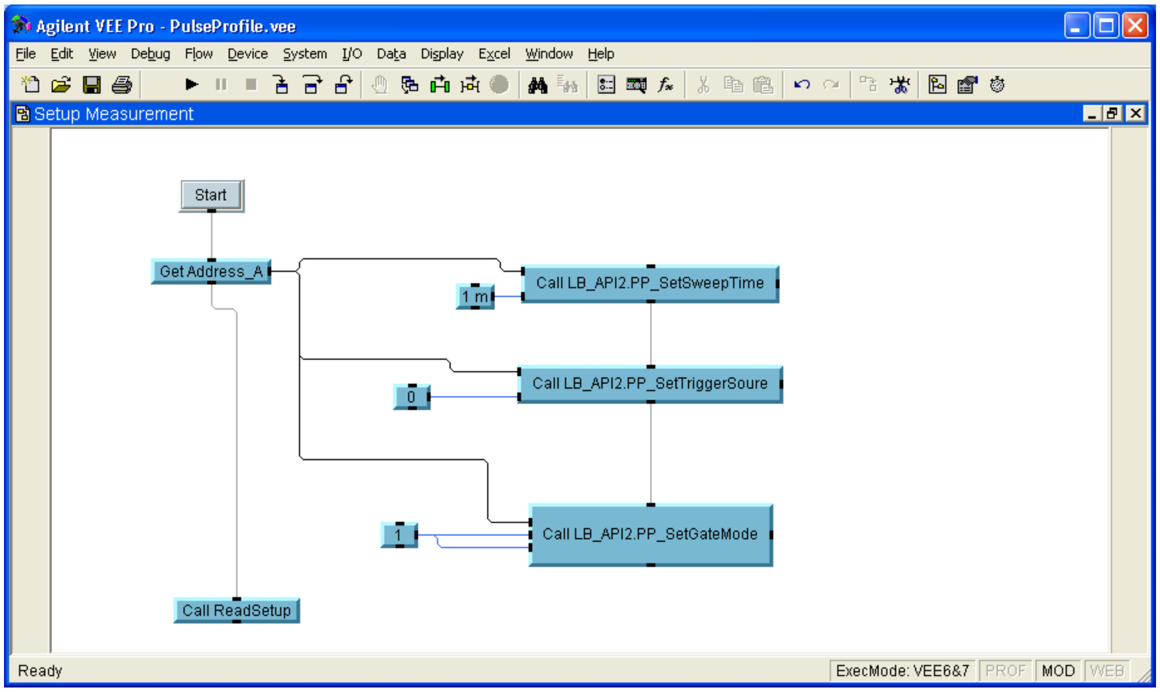

Agilent VEE Pro - PulseProfile.vee
File Edit View Debug Flow Device System I/O Data Display Excel Window Help
Setup Measurement
Start
Get Address_A
1 m
Call LB_API2.PP_SetSweepTime
0
Call LB_API2.PP_SetTriggerSoure
1
Call LB_API2.PP_SetGateMode
Call ReadSetup
Ready
ExecMode: VEE6&7
PROF
MOD
WEB

Figure 8: The Setup Measurement UserObject sets some of the default values of the sensor. Note that the sequence out pin of Get Address_A prevents the settings from being read until all three default values are set.

Next, the ReadSetup UserFunction is called. This UserFunction is used frequently to retrieve the sensor settings. It returns the trace length, the sweep time, the trigger edge, the trigger level, the trigger source, and the filter cutoff frequency. All of these are stored in variables.

The Measurement Panel

The panel view of the measurement panel UserObject was shown in figure 1. The detail view is shown below.

Essentially, there are two loops. The outer loop waits for any events to occur (such as a button being pressed). The inner loop retrieves the latest trace and displays it on the upper graph (Current Trace).

For example, if the user changes the trigger edge enum, the new trigger edge will be set and the loop will start again. Note that, at the beginning of the next iteration, the loop will read back the trigger edge and update the front panel.

