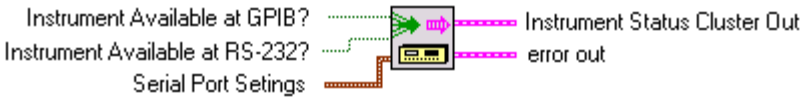
EPM2000 LabVIEW Building Applications Instructions

Introduction

The EPM2000 LabVIEW VI library is a collection of 57 configuration VIs that allow the construction of a wide variety of custom applications in a short time. The library contains VIs that help the EPM2000 function via remote—it does not contain plotting, charting, or file routines which are currently available in LabVIEW. Regularly check the Molectron website (*http://www.molectron.com/*) for updates to this and other software.

The beginning of any application is to compose the Status Cluster, which defines how the computer communicates with the EPM2000. The VI EPM2000 Compose Status Cluster.vi does exactly that. In the case of GPIB, set the Boolean value at the *Instrument Available at GPIB?* Terminal to *true*—the VI will automatically search the GPIB bus for the instrument. In the case of RS-232, the Boolean value at the *Instrument Available at RS-232?* Terminal must be set to *true* and the *Serial Port Settings* cluster needs to be properly set up.



EPM2000 Compose Status Cluster.vi

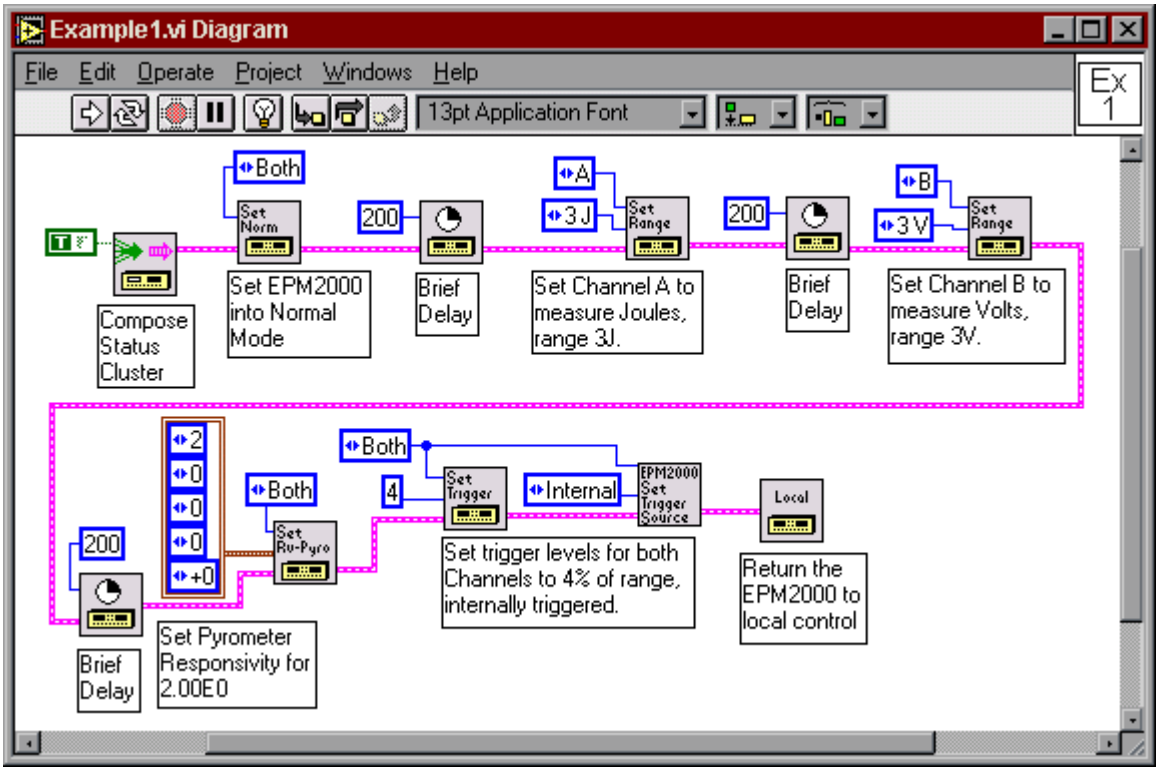
Once a Status Cluster is composed, VIs from the library can be strung in sequence to form any application. Some are standalone—they require no arguments. Others (such as *EPM2000 Set Units.vi*) require arguments. The easiest way to supply most of these arguments is to use the LabVIEW wiring tool to form a constant at the terminal. In most cases, the constant will supply all possible values for the terminal. Any VI may be more thoroughly examined by double-clicking it.

Sometimes, because of how the EPM2000 processes commands, a delay might be required. In such a case, use the *EPM2000 Command Delay.vi*, which preserves the data dependency of the Status Cluster.

The *EPM2000 Memory Store.vi* and the *EPM2000 Memory Recall.vi* are useful when frequently working with different settings in the same program. It is possible to program up to four different instrument states and save them to respective memory locations in the EPM2000. Recalling a stored state from EPM2000 memory is much faster than reprogramming the same variables via remote.

Below are examples showing some common implementations.

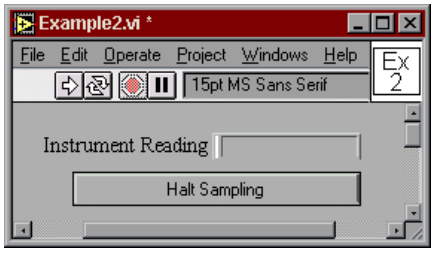
Example 1

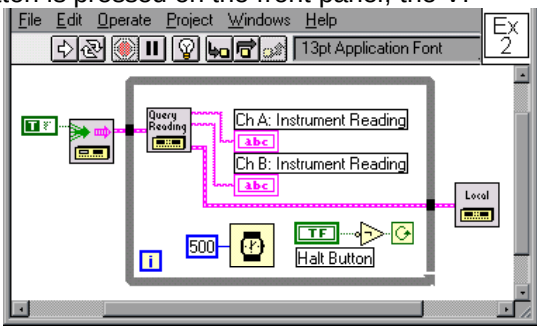


This is a diagram of a VI that locates the EPM2000, sets both channels to *Normal* mode, sets *Channel A* to measure Joules in the 3 Joule range, sets *Channel B* to measure volts in the 3 volt range, sets the pyrometer responsivity for both channels to 2.000 (*Channel B*, measuring voltage, doesn't care about its Rv setting for now), and sets both triggers for an internal level of 4% of full range. **Note**: Delays had to be added to assure that all commands are properly processed by the EPM2000. This VI is saved as *Example1.vi*.

Example 2

The VI in this example reads the front panel of the EPM2000 twice every second and reports the value to the String Indicator. It uses *EPM2000 Query Reading.vi*, which is useful for quick front-panel readings. The output string can be parsed by a custom routine and the data used to fill an array, plot to a chart, or trigger any other sort of function. When the **Halt Sampling** button is pressed on the front panel, the VI stops sampling and returns the EPM2000 to local control. It is useful to always return the VI to local control at the end of an application. During GPIB communications, the EPM2000 front panel is disabled and the only way to recover (outside of programmatic control) is to switch the instrument on and off. This, in turn, deletes buffer contents and may also disturb instrument settings. This VI is saved as *Example2.vi*.

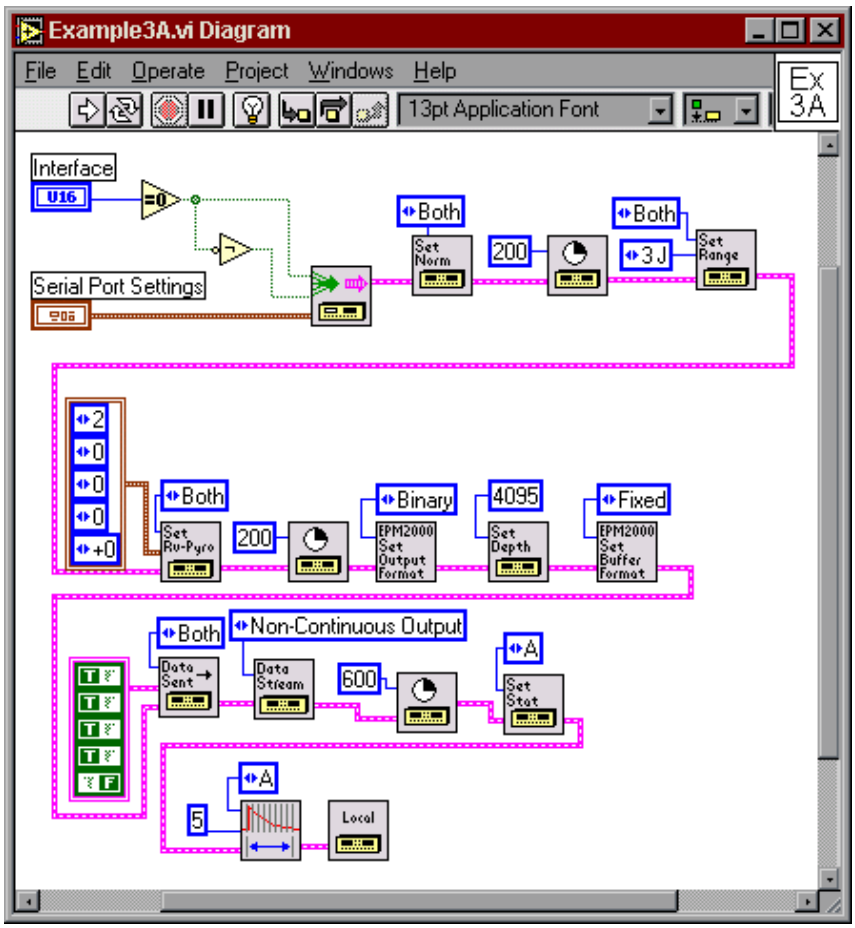




Example 3

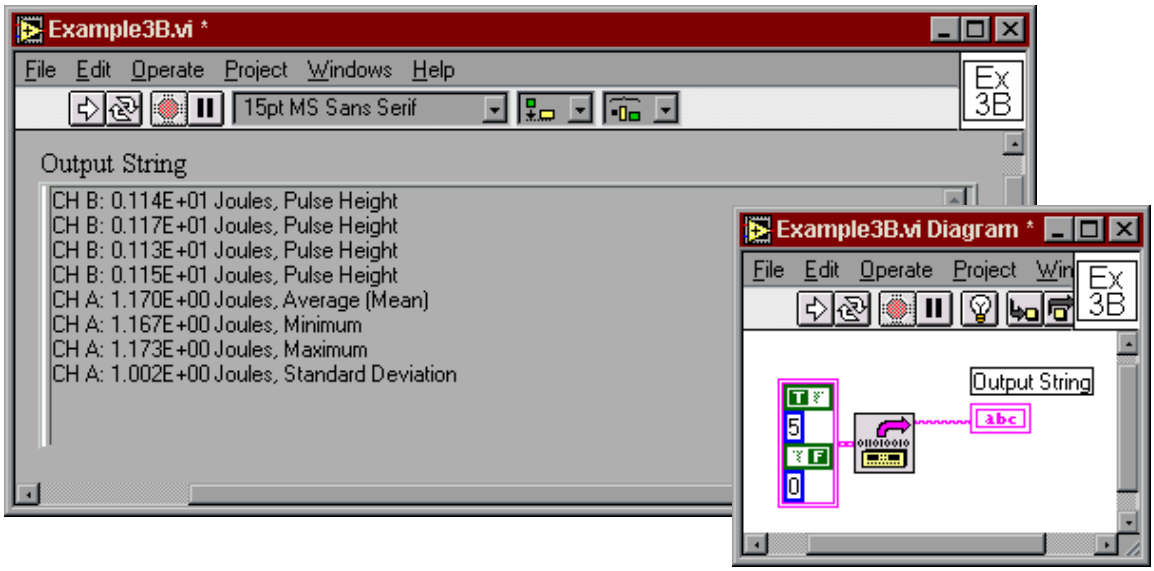
Using the *EPM2000 Query Reading.vi* to read data from the EPM2000 is adequate for quick measurements. Often, however, a series of measurements—perhaps an entire statistical series (*minimum*, *maximum*, *standard deviation*, and *mean of a series of pulses*)—must be taken. The most appropriate VI to use when faced with this situation is *EPM2000 Read Buffer.vi*.

In this example, the EPM2000 must be prepared to transmit the desired data in binary form from its buffer. *EPM2000 Read Buffer.vi* only interprets binary data. The following VI (saved as *Example3A.vi*) performs this task. Once preparations are complete, the EPM2000 is ready to receive five pulses. After the five pulses have been injected, executing *Example3B.vi* dumps the EPM2000 buffer and interprets the data.



Note: This example uses pulses of approximately 2.4 volts (approximately 1.17 Joules). The output should appear similar to the image of the front panel in *Example 3B*, although the actual data may be different. Activating scrollbars in the output window make it is possible to see much more data than what currently appears in the viewing area.

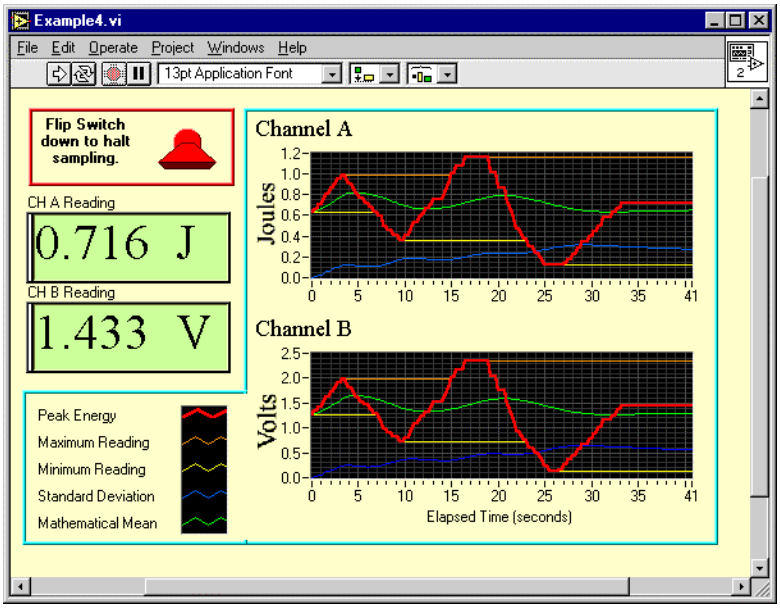
Note: The values of the Instrument Cluster that appear in the *Example 3B* diagram are hard-coded. Ordinarily, the Instrument Cluster data already exists in the program and is accessible. In this example, however, the Instrument Cluster is composed after the EPM2000 buffer is dumped. Re-establishing the EPM2000 address while the buffer contains data may result in loss of data due to the GPIB search algorithm.



The *EPM2000 Read Buffer.vi* can be used to repeatedly dump and translate the buffer.

Example 4

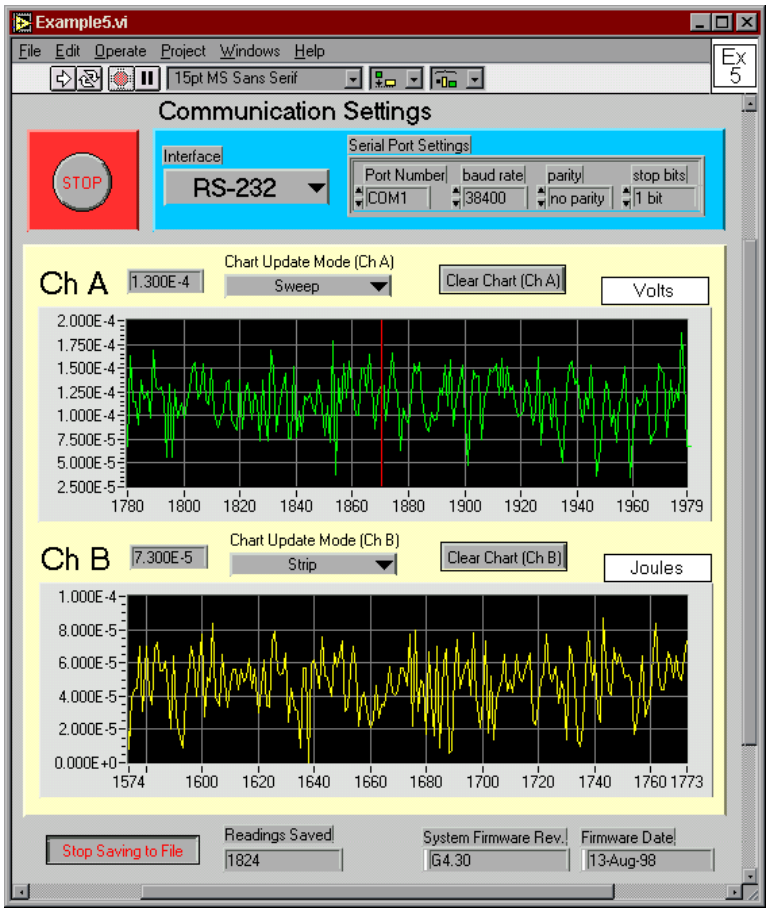
Example 4 demonstrates how to collect data from the EPM2000 at a regular rate (which is 4 Hz in this example). It also shows how to average and determine Standard Deviations, and display the collected data on a real-time display.



Example 4: Front Panel

Example 5

Example 5 shows how to collect data from the EPM2000 at a pulse-to-pulse rate. It also shows how to save the data to a file for later analysis, and display the collected data on a real-time display.



Example 5: Front Panel

List of Included VIs

EPM2000 Command Delay.vi

Waits a specified number of milliseconds before continuing, forcing a hold on the Instrument Status line in the process.

EPM2000 Compose Status Cluster.vi

Most of the VIs in this library utilize a cluster named *Instrument Status*, which contains communication details about the EPM2000. *EPM2000 Compose Status Cluster* allows a user to set those communication values from the beginning.

EPM2000 Current State.vi

Queries the EPM2000 and reports back the current state of several instrument options. The data are clustered out and can be pulled by name.

EPM2000 Get String.vi

Retrieves an ASCII string from the EPM2000, using the communications protocols derived from the Status Cluster. Most VIs communicating with the EPM2000 will use this VI.

EPM2000 Identification.vi

Returns *Instrument Name*, *Firmware Revision*, and *Firmware Date*.

EPM2000 Memory Recall.vi

Determines a memory location internal to the EPM2000 (*1*, *2*, *3*, or *4*). This memory location contains settings for the EPM2000 *Trigger Source*, *Trigger Level*, *Trigger Holdoff*, *Average Batch Size*, *Range*, *Pyroelectric Rv*, *Units*, *Display Speedup*, *Line Frequency*, *Statistics Batch Size*, and *Statistics Restart Mode*. This VI reassigns those remembered settings.

EPM2000 Memory Store.vi

Determines a memory location internal to the EPM2000 (*1*,*2*, *3*, or *4*). This memory location will then contain the EPM2000 *Trigger Source*, *Trigger Level*, *Trigger Holdoff*, *Average Batch Size*, *Range*, *Pyroelectric Rv*, *Units*, *Display Speedup*, *Line Frequency*, *Statistics Batch Size*, and *Statistics Restart Mode*. Use *EPM2000 Memory Recall.vi* to retrieve these settings.

EPM2000 Query Average Batch Size.vi

Determines the *size* (2–99) of the average batch.

EPM2000 Query Buffer Format.vi

Determines if the EPM2000 output buffer is set to *circular* or *fixed*.

EPM2000 Query Communications Parameters.vi

Determines the values of the Communications Parameters: *baud rate*, *parity* and *number of stop bits* for the RS-232 interface—as well as *GPIB address*, if the GPIB function exists.

EPM2000 Query Mode.vi

Determines which mode (*Statistics*, *Average*, *Normal*) is currently being used by the EPM2000.

EPM2000 Query Output Depth.vi

Determines the depth of the *Output Buffer* in bytes.

EPM2000 Query Output Format.vi

Determines whether the EPM2000 output will be of the *Binary*, *ASCII*, or *ASCII+* form.

EPM2000 Query Probe Type.vi

Determines which probe the EPM2000 expects to read: *Thermopile*, *Pyroelectric*, or *Silicon*.

EPM2000 Query Range.vi

Reads the EPM2000 range and measurement unit and also reads the maximum and minimum possible ranges given the current units.

EPM2000 Query Reading.vi

Reads the values and units displayed on the LCD, and returns a string output.

EPM2000 Query Responsivity.vi

Queries both the thermopile and pyroelectric responsivity (conversion factor). Responds into a cluster.

EPM2000 Query Statistics Batch Size.vi

Determines the size of the Statistics Batch for the specified channel.

EPM2000 Query Statistics Restart Mode.vi

Reads the Statistics Restart Mode: *Manual* or *Automatic*. If the EPM2000 is set to *Manual restart* mode, use *EPM2000 Statistics Start* to initiate a new batch.

EPM2000 Query Statistics Send Data.vi

Determines data that will be transmitted from the EPM2000 during live collection: *Average*, *Minimum Value*, *Maximum Value*, *Standard Deviation*, and/or *Header Data*.

EPM2000 Query System Parameters.vi

Determines the values of the System Parameters: *Audio Alarm*, *Backlight*, *Speedup Function*, and *Operating Line Frequency*.

EPM2000 Query Trigger Holdoff.vi

Determines the time (ms) the EPM2000 will wait after a valid trigger before looking for the next valid trigger.

EPM2000 Query Trigger Level.vi

Determines the level (% of full range) at which the specified EPM2000 channel triggers for a valid pulse.

EPM2000 Query Trigger Source.vi

Determines the source of the EPM2000 channel trigger: *Internal*, *External* (positive rising edge), *Slaved to the trigger bus*, or *External* (negative rising edge).

EPM2000 Query Units.vi

Determines which units the EPM2000 channels are measuring: *Joules*, *Volts*, *Watts*, or *frequency in Hertz*.

EPM2000 Read Buffer.vi

Retrieves and interprets the contents of the EPM2000 data buffer. **Note**: Buffer contents *must* be in binary form.

EPM2000 Reset Unit.vi

Sends a reset to the EPM2000, which releases the EPM2000 from GPIB control. **Note**: The next GPIB command reinstates control, and resets the thermopile responsivity.

EPM2000 Restore Factory Defaults.vi

Sets the EPM2000 to factory default values (as specified in the operator manual).

EPM2000 Return to Local Control.vi

Places the EPM2000 into local control.

EPM2000 Self Test.vi

Tests the EPM2000. Default output is *FALSE*, so Self Test returns *FALSE* if there is a problem—e.g., wrong GPIB address—communicating with the EPM2000.

EPM2000 Send String.vi

Sends an ASCII command to the EPM2000, using the communication protocols derived from the Status Cluster. Most VIs communicating with the EPM2000 use this VI.

EPM2000 Set Average Batch Size.vi

Allows the user to select the batch quantity for *Average* mode.

EPM2000 Set Average Mode.vi

Sets a specified EPM2000 channel to *Average* mode.

EPM2000 Set Buffer Format.vi

Sets the EPM2000 output buffer to *circular* or *fixed*.

EPM2000 Set Communication Parameters.vi

Sets the values of the Communications Parameters: *baud rate*, *parity* and *number of stop bits* for the RS-232 interface—as well as *GPIB address*, if the GPIB function exists. Parameters that have been remotely set do not take effect until the EPM2000 resets, so the final act of the VI is to reset the EPM2000. **Note**: If undesired, this final reset may be removed.

EPM2000 Set Datastream.vi

Sets the EPM2000 output to either *continuous* or *deactivate*, according to the input choice.

EPM2000 Set Mode.vi

Sets the specified EPM2000 channel to *Statistics*, *Batch*, or *Normal* mode.

EPM2000 Set Normal Mode.vi

Sets a specified EPM2000 channel to *Normal* mode.

EPM2000 Set Output Depth.vi

Sets the number of bytes the Output Buffer will hold. Once the Output Buffer is full, new data either replaces old data or is discarded. **Note**: Use *EPM2000 Query Buffer Format.vi* to determine data disposition.

EPM2000 Set Output Dump.vi

Sets the EPM2000 output to *non-continuous* and instructs the EPM2000 to immediately dump its buffer. **Note**: This VI can be modified to *not* change the EPM2000 data streaming. However, if the modified VI is executed on an EPM2000 with output set to *continuous*, no valid result will occur.

EPM2000 Set Output Format.vi

Sets the EPM2000 output mode to *Binary*, *ASCII*, or *ASCII+*. The manual contains more detailed information about the nature of the output data in those modes.

EPM2000 Set Pyro Responsivity.vi

Sets the pyroelectric responsivity (conversion factor). See the VI front panel for details regarding the upper and lower limits of the Rv. It may be convenient to copy the controls from the same front panel for other applications.

EPM2000 Set Range.vi

Sets the possible *Joule*, *Volt*, or *Watt* ranges, and the appropriate units, for the EPM2000. Example: Switching from 10 V range to 3 J range first sets the unit to *Joules* and then selects the range—if the Responsivity permits such a range. **Note**: Watts requires that a PowerMax or equivalent probe be attached to the EPM2000.

EPM2000 Set Statistics Batch Size.vi

Allows the user to select the batch quantity for *Statistics* mode.

EPM2000 Set Statistics Mode.vi

Sets a specified EPM2000 channel to *Statistics* mode.

EPM2000 Set Statistics Restart Mode.vi

Sets *Statistics* batch to either restart automatically, or hold the current batch until restarted manually.

EPM2000 Set Statistics Send Data.vi

Instructs the EPM2000 as to which data to transmit during live collection: *Average*, *Minimum Value*, *Maximum Value*, *Standard Deviation*, and/or *Header Data*.

EPM2000 Set System Parameters.vi

Sets the values of the System parameters: *Audio Alarm*, *Backlight*, *Speedup Function*, and *Operating Line Frequency*.

EPM2000 Set Thermo Responsivity.vi

Sets the thermopile responsivity (conversion factor). See the VI front panel for details regarding the upper and lower limits of the Rv. It may be convenient to copy the controls from the same front panel for other applications.

EPM2000 Set Trigger Configuration.vi

Sets three major trigger variables for the specified channel of the EPM2000.

EPM2000 Set Trigger Holdoff.vi

Sets the amount of time (ms) the EPM2000 waits after a valid trigger before looking for the next valid trigger.

EPM2000 Set Trigger Level.vi

Sets the level (% of full range) at which the EPM2000 triggers for a valid pulse.

EPM2000 Set Trigger Source.vi

Sets the EPM2000 to look for one of the following triggers: *Internal* (use *EPM2000 Set Trigger Level.vi* to determine the appropriate level), a *positive edge* at the External Trigger jack, a *negative edge* at the External Trigger jack, or *Slaved to the Trigger Bus*. The EPM2000 does not permit both channels to be set to slave via remote from the Trigger Bus.

EPM2000 Set Units.vi

Sets the EPM2000 channel to read *Volts*, *Joules*, *Watts*, or *frequency in Hertz*.

EPM2000 Set Zero.vi

Directs the EPM2000 to use the current thermopile input as the zero offset. This VI has no effect while taking samples from a pyroelectric probe.

EPM2000 Statistics Start.vi

Sends a **restart for statistics** command to the EPM2000 and resets the count if the EPM2000 hasn't finished counting. This VI also places the EPM2000 in *Statistics* mode, if it isn't already in that mode.

EPM2000 vi Tree.vi

Diagrams all the VIs in the EPM2000 library and groups them according to their function.

Search GPIB.vi

Searches the GPIB bus using ***IDN** to query every address—searching for the specified string value in the response. Any instrument or device on the GPIB bus which responds to the ***IDN?** command can use this VI.

Serial Setup.VI

Initializes the serial port and then uses the ***IDN?** command to determine if an EPM2000 is attached.