RTP Series

API Programming Reference

© 2019 WTG

5/13/2019

Boonton

This page i s intentionally left blank.

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

© 2019 WTG

Modules

1 Modules

Power Sensor Library 1.1.0

Modules

Here is a list of all modules:

Generated by doxygen 1.8.15

1.1 Scpi Functions

Power Sensor Library 1.1.0

Functions

Scpi Functions

© 2019 WTG

Functions

EXPORT int	**PwrSnsr_SendSCPICommand** (**SessionID** Vi, const char *Command)
	Send a SCPI command to the instrument. More...
EXPORT int	**PwrSnsr_ReadSCPI** (**SessionID** Vi, int ValueBufferSize, long *ValueActualSize, char Value[], int Timeout)
	Read a SCPI string response from the instrument. More...
EXPORT int	**PwrSnsr_SendSCPIToNamedParser** (**SessionID** Vi, const char *name, const char *Command)
	Send a SCPI command to the instrument using a named SCPI parser. More...
EXPORT int	**PwrSnsr_ReadSCPIFromNamedParser** (**SessionID** Vi, const char *name, int ValueBufferSize, long *ValueActualSize, char Value[], int Timeout)
	Read a SCPI string response from the instrument. More...
EXPORT int	**PwrSnsr_SendSCPIBytes** (**SessionID** Vi, int CommandBufferSize, char Command[])
	Send a SCPI command as a byte array. More...
EXPORT int	**PwrSnsr_ReadSCPIBytes** (**SessionID** Vi, int ValueBufferSize, char Value[], long *ValueActualSize, int Timeout)
	Read a SCPI byte array response from the instrument. More...

Detailed Description

© 2019 WTG

SCPI command functions

Function Documentation

◆ PwrSnsr_ReadSCPI()

EXPORT int PwrSnsr_ReadSCPI (

SessionID	Vi,
int	ValueBufferSize,
long *	ValueActualSize,
char	Value[],
int	Timeout

)

Read a SCPI string response from the instrument.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ValueBufferSize	Number of elements in Value.
ValueActualSize	Number of elements actually written to Value.
Value	The string returned from the instrument SCPI interface.
Timeout	Time out in milliseconds for the read operation. Use -1 for infinite and -2 to use the existing time out value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadSCPIBytes()

EXPORT int PwrSnsr_ReadSCPIBytes (

SessionID	Vi,
int	ValueBufferSize,
char	Value[],
long *	ValueActualSize,
int	Timeout

© 2019 WTG

)

Read a SCPI byte array response from the instrument.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ValueBufferSize	Number of elements in Value.
Value	The byte array returned from the instrument SCPI interface.
ValueActualSize	
Timeout	Time out in milliseconds for the read operation. Use -1 for infinite and -2 to use the existing time out value. Time out in milliseconds for the read operation. Use -1 for infinite and -2 to use the existing time out value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadSCPIFromNamedParser()

EXPORT int PwrSnsr_ReadSCPIFromNamedParser (

SessionID	Vi,
const char *	name,
int	ValueBufferSize,
long *	ValueActualSize,
char	Value[],
int	Timeout

)

Read a SCPI string response from the instrument.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
name	Name of the parser. If parser doesn't exist, returns PWR_SNSR_ERROR_NULL_POINTER. PwrSnsr_SendSCPIToNamedParser can be used to create a named parser.
ValueBufferSize	Number of elements in Value.

© 2019 WTG

ValueActualSize	Number of elements actually written to Value.
Value	The string returned from the instrument SCPI interface.
Timeout	Time out in milliseconds for the read operation. Use -1 for infinite and -2 to use the existing time out value.

Returns

Success (0) or error code.

◆PwrSnsr_SendSCPIBytes()

EXPORT int PwrSnsr_SendSCPIBytes (**SessionID** Vi, int CommandBufferSize, char Command[])

Send a SCPI command as a byte array.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
CommandBufferSize	Number of elements in Command.
Command	Command to send.

Returns

Success (0) or error code.

◆PwrSnsr_SendSCPICommand()

EXPORT int PwrSnsr_SendSCPICommand (**SessionID** Vi, const char * Command)

Send a SCPI command to the instrument.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Command

Returns

Success (0) or error code.

◆ PwrSnsr_SendSCPIToNamedParser()

EXPORT int PwrSnsr_SendSCPIToNamedParser (**SessionID** Vi, const char * name, const char * Command)

Send a SCPI command to the instrument using a named SCPI parser.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
name	Name of the parser. Creates a new parser if the name is not already used.
Command	

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.2 Session Management

Power Sensor Library 1.1.0

Functions

Session Management

Functions

EXPORT int	**PwrSnsr_FindResources** (const char *Delimiter, int ValBufferSize, char Val[])

© 2019 WTG

	Returns a delimited string of available resources. These strings can be used in the initialize function to open a session to an instrument. More...
EXPORT int	**PwrSnsr_GetMinimumSupportedFirmware** (int *Version)
	Gets the minimum supported firmware as an integer. Format is YYYYMMDD. More...
EXPORT int	**PwrSnsr_SetTimeOut** (**SessionID** Vi, long Milliseconds)
	Sets the time out in milliseconds for I/O. More...
EXPORT int	**PwrSnsr_GetTimeOut** (**SessionID** Vi, long *Val)
	Returns the time out value for I/O in milliseconds. More...
EXPORT int	**PwrSnsr_init** (char *ResourceName, **SessionID** *Vi)
	Initialize a communication session with a supported USB power sensor. More...
EXPORT int	**PwrSnsr_close** (**SessionID** Vi)
	Closes the I/O session to the instrument. Driver methods and properties that access the instrument are not accessible after Close is called. More...
EXPORT int	**PwrSnsr_GetError** (**SessionID** Vi, int *ErrorCode, int ErrorDescriptionBufferSize, char ErrorDescription[])
	This function retrieves and then clears the error information for the session. Normally, the error information describes the first error that occurred since the user last called the Get Error or Clear Error function. More...
EXPORT int	**PwrSnsr_ClearError** (**SessionID** Vi)

© 2019 WTG

	This function clears the error code and error description for the given session. More...
EXPORT int	**PwrSnsr_reset** (**SessionID** Vi)
EXPORT int	**PwrSnsr_self_test** (**SessionID** Vi, int *TestResult)
	Performs an instrument self test, waits for the instrument to complete the test, and queries the instrument for the results. If the instrument passes the test, TestResult is 0. More...
EXPORT int	**PwrSnsr_SetFetchLatency** (**SessionID** Vi, int Latency)
	Set the period the library waits to update fetch measurements in ms. More...
EXPORT int	**PwrSnsr_GetFetchLatency** (**SessionID** Vi, int *Latency)
	Get the period the library waits to update fetch measurements in ms. More...

Detailed Description

Session management functions

Function Documentation

◆ PwrSnsr_ClearError()

EXPORT int PwrSnsr_ClearError (**SessionID** Vi)

This function clears the error code and error description for the given session.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_close()

EXPORT int PwrSnsr_close (**SessionID** Vi)

Closes the I/O session to the instrument. Driver methods and properties that access the instrument are not accessible after Close is called.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_FindResources()

EXPORT int PwrSnsr_FindResources (const char * Delimiter, int ValBufferSize, char Val[])

Returns a delimited string of available resources. These strings can be used in the initialize function to open a session to an instrument.

Parameters

Delimiter	The string used to delimit the list of resources ie. "|", " ", ";", etc.
ValBufferSize	Number of elements in Val.
Val	The return string.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetError()

EXPORT int PwrSnsr_GetError (**SessionID** Vi, int * ErrorCode, int ErrorDescriptionBufferSize, char ErrorDescription[])

This function retrieves and then clears the error information for the session. Normally, the error information describes the first error that occurred since the user last called the Get Error or Clear Error function.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ErrorCode	
ErrorDescriptionBufferSize	
ErrorDescription	

Returns

Success (0) or error code.

◆PwrSnsr_GetFetchLatency()

EXPORT int PwrSnsr_GetFetchLatency (**SessionID** Vi, int * Latency)

Get the period the library waits to update fetch measurements in ms.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Latency	Fetch latency in ms.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetMinimumSupportedFirmware()

EXPORT int PwrSnsr_GetMinimumSupportedFirmware (int * Version)

Gets the minimum supported firmware as an integer. Format is YYYYMMDD.

Returns

Success (0) or error code.

◆PwrSnsr_GetTimeOut()

EXPORT int PwrSnsr_GetTimeOut (**SessionID** Vi, long * Val)

Returns the time out value for I/O in milliseconds.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Val	Time out in milliseconds. -1 denote infinite time out.

Returns

Success (0) or error code.

◆PwrSnsr_init()

EXPORT int PwrSnsr_init (char * ResourceName, **SessionID** * Vi)

Initialize a communication session with a supported USB power sensor.

Parameters

ResourceName	Name of the resource. The resource descriptor is in the following format: USB::[VID]::[PID]::[Serial Number]::SNSR

© 2019 WTG

Where serial number is the USB power meter's serial number in decimal format, and the VID and PID are in hexidecimal format.
e.g. For serial number 1234, VID of 0x1bfe and PID of 0x5500:
USB::0x1BFE::0x5500::1234::SNSR
Multiple channel synthetic meters can be defined by combining more than one descriptor separated by a semicolon. Channel assignment is determined by the order in the list, in other words CH1 would be the first listed resource,
CH2 the second resource, etc.
e.g. Define a synthetic peak power meter using serial number 1234 for CH1 and serial number 4242 for CH2:
USB::0x1BFE::0x5500::1234::SNSR;USB::0x1BFE::0x5500::4242::SNSR

Vi The SessionID handle

Returns

Success (0) or error code.

◆PwrSnsr_reset()

EXPORT int PwrSnsr_reset (**SessionID** Vi)

Places the instrument in a known state.

Parameters

Vi The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_self_test()

EXPORT int PwrSnsr_self_test (**SessionID** Vi, int * TestResult)

© 2019 WTG

Performs an instrument self test, waits for the instrument to complete the test, and queries the instrument for the results. If the instrument passes the test, TestResult is 0.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
TestResult	Error or success code.

Returns

Success (0) or error code.

◆PwrSnsr_SetFetchLatency()

EXPORT int PwrSnsr_SetFetchLatency (**SessionID** Vi, int Latency)

Set the period the library waits to update fetch measurements in ms.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Latency	Fetch latency in ms.

Returns

Success (0) or error code.

◆PwrSnsr_SetTimeOut()

EXPORT int PwrSnsr_SetTimeOut (**SessionID** Vi, long Milliseconds)

Sets the time out in milliseconds for I/O.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Milliseconds	Time out in milliseconds. Use -1 for infinite time out.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.3 Measurements

Power Sensor Library 1.1.0

Functions

Measurements

Functions	
EXPORT int	**PwrSnsr_MeasurePower** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return average power using a default instrument configuration in Modulated Mode and dBm units. Instrument remains stopped in Modulated Mode after a measurement. More...
EXPORT int	**PwrSnsr_FetchCWPower** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the most recently acquired CW power. More...
EXPORT int	**PwrSnsr_MeasureVoltage** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return average voltage using a default instrument configuration in Modulated Mode

© 2019 WTG

	and volts units. Instrument remains stopped in Modulated Mode after a measurement. More...
EXPORT int	**PwrSnsr_ReadWaveformMinMax** (**SessionID** Vi, const char *Channel, int MinWaveformBufferSize, float MinWaveform[], int *MinWaveformActualSize, int MaxWaveformBufferSize, float MaxWaveform[], int *MaxWaveformActualSize, int WaveformArrayBufferSize, float WaveformArray[], int *WaveformArrayActualSize)
	Initiates an acquisition on all enabled channels, waits (up to MaxTime) for the acquisition to complete, and returns the min/max waveforms for this channel. Call FetchMinMaxWaveform to obtain the min/max waveforms for other channels. More...
EXPORT int	**PwrSnsr_ReadWaveform** (**SessionID** Vi, const char *Channel, int WaveformArrayBufferSize, float WaveformArray[], int *WaveformArrayActualSize)
	Initiates an acquisition on all enabled channels, waits (up to MaxTime) for the acquisition to complete, and returns the waveform for this channel. Call FetchWaveform to obtain the waveforms for other channels. More...
EXPORT int	**PwrSnsr_FetchWaveformMinMax** (**SessionID** Vi, const char *Channel, int MinWaveformBufferSize, float MinWaveform[], int *MinWaveformActualSize, int MaxWaveformBufferSize, float MaxWaveform[], int *MaxWaveformActualSize, int WaveformArrayBufferSize, float WaveformArray[], int *WaveformArrayActualSize)
	Returns the previously acquired minimum and maximum waveforms for this specified channel. The acquisition must be made prior

© 2019 WTG

	to calling this method. Call this method separately for each channel. More...
EXPORT int	**PwrSnsr_FetchWaveform** (**SessionID** Vi, const char *Channel, int WaveformArrayBufferSize, float WaveformArray[], int *WaveformArrayActualSize)
	Returns a previously acquired waveform for this channel. The acquisition must be made prior to calling this method. Call this method separately for each channel. More...
EXPORT int	**PwrSnsr_FetchPowerArray** (**SessionID** Vi, const char *Channel, float *PulsePeak, **PwrSnsrCondCodeEnum** *PulsePeakValid, float *PulseCycleAvg, **PwrSnsrCondCodeEnum** *PulseCycleAvgValid, float *PulseOnAvg, **PwrSnsrCondCodeEnum** *PulseOnValid, float *IEEETop, **PwrSnsrCondCodeEnum** *IEEETopValid, float *IEEEBottom, **PwrSnsrCondCodeEnum** *IEEEBottomValid, float *Overshoot, **PwrSnsrCondCodeEnum** *OvershootValid, float *Droop, **PwrSnsrCondCodeEnum** *DroopValid)
	Returns an array of the current automatic amplitude measurements performed on a periodic pulse waveform. More...
EXPORT int	**PwrSnsr_FetchTimeArray** (**SessionID** Vi, const char *Channel, float *Frequency, **PwrSnsrCondCodeEnum** *FrequencyValid, float *Period, **PwrSnsrCondCodeEnum** *PeriodValid, float *Width, **PwrSnsrCondCodeEnum** *WidthValid, float *Offtime, **PwrSnsrCondCodeEnum** *OfftimeValid, float *DutyCycle, **PwrSnsrCondCodeEnum** *DutyCycleValid, float *Risetime, **PwrSnsrCondCodeEnum** *RisetimeValid, float *Falltime, **PwrSnsrCondCodeEnum** *FalltimeValid, float *EdgeDelay, **PwrSnsrCondCodeEnum** *EdgeDelayValid, float *Skew, **PwrSnsrCondCodeEnum** *SkewValid)

© 2019 WTG

	Returns an array of the current automatic timing measurements performed on a periodic pulse waveform. More...
EXPORT int	**PwrSnsr_FetchCWArray** (**SessionID** Vi, const char *Channel, float *PeakAverage, **PwrSnsrCondCodeEnum** *PeakAverageValid, float *PeakMax, **PwrSnsrCondCodeEnum** *PeakMaxValid, float *PeakMin, **PwrSnsrCondCodeEnum** *PeakMinValid, float *PeakToAvgRatio, **PwrSnsrCondCodeEnum** *PeakToAvgRatioValid)
	Returns the current average, maximum, minimum powers or voltages and the peak-to-average ratio of the specified channel. Units are the same as the channel units. Note the peak-to-average ratio and marker ratio are returned in dB for logarithmic channel units, and percent for all other channel units. More...
EXPORT int	**PwrSnsr_FetchRiseTime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the interval between the first signal crossing of the proximal line to the first signal crossing of the distal line. More...
EXPORT int	**PwrSnsr_FetchWidth** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the pulse width, i.e. the interval between the first and second signal crossings of the mesial line. More...
EXPORT int	**PwrSnsr_FetchPulsePeak** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the peak amplitude during the pulse. More...

© 2019 WTG

EXPORT int	**PwrSnsr_FetchPulseOnAverage** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Average power of the ON portion of the pulse. More...
EXPORT int	**PwrSnsr_FetchPulseCycleAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the average power of the entire pulse. More...
EXPORT int	**PwrSnsr_FetchPRF** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency). More...
EXPORT int	**PwrSnsr_FetchPeriod** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the interval between two successive pulses. (Reciprocal of the Pulse RepetitionFrequency) More...
EXPORT int	**PwrSnsr_FetchOvershoot** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the difference between the distortion following a major transition and the IEEE top line in dB or percent, depending on the channel units. More...
EXPORT int	**PwrSnsr_FetchOfftime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the time a repetitive pulse is off. (Equal to the pulse period minus the

© 2019 WTG

	pulsewidth). More...
EXPORT int	**PwrSnsr_FetchIEEETop** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the IEEE-defined top line, i.e. the portion of a pulse waveform which represents the second nominal state of a pulse. More...
EXPORT int	**PwrSnsr_FetchIEEEBottom** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a pulse departs and to which it ultimately returns. More...
EXPORT int	**PwrSnsr_FetchFallTime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the interval between the last signal crossing of the distal line to the last signalcrossing of the proximal line. More...
EXPORT int	**PwrSnsr_FetchEdgeDelay** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns time offset from the trigger reference to the first mesial transition level of either slope on the waveform. More...
EXPORT int	**PwrSnsr_FetchDutyCycle** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Returns the ratio of the pulse on-time to off-time. More...

© 2019 WTG

EXPORT int	**PwrSnsr_ReadCWPower** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
EXPORT int	**PwrSnsr_ReadCWArray** (**SessionID** Vi, const char *Channel, float *PeakAverage, **PwrSnsrCondCodeEnum** *PeakAverageValid, float *PeakMax, **PwrSnsrCondCodeEnum** *PeakMaxValid, float *PeakMin, **PwrSnsrCondCodeEnum** *PeakMinValid, float *PeakToAvgRatio, **PwrSnsrCondCodeEnum** *PeakToAvgRatioValid)
	Returns the current average, maximum, minimum powers or voltages and the peak-to-average ratio of the specified channel. Units are the same as the channel's units. Note the peak-to-average ratio and marker ratio are returned in dB for logarithmic channel units, and percent for all other channel units. More...
EXPORT int	**PwrSnsr_ReadPowerArray** (**SessionID** Vi, const char *Channel, float *PulsePeak, **PwrSnsrCondCodeEnum** *PulsePeakValid, float *PulseCycleAvg, **PwrSnsrCondCodeEnum** *PulseCycleAvgValid, float *PulseOnAvg, **PwrSnsrCondCodeEnum** *PulseOnValid, float *IEEETop, **PwrSnsrCondCodeEnum** *IEEETopValid, float *IEEEBottom, **PwrSnsrCondCodeEnum** *IEEEBottomValid, float *Overshoot, **PwrSnsrCondCodeEnum** *OvershootValid, float *Droop, **PwrSnsrCondCodeEnum** *DroopValid)
	Returns an array of the current automatic amplitude measurements performed on a periodic pulse waveform. More...
EXPORT int	**PwrSnsr_ReadTimeArray** (**SessionID** Vi, const char *Channel, float *Frequency, **PwrSnsrCondCodeEnum** *FrequencyValid, float *Period, **PwrSnsrCondCodeEnum**

© 2019 WTG

	*PeriodValid, float *Width, **PwrSnsrCondCodeEnum** *WidthValid, float *Offtime, **PwrSnsrCondCodeEnum** *OfftimeValid, float *DutyCycle, **PwrSnsrCondCodeEnum** *DutyCycleValid, float *Risetime, **PwrSnsrCondCodeEnum** *RisetimeValid, float *Falltime, **PwrSnsrCondCodeEnum** *FalltimeValid, float *EdgeDelay, **PwrSnsrCondCodeEnum** *EdgeDelayValid, float *Skew, **PwrSnsrCondCodeEnum** *SkewValid)
	Returns an array of the current automatic timing measurements performed on a periodic pulse waveform. More...
EXPORT int	**PwrSnsr_ReadDutyCycle** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the ratio of the pulse on-time to off-time. More...
EXPORT int	**PwrSnsr_ReadEdgeDelay** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns time offset from the trigger reference to the first mesial transition level of either slope on the waveform. More...
EXPORT int	**PwrSnsr_ReadFallTime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the interval between the last signal crossing of the distal line to the last signal crossing of the proximal line. More...
EXPORT int	**PwrSnsr_ReadIEEEBottom** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a

© 2019 WTG

	pulse departs and to which it ultimately returns. More...
EXPORT int	**PwrSnsr_ReadIEEETop** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the IEEE-defined top line, i.e. the portion of a pulse waveform which represents the second nominal state of a pulse. More...
EXPORT int	**PwrSnsr_ReadOfftime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the time a repetitive pulse is off. (Equal to the pulse period minus the pulse width). More...
EXPORT int	**PwrSnsr_ReadOvershoot** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the difference between the distortion following a major transition and the IEEE top line in dB or percent, depending on the channel units. More...
EXPORT int	**PwrSnsr_ReadPeriod** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the interval between two successive pulses. More...
EXPORT int	**PwrSnsr_ReadPRF** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency). More...
EXPORT int	**PwrSnsr_ReadPulseCycleAvg** (**SessionID** Vi, const char *Channel,

© 2019 WTG

	PwrSnsrCondCodeEnum *CondCode, float *Val)
	Returns the average power of the entire pulse. More...
EXPORT int	**PwrSnsr_ReadPulseOnAverage** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Average power of the ON portion of the pulse. More...
EXPORT int	**PwrSnsr_ReadPulsePeak** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the peak amplitude during the pulse. More...
EXPORT int	**PwrSnsr_ReadRiseTime** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the interval between the first signal crossing of the proximal line to the first signal crossing of the distal line. More...
EXPORT int	**PwrSnsr_ReadWidth** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the pulse width, i.e. the interval between the first and second signal crossings of the mesial line. More...

Detailed Description

General measurement functions

Function Documentation

© 2019 WTG

◆PwrSnsr_FetchCWArray()

EXPORT int PwrSnsr_FetchCWArray (

SessionID	Vi,
const char *	Channel,
float *	PeakAverage,
PwrSnsrCondCodeEnum *	PeakAverageValid,
float *	PeakMax,
PwrSnsrCondCodeEnum *	PeakMaxValid,
float *	PeakMin,
PwrSnsrCondCodeEnum *	PeakMinValid,
float *	PeakToAvgRatio,
PwrSnsrCondCodeEnum *	PeakToAvgRatioValid

)

Returns the current average, maximum, minimum powers or voltages and the peak-to-average ratio of the specified channel. Units are the same as the channel units. Note the peak-to-average ratio and marker ratio are returned in dB for logarithmic channel units, and percent for all other channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PeakAverage	Average power of the peak power envelope.
PeakAverageValid	Condition code.
PeakMax	maximum power of the peak power envelope.
PeakMaxValid	Condition code.
PeakMin	Minimum power of the peak power envelope.
PeakMinValid	Condition code.
PeakToAvgRatio	Peak to average ratio.
PeakToAvgRatioValid	Condition code.

Returns

© 2019 WTG

Success (0) or error code.

◆PwrSnsr_FetchCWPower()

EXPORT int PwrSnsr_FetchCWPower (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	CondCode,
float *	Val

)

Returns the most recently acquired CW power.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	CW power in channel units.

Returns

Success (0) or error code.

◆PwrSnsr_FetchDutyCycle()

EXPORT int PwrSnsr_FetchDutyCycle (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	IsValid,
float *	Val

)

Returns the ratio of the pulse on-time to off-time.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchEdgeDelay()

EXPORT int PwrSnsr_FetchEdgeDelay (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	IsValid,
float *	Val

)

Returns time offset from the trigger reference to the first mesial transition level of either slope on the waveform.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	
Val	

Returns

Success (0) or error code.

◆PwrSnsr_FetchFallTime()

EXPORT int PwrSnsr_FetchFallTime (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	IsValid,
float *	Val

© 2019 WTG

)

Returns the interval between the last signal crossing of the distal line to the last signalcrossing of the proximal line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchIEEEBottom()

EXPORT int PwrSnsr_FetchIEEEBottom (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	IsValid,
float *	Val

)

Returns the IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a pulse departs and to which it ultimately returns.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_FetchIEEETop()

EXPORT int PwrSnsr_FetchIEEETop (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * IsValid, float * Val)

Returns the IEEE-defined top line, i.e. the portion of a pulse waveform which represents the second nominal state of a pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchOfftime()

EXPORT int PwrSnsr_FetchOfftime (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * IsValid, float * Val)

Returns the time a repetitive pulse is off. (Equal to the pulse period minus the pulsewidth).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchOvershoot()

EXPORT int PwrSnsr_FetchOvershoot (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * IsValid, float * Val)

Returns the difference between the distortion following a major transition and the IEEE top line in dB or percent, depending on the channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchPeriod()

EXPORT int PwrSnsr_FetchPeriod (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * IsValid, float * Val)

© 2019 WTG

Returns the interval between two successive pulses. (Reciprocal of the Pulse RepetitionFrequency)

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchPowerArray()

```
EXPORT int PwrSnsr_FetchPowerArray (
    SessionID                   Vi,
    const char *                Channel,
    float *                     PulsePeak,
    PwrSnsrCondCodeEnum *       PulsePeakValid,
    float *                     PulseCycleAvg,
    PwrSnsrCondCodeEnum *       PulseCycleAvgValid,
    float *                     PulseOnAvg,
    PwrSnsrCondCodeEnum *       PulseOnValid,
    float *                     IEEETop,
    PwrSnsrCondCodeEnum *       IEEETopValid,
    float *                     IEEEBottom,
    PwrSnsrCondCodeEnum *       IEEEBottomValid,
    float *                     Overshoot,
    PwrSnsrCondCodeEnum *       OvershootValid,
    float *                     Droop,
    PwrSnsrCondCodeEnum *       DroopValid
```


© 2019 WTG

)

Returns an array of the current automatic amplitude measurements performed on a periodic pulse waveform.

Measurements performed are: peak amplitude during the pulse, average amplitude over a full cycle of the pulse waveform, average amplitude during the pulse, IEEE top amplitude, IEEE bottom amplitude, and overshoot. Units are the same as the channel's units.

Note the pulse overshoot is returned in dB for logarithmic channel units, and percent for all other units. Also, the pulse ?ON interval used for peak and average calculations is defined by the SENSe:PULSe:STARTGT and :ENDGT time gating settings.

A full pulse (rise and fall) must be visible on the display to make average and peak pulse power measurements, and a full cycle of the waveform must be visible to calculate average cycle amplitude.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PulsePeak	The peak amplitude during the pulse.
PulsePeakValid	Condition code.
PulseCycleAvg	Average cycle amplitude.
PulseCycleAvgValid	Condition code.
PulseOnAvg	Average power of the ON portion of the pulse.
PulseOnValid	Condition code.
IEEETop	The IEEE-defined top line, i.e. the portion of a pulse waveform, which represents the second nominal state of a pulse.
IEEETopValid	Condition code.
IEEEBottom	The IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a pulse departs and to which it ultimately returns.
IEEEBottomValid	Condition code.
Overshoot	The difference between the distortion following a major transition and the IEEE top

© 2019 WTG

	line in dB or percent,depending on the channel units.
OvershootValid	Condition code.
Droop	Pulse droop.
DroopValid	Condition code.

Returns

Success (0) or error code.

◆PwrSnsr_FetchPRF()

EXPORT int PwrSnsr_FetchPRF (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	IsValid,
float *	Val

)

Returns the number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchPulseCycleAvg()

EXPORT int PwrSnsr_FetchPulse CycleAvg (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	IsValid,
float *	Val

© 2019 WTG

)

Returns the average power of the entire pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchPulseOnAverage()

```
EXPORT int PwrSnsr_FetchPulseOnAverage (
    SessionID                Vi,
    const char *             Channel,
    PwrSnsrCondCodeEnum *    IsValid,
    float *                  Val
)
```

Average power of the ON portion of the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchPulsePeak()

```
EXPORT int PwrSnsr_FetchPulse (
    SessionID    Vi,
```


© 2019 WTG

Peak

	const char *	Channel,
	PwrSnsrCondCodeEnum *	IsValid,
	float *	Val

)

Returns the peak amplitude during the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchRiseTime()

EXPORT int PwrSnsr_FetchRiseTime (

SessionID	Vi,	
const char *	Channel,	
PwrSnsrCondCodeEnum *	IsValid,	
float *	Val	

)

Returns the interval between the first signal crossing of the proximal line to the first signal crossing of the distal line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_FetchTimeArray()

```
EXPORT int PwrSnsr_FetchTimeArray (
    SessionID Vi,
    const char * Channel,
    float * Frequency,
    PwrSnsrCondCodeEnum * FrequencyValid,
    float * Period,
    PwrSnsrCondCodeEnum * PeriodValid,
    float * Width,
    PwrSnsrCondCodeEnum * WidthValid,
    float * Offtime,
    PwrSnsrCondCodeEnum * OfftimeValid,
    float * DutyCycle,
    PwrSnsrCondCodeEnum * DutyCycleValid,
    float * Risetime,
    PwrSnsrCondCodeEnum * RisetimeValid,
    float * Falltime,
    PwrSnsrCondCodeEnum * FalltimeValid,
    float * EdgeDelay,
    PwrSnsrCondCodeEnum * EdgeDelayValid,
    float * Skew,
    PwrSnsrCondCodeEnum * SkewValid
)
```

Returns an array of the current automatic timing measurements performed on a periodic pulse waveform.

Measurements performed are: the frequency, period, width, offtime and duty cycle of the pulse waveform, and the risetime and falltime of the edge transitions. For each of the

© 2019 WTG

measurements to be performed, the appropriate items to be measured must within the trace window. Pulse frequency, period, offtime and duty cycle measurements require that an entire cycle of the pulse waveform (minimum of three edge transitions) be present. Pulse width measurement requires that at least one full pulse is visible, and is most accurate if the pulse width is at least 0.4 divisions. Risetime and falltime measurements require that the edge being measured is visible, and will be most accurate if the transition takes at least 0.1 divisions. It is always best to have the power meter set on the fastest timebase possible that meets the edge visibility restrictions. Set the trace averaging as high as practical to reduce fluctuations and noise in the pulse timing measurements. Note that the timing of the edge transitions is defined by the settings of the SENSe:PULSe:DISTal, :MESIal and :PROXimal settings; see the descriptions Forthose commands. Units are the same as the channel's units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Frequency	The number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency).
FrequencyValid	Condition code.
Period	The interval between two successive pulses.
PeriodValid	Condition code.
Width	The interval between the first and second signal crossings of the mesial line.
WidthValid	Condition code.
Offtime	The time a repetitive pulse is off. (Equal to the pulse period minus the pulse width).
OfftimeValid	Condition code.
DutyCycle	The ratio of the pulse on-time to period.
DutyCycleValid	Condition code.
Risetime	The interval between the first signal crossing of the proximal line to the first signal crossing of the distal line.
RisetimeValid	Condition code.
Falltime	The interval between the last signal crossing of the distal line to the last signal crossing of the proximal line.

© 2019 WTG

FalltimeValid	Condition code.
EdgeDelay	Time offset from the trigger reference to the first mesial transition level of either slope on the waveform.
EdgeDelayValid	Condition code.
Skew	The trigger offset between the assigned trigger channel and this channel.
SkewValid	Condition code.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchWaveform()

```
EXPORT int PwrSnsr_FetchWaveform (
    SessionID Vi,
    const char * Channel,
    int WaveformArrayBufferSize,
    float WaveformArray[],
    int * WaveformArrayActualSize
)
```

Returns a previously acquired waveform for this channel. The acquisition must be made prior to calling this method. Call this method separately for each channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
WaveformArrayBufferSize	Size in bytes of the Waveform buffer.
WaveformArray	The array contains the average waveform. Units for the individual array elements are in the channel units setting.
WaveformArrayActualSize	Size in bytes of the data written to WaveformArray.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_FetchWaveformMinMax()

```
EXPORT int PwrSnsr_FetchWaveformMinMax (
        SessionID                Vi,
        const char *             Channel,
        int                      MinWaveformBufferSize,
        float                    MinWaveform[],
        int *                    MinWaveformActualSize,
        int                      MaxWaveformBufferSize,
        float                    MaxWaveform[],
        int *                    MaxWaveformActualSize,
        int                      WaveformArrayBufferSize,
        float                    WaveformArray[],
        int *                    WaveformArrayActualSize
)
```

Returns the previously acquired minimum and maximum waveforms for this specified channel. The acquisition must be made prior to calling this method. Call this method separately for each channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MinWaveformBufferSize	Size in bytes of the MinWaveform buffer.
MinWaveform	This array contains the min waveform. Units for the individual array elements are in the channel units setting.
MinWaveformActualSize	Size in bytes of the data written to MinWaveform.
MaxWaveformBufferSize	Size in bytes of the MaxWaveform buffer.
MaxWaveform	This array contains the max waveform. Units for the individual array elements are in the channel units setting.

© 2019 WTG

MaxWaveformActualSize	Size in bytes of the data written to MaxWaveform.
WaveformArrayBufferSize	Size in bytes of the Waveform buffer.
WaveformArray	The array contains the average waveform. Units for the individual array elements are in the channel units setting.
WaveformArrayActualSize	Size in bytes of the data written to WaveformArray.

Returns

Success (0) or error code.

◆PwrSnsr_FetchWidth()

EXPORT int PwrSnsr_FetchWidth (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	IsValid,
float *	Val

)

Returns the pulse width, i.e. the interval between the first and second signal crossings of the mesial line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement return value.

Returns

Success (0) or error code.

◆PwrSnsr_MeasurePower()

EXPORT int PwrSnsr_MeasurePower (

SessionID	Vi,
const char *	Channel,

© 2019 WTG

		PwrSnsrCondCode Enum *	CondCode,
		float *	Val
)			

Return average power using a default instrument configuration in Modulated Mode and dBm units. Instrument remains stopped in Modulated Mode after a measurement.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	Average power in dBm

Returns

Success (0) or error code.

◆ PwrSnsr_MeasureVoltage()

EXPORT int PwrSnsr_MeasureVoltage	(	**SessionID**	Vi,
		const char *	Channel,
		PwrSnsrCondCode Enum *	CondCode,
		float *	Val
	)		

Return average voltage using a default instrument configuration in Modulated Mode and volts units. Instrument remains stopped in Modulated Mode after a measurement.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	Average voltage in linear volts.

Returns

Success (0) or error code.

© 2019 WTG

◆ PwrSnsr_ReadCWArray()

```
EXPORT int PwrSnsr_ReadCWArray (
        SessionID Vi,
        const char * Channel,
        float * PeakAverage,
        PwrSnsrCondCodeEnum * PeakAverageValid,
        float * PeakMax,
        PwrSnsrCondCodeEnum * PeakMaxValid,
        float * PeakMin,
        PwrSnsrCondCodeEnum * PeakMinValid,
        float * PeakToAvgRatio,
        PwrSnsrCondCodeEnum * PeakToAvgRatioValid
)
```

Returns the current average, maximum, minimum powers or voltages and the peak-to-average ratio of the specified channel. Units are the same as the channel's units. Note the peak-to-average ratio and marker ratio are returned in dB for logarithmic channel units, and percent for all other channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PeakAverage	Average power of the peak power envelope.
PeakAverageValid	Condition code.
PeakMax	Maximum power of the peak power envelope.
PeakMaxValid	Condition code.
PeakMin	Minimum power of the peak power envelope.
PeakMinValid	Condition code.
PeakToAvgRatio	Peak to average ratio.
PeakToAvgRatioValid	Condition code.

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_ReadCWPower()

```
EXPORT int PwrSnsr_ReadCWPower (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * IsValid,
    float * Val
)
```

Initiates a CW power acquisition and returns the result in channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsValid	Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadDutyCycle()

```
EXPORT int PwrSnsr_ReadDutyCycle (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Returns the ratio of the pulse on-time to off-time.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadEdgeDelay()

```
EXPORT int PwrSnsr_ReadEdgeDelay (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Returns time offset from the trigger reference to the first mesial transition level of either slope on the waveform.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadFallTime()

```
EXPORT int PwrSnsr_ReadFallTime (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
```


© 2019 WTG

float * Val

)

Returns the interval between the last signal crossing of the distal line to the last signal crossing of the proximal line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadIEEEBottom()

EXPORT int PwrSnsr_ReadIEEEBottom (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * CondCode, float * Val)

Returns the IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a pulse departs and to which it ultimately returns.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_ReadIEEETop()

```
EXPORT int PwrSnsr_ReadIEEETop (
                                SessionID Vi,
                                const char * Channel,
                                PwrSnsrCondCodeEnum * CondCode,
                                float * Val
                                )
```

Returns the IEEE-defined top line, i.e. the portion of a pulse waveform which represents the second nominal state of a pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadOfftime()

```
EXPORT int PwrSnsr_ReadOfftime (
                                SessionID Vi,
                                const char * Channel,
                                PwrSnsrCondCodeEnum * CondCode,
                                float * Val
                                )
```

Returns the time a repetitive pulse is off. (Equal to the pulse period minus the pulse width).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadOvershoot()

EXPORT int PwrSnsr_ReadOvershoot (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	CondCode,
float *	Val

)

Returns the difference between the distortion following a major transition and the IEEE top line in dB or percent, depending on the channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadPeriod()

EXPORT int PwrSnsr_ReadPeriod (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	CondCode,
float *	Val

© 2019 WTG

)

Returns the interval between two successive pulses.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadPowerArray()

EXPORT int PwrSnsr_ReadPowerArray (

SessionID	Vi,
const char *	Channel,
float *	PulsePeak,
PwrSnsrCondCodeEnum *	PulsePeakValid,
float *	PulseCycleAvg,
PwrSnsrCondCodeEnum *	PulseCycleAvgValid,
float *	PulseOnAvg,
PwrSnsrCondCodeEnum *	PulseOnValid,
float *	IEEETop,
PwrSnsrCondCodeEnum *	IEEETopValid,
float *	IEEEBottom,
PwrSnsrCondCodeEnum *	IEEEBottomValid,
float *	Overshoot,
PwrSnsrCondCodeEnum *	OvershootValid,
float *	Droop,
PwrSnsrCondCodeEnum *	DroopValid

© 2019 WTG

)

Returns an array of the current automatic amplitude measurements performed on a periodic pulse waveform.

Measurements performed are: peak amplitude during the pulse, average amplitude over a full cycle of the
pulse waveform, average amplitude during the pulse, IEEE top amplitude, IEEE bottom amplitude, and overshoot.
Units are the same as the channel's units. Note the pulse overshoot is returned in dB for logarithmic channel units,
and percent for all other units. Also, the pulse ON interval used for peak and average calculations is
defined by the SENSe:PULSe:STARTGT and :ENDGT time gating settings.
A full pulse (rise and fall) must be visible on the display to make average and peak pulse power measurements,
and a full cycle of the waveform must be visible to calculate average cycle amplitude.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PulsePeak	The peak amplitude during the pulse.
PulsePeakValid	Condition code.
PulseCycleAvg	Average cycle amplitude.
PulseCycleAvgValid	Condition code.
PulseOnAvg	Average power of the ON portion of the pulse.
PulseOnValid	Condition code.
IEEETop	The IEEE-defined top line, i.e. the portion of a pulse waveform, which represents the second nominal state of a pulse.
IEEETopValid	Condition code.
IEEEBottom	The IEEE-define base line, i.e. The two portions of a pulse waveform which represent the first nominal state from which a pulse departs and to which it ultimately returns.
IEEEBottomValid	Condition code.
Overshoot	The difference between the distortion following a major transition and the IEEE top

© 2019 WTG

	line in dB or percent,depending on the channel units.
OvershootValid	Condition code.
Droop	Pulse droop.
DroopValid	Condition code.

Returns

Success (0) or error code.

◆PwrSnsr_ReadPRF()

EXPORT int PwrSnsr_ReadPRF (**SessionID** Vi, const char * Channel, **PwrSnsrCondCode Enum** * CondCode, float * Val)

Returns the number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadPulseCycleAvg()

EXPORT int PwrSnsr_ReadPulseCycleAvg (**SessionID** Vi, const char * Channel, **PwrSnsrCondCode Enum** * CondCode,

© 2019 WTG

float * Val

)

Returns the average power of the entire pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadPulseOnAverage()

EXPORT int PwrSnsr_ReadPulseOnAverage (

SessionID Vi,
const char * Channel,
PwrSnsrCondCode Enum * CondCode,
float * Val

)

Average power of the ON portion of the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_ReadPulsePeak()

EXPORT int PwrSnsr_ReadPulsePeak (
SessionID Vi,
const char * Channel,
PwrSnsrCondCodeEnum * CondCode,
float * Val
)

Returns the peak amplitude during the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadRiseTime()

EXPORT int PwrSnsr_ReadRiseTime (
SessionID Vi,
const char * Channel,
PwrSnsrCondCodeEnum * CondCode,
float * Val
)

Returns the interval between the first signal crossing of the proximal line to the first signal crossing of the distal line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadTimeArray()

EXPORT int PwrSnsr_ReadTimeArray (

SessionID	Vi,
const char *	Channel,
float *	Frequency,
PwrSnsrCondCodeEnum *	FrequencyValid,
float *	Period,
PwrSnsrCondCodeEnum *	PeriodValid,
float *	Width,
PwrSnsrCondCodeEnum *	WidthValid,
float *	Offtime,
PwrSnsrCondCodeEnum *	OfftimeValid,
float *	DutyCycle,
PwrSnsrCondCodeEnum *	DutyCycleValid,
float *	Risetime,
PwrSnsrCondCodeEnum *	RisetimeValid,
float *	Falltime,
PwrSnsrCondCodeEnum *	FalltimeValid,
float *	EdgeDelay,
PwrSnsrCondCodeEnum *	EdgeDelayValid,
float *	Skew,
PwrSnsrCondCodeEnum *	SkewValid

© 2019 WTG

)

Returns an array of the current automatic timing measurements performed on a periodic pulse waveform.

Measurements performed are: the frequency, period, width, offtime and duty cycle of the pulse waveform, and the risetime and falltime of the edge transitions. For each of the measurements to be performed, the appropriate items to be measured must within the trace window. Pulse frequency, period, offtime and duty cycle measurements require that an entire cycle of the pulse waveform (minimum of three edge transitions) be present. Pulse width measurement requires that at least one full pulse is visible, and is most accurate if the pulse width is at least 0.4 divisions. Risetime and falltime measurements require that the edge being measured is visible, and will be most accurate if the transition takes at least 0.1 divisions. It is always best to have the power meter set on the fastest timebase possible that meets the edge visibility restrictions. Set the trace averaging as high as practical to reduce fluctuations and noise in the pulse timing measurements. Note that the timing of the edge transitions is defined by the settings of the SENSe:PULSe:DISTal, :MESIal and :PROXimal settings; see the descriptions For those commands. Units are the same as the channel's units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Frequency	The number of cycles of a repetitive signal that take place in one second (Pulse Repetition Frequency).
FrequencyValid	Condition code.
Period	The interval between two successive pulses.
PeriodValid	Condition code.
Width	The interval between the first and second signal crossings of the mesial line.
WidthValid	Condition code.
Offtime	The time a repetitive pulse is off. (Equal to the pulse period minus the pulse width).
OfftimeValid	Condition code.
DutyCycle	The ratio of the pulse on-time to period.
DutyCycleValid	Condition code.

© 2019 WTG

Risetime	The interval between the first signal crossing of the proximal line to the first signal crossing of the distal line.
RisetimeValid	Condition code.
Falltime	The interval between the last signal crossing of the distal line to the last signal crossing of the proximal line.
FalltimeValid	Condition code.
EdgeDelay	Time offset from the trigger reference to the first mesial transition level of either slope on the waveform.
EdgeDelayValid	Condition code.
Skew	The trigger offset between the assigned trigger channel and this channel.
SkewValid	Condition code.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadWaveform()

```
EXPORT int PwrSnsr_ReadWaveform (
                                  SessionID                 Vi,
                                  const char *              Channel,
                                  int                       WaveformArrayBufferSize,
                                  float                     WaveformArray[],
                                  int *                     WaveformArrayActualSize
                                 )
```

Initiates an acquisition on all enabled channels, waits (up to MaxTime) for the acquisition to complete, and returns the waveform for this channel. Call FetchWaveform to obtain the waveforms for other channels.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
WaveformArrayBufferSize	Size in bytes of the Waveform buffer.

© 2019 WTG

WaveformArray	The array contains the average waveform. Units for the individual array elements are in the channel units setting.
WaveformArrayActualSize	Size in bytes of the data written to WaveformArray.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadWaveformMinMax()

EXPORT int PwrSnsr_ReadWaveformMinMax (

SessionID	Vi,
const char *	Channel,
int	MinWaveformBufferSize,
float	MinWaveform[],
int *	MinWaveformActualSize,
int	MaxWaveformBufferSize,
float	MaxWaveform[],
int *	MaxWaveformActualSize,
int	WaveformArrayBufferSize,
float	WaveformArray[],
int *	WaveformArrayActualSize

)

Initiates an acquisition on all enabled channels, waits (up to MaxTime) for the acquisition to complete, and returns the min/max waveforms for this channel. Call FetchMinMaxWaveform to obtain the min/max waveforms for other channels.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MinWaveformBufferSize	Size in bytes of the MinWaveform buffer.

© 2019 WTG

MinWaveform	This array contains the min waveform. Units for the individual array elements are in the channel units setting.
MinWaveformActualSize	Size in bytes of the data written to MinWaveform.
MaxWaveformBufferSize	Size in bytes of the MaxWaveform buffer.
MaxWaveform	This array contains the max waveform. Units for the individual array elements are in the channel units setting.
MaxWaveformActualSize	Size in bytes of the data written to MaxWaveform.
WaveformArrayBufferSize	Size in bytes of the Waveform buffer.
WaveformArray	The array contains the average waveform. Units for the individual array elements are in the channel units setting.
WaveformArrayActualSize	Size in bytes of the data written to WaveformArray.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadWidth()

```
EXPORT int
PwrSnsr_ReadWidth (
                                SessionID       Vi,
                                const char *    Channel,
                                PwrSnsrCondCode
                                Enum *          CondCode,
                                float *         Val
                                )
```

Returns the pulse width, i.e. the interval between the first and second signal crossings of the mesial line.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

© 2019 WTG

Success (0) or error code.

Generated by doxygen 1.8.15

1.4 Trigger

Power Sensor Library 1.1.0

Functions

Trigger

Functions

EXPORT int	**PwrSnsr_GetTrigDelay** (**SessionID** Vi, float *Delay)
	Return the trigger delay time in seconds with respect to the trigger for the trigger display location in the LEFT position. More...
EXPORT int	**PwrSnsr_SetTrigDelay** (**SessionID** Vi, float Delay)
	Sets the trigger delay time in seconds with respect to the trigger for the trigger display location in the LEFT position. More...
EXPORT int	**PwrSnsr_GetTrigHoldoff** (**SessionID** Vi, float *Holdoff)
	Return the trigger holdoff time in seconds. More...
EXPORT int	**PwrSnsr_SetTrigHoldoff** (**SessionID** Vi, float Holdoff)
	Sets the trigger holdoff time in seconds. More...
EXPORT int	**PwrSnsr_GetTrigHoldoffMode** (**SessionID** Vi, **PwrSnsrHoldoffModeEnum** *HoldoffMode)

© 2019 WTG

	Returns the holdoff mode to normal or gap holdoff. More...
EXPORT int	**PwrSnsr_SetTrigHoldoffMode** (**SessionID** Vi, **PwrSnsrHoldoffModeEnum** HoldoffMode)
	Sets the holdoff mode to normal or gap holdoff. More...
EXPORT int	**PwrSnsr_GetTrigLevel** (**SessionID** Vi, float *Level)
	Return the trigger level for synchronizing data acquisition with a pulsed input signal. More...
EXPORT int	**PwrSnsr_SetTrigLevel** (**SessionID** Vi, float Level)
	Set the trigger level for synchronizing data acquisition with a pulsed input signal. More...
EXPORT int	**PwrSnsr_GetTrigMode** (**SessionID** Vi, **PwrSnsrTriggerModeEnum** *Mode)
	Return the trigger mode for synchronizing data acquisition with pulsed signals. More...
EXPORT int	**PwrSnsr_SetTrigMode** (**SessionID** Vi, **PwrSnsrTriggerModeEnum** Mode)
	Set the trigger mode for synchronizing data acquisition with pulsed signals. More...
EXPORT int	**PwrSnsr_GetTrigPosition** (**SessionID** Vi, **PwrSnsrTriggerPositionEnum** *Position)
	Return the position of the trigger event on displayed sweep. More...
EXPORT int	**PwrSnsr_SetTrigPosition** (**SessionID** Vi, **PwrSnsrTriggerPositionEnum** Position)
	Set the position of the trigger event on displayed sweep. More...
EXPORT int	**PwrSnsr_GetTrigSource** (**SessionID** Vi, **PwrSnsrTriggerSourceEnum** *Source)

© 2019 WTG

	Set the signal the power meter monitors for a trigger. It can be channel external input, or independent. More...
EXPORT int	**PwrSnsr_SetTrigSource** (**SessionID** Vi, **PwrSnsrTriggerSourceEnum** Source)
	Get the signal the power meter monitors for a trigger. It can be channel external input, or independent. More...
EXPORT int	**PwrSnsr_GetTrigStatus** (**SessionID** Vi, **PwrSnsrTriggerStatusEnum** *Status)
	The status of the triggering system. Update rate is controlled by FetchLatency setting. More...
EXPORT int	**PwrSnsr_GetTrigVernier** (**SessionID** Vi, float *Vernier)
	Return the fine position of the trigger event on the power sweep. More...
EXPORT int	**PwrSnsr_SetTrigVernier** (**SessionID** Vi, float Vernier)
	Set the fine position of the trigger event on the power sweep. More...
EXPORT int	**PwrSnsr_GetTrigSlope** (**SessionID** Vi, **PwrSnsrTriggerSlopeEnum** *Slope)
	Return the trigger slope or polarity. More...
EXPORT int	**PwrSnsr_SetTrigSlope** (**SessionID** Vi, **PwrSnsrTriggerSlopeEnum** Slope)
	Sets the trigger slope or polarity. More...
EXPORT int	**PwrSnsr_SetTrigOutMode** (**SessionID** Vi, const char *Channel, int Mode)
	Sets the trigger out/mult io mode. Setting trigger mode overrides this command. More...

© 2019 WTG

Detailed Description

Trigger related functions

Function Documentation

◆PwrSnsr_GetTrigDelay()

EXPORT int PwrSnsr_GetTrigDelay (**SessionID** Vi, float * Delay)

Return the trigger delay time in seconds with respect to the trigger for the trigger display location in the LEFT position.

Positive values cause the actual trigger to occur after the trigger condition is met. This places the trigger event to the left of the trigger point on the display, and is useful for viewing events during a pulse, some fixed delay time after the rising edge trigger. Negative trigger delay places the trigger event to the right of the trigger point on the display, and is useful for looking at events before the trigger edge.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Delay	

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigHoldoff()

EXPORT int PwrSnsr_GetTrigHoldoff (**SessionID** Vi, float * Holdoff

© 2019 WTG

)

Return the trigger holdoff time in seconds.

Trigger holdoff is used to disable the trigger for a specified amount of time after each trigger event. The holdoff time starts immediately after each valid trigger edge, and will not permit any new triggers until the time has expired. When the holdoff time is up, the trigger re-arms, and the next valid trigger event (edge) will cause a new sweep. This feature is used to help synchronize the power meter with burst waveforms such as a TDMA or GSM frame. The trigger holdoff resolution is 10 nanoseconds, and it should be set to a time that is just slightly shorter than the frame repetition interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Holdoff	

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigHoldoffMode()

EXPORT int PwrSnsr_GetTrigHoldoffMode (

SessionID	Vi,
PwrSnsrHoldoffModeEnum *	HoldoffMode

)

Returns the holdoff mode to normal or gap holdoff.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
HoldoffMode	holdoff mode.

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigLevel()

© 2019 WTG

EXPORT int PwrSnsr_GetTrigLevel (**SessionID** Vi, float * Level)

Return the trigger level for synchronizing data acquisition with a pulsed input signal.

The internal trigger level entered should include any global offset and will also be affected by the frequency cal factor. The available internal trigger level range is sensor dependent. The trigger level is set and returned in dBm. This setting is only valid for normal and auto trigger modes.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Level	Trigger level in dBm.

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigMode()

EXPORT int PwrSnsr_GetTrigMode (**SessionID** Vi, **PwrSnsrTriggerModeEnum** * Mode)

Return the trigger mode for synchronizing data acquisition with pulsed signals.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Mode	Trigger mode.

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigPosition()

© 2019 WTG

EXPORT int PwrSnsr_GetTrigPosition (**SessionID** Vi, **PwrSnsrTriggerPositionEnum** * Position)

Return the position of the trigger event on displayed sweep.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Position	Trigger position.

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigSlope()

EXPORT int PwrSnsr_GetTrigSlope (**SessionID** Vi, **PwrSnsrTriggerSlopeEnum** * Slope)

Return the trigger slope or polarity.

When set to positive, trigger events will be generated when a signal rising edge crosses the trigger level threshold. When negative, trigger events are generated on the falling edge of the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Slope	

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigSource()

© 2019 WTG

EXPORT int PwrSnsr_GetTrigSource (**SessionID** Vi, **PwrSnsrTriggerSourceEnum** * Source)

Set the signal the power meter monitors for a trigger. It can be channel external input, or independent.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Source	

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigStatus()

EXPORT int PwrSnsr_GetTrigStatus (**SessionID** Vi, **PwrSnsrTriggerStatusEnum** * Status)

The status of the triggering system. Update rate is controlled by FetchLatency setting.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Status	Status of the trigger.

Returns

Success (0) or error code.

◆PwrSnsr_GetTrigVernier()

EXPORT int PwrSnsr_GetTrigVernier (**SessionID** Vi, float * Vernier

© 2019 WTG

)

Return the fine position of the trigger event on the power sweep.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Vernier	Trigger position -30.0 to 30.0 (0.0 = left, 5.0 = middle, 10.0 = Right).

Returns

Success (0) or error code.

◆PwrSnsr_SetTrigDelay()

EXPORT int PwrSnsr_SetTrigDelay	(	**SessionID**	Vi,
		float	Delay
	)		

Sets the trigger delay time in seconds with respect to the trigger for the trigger display location in the LEFT position.

Positive values cause the actual trigger to occur after the trigger condition is met. This places the trigger event to the left of the trigger point on the display, and is useful for viewing events during a pulse, some fixed delay time after the rising edge trigger. Negative trigger delay places the trigger event to the right of the trigger point on the display, and is useful for looking at events before the trigger edge.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Delay	

Returns

Success (0) or error code.

◆PwrSnsr_SetTrigHoldoff()

EXPORT int PwrSnsr_SetTrigHold	(	**SessionID**	Vi,

© 2019 WTG

off

float Holdoff

)

Sets the trigger holdoff time in seconds.

Trigger holdoff is used to disable the trigger for a specified amount of time after each trigger event. The holdoff time starts immediately after each valid trigger edge, and will not permit any new triggers until the time has expired. When the holdoff time is up, the trigger re-arms, and the next valid trigger event (edge) will cause a new sweep. This feature is used to help synchronize the power meter with burst waveforms such as a TDMA or GSM frame. The trigger holdoff resolution is 10 nanoseconds, and it should be set to a time that is just slightly shorter than the frame repetition interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Holdoff	

Returns

Success (0) or error code.

◆ PwrSnsr_SetTrigHoldoffMode()

EXPORT int PwrSnsr_SetTrigHoldoffMode (**SessionID** Vi, **PwrSnsrHoldoffModeEnum** HoldoffMode)

Sets the holdoff mode to normal or gap holdoff.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
HoldoffMode	Holdoff mode.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetTrigLevel()

EXPORT int PwrSnsr_SetTrigLevel (**SessionID** Vi, float Level)

Set the trigger level for synchronizing data acquisition with a pulsed input signal.

The internal trigger level entered should include any global offset and will also be affected by the frequency cal factor. The available internal trigger level range is sensor dependent. The trigger level is set and returned in dBm. This setting is only valid for normal and auto trigger modes.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Level	Trigger level in dBm.

Returns

Success (0) or error code.

◆PwrSnsr_SetTrigMode()

EXPORT int PwrSnsr_SetTrigMode (**SessionID** Vi, **PwrSnsrTriggerModeEnum** Mode)