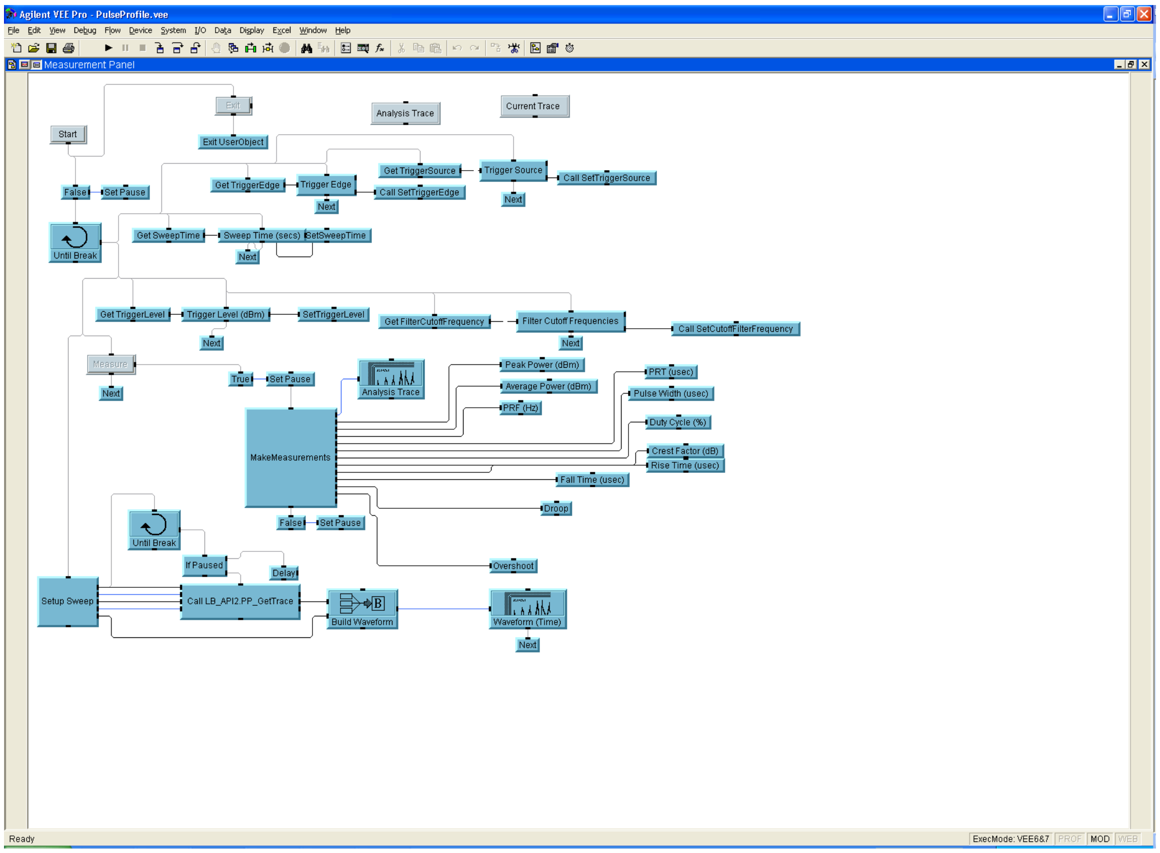


Figure 9: The detail view of the Measurement Panel demonstrates how measurements are made. Note that clicking on any control will start a new measurement loop execution.

Changing Settings

All of the settings are on the left side of the panel. Simply change the value or selection as desired and the power sensor will be updated immediately. Please note that some changes may have no effect – for example, changing the trigger level will have no effect if the trigger source is set to “Internal Auto”.

After each change, all of the sensor values are read (using the ReadSetup UserFunction) and the values on the front panel are updated.

Measuring Quantities

The most complex part of the Measurement Panel UserObject is the “Measure” execution. All of these calculations are instrument measurements are performed in the Measurements UserObject. The most important points concerning these measurements are:

1) The current trace is transferred to the analysis trace and all calculations are performed on the analysis trace.
2) Many of these calculations require two peaks. If there are less than two peaks, a warning is displayed and no measurements are made.
3) There are other warnings as well if the pulse size is too large or small relative to the sweep time.
4) All of these measurements are gated measurements. Gate 1 is always set prior to the measurements being made. There are extensive comments in the code concerning how the gate is set for each measurement.

Also, please note that the acquisition of new traces is paused while the calculations are made.

Using and Modifying the Code

There are approximately 250 functions available in the DLL. Each of these functions is documented in more detail in the LadyBug User's Manual. Most are extremely straightforward and easy to use.

After the DLL is imported, they can be found in the Program Explorer under Compiled Functions → LB_API2. They can also be found in the function and object browser under Compile Functions in the LB_API2 category.

Sensor Communication

Communication with LadyBug power sensors is done through the index, the serial number, or the address of the sensor. In this example code, all communication is done through the address. This is because the address covers more functionality than the others and because it makes the code more flexible. For example, the user cannot change the serial number of a sensor but they can change the address.

Adding to Existing VEE Files

The UserObjects distributed with this code can be added directly to any existing VEE code. Be sure to import the DLL in the software before calling any of the functions. With a minimal amount of experience, you would probably prefer writing your own UserObjects and UserFunctions directly with the function calls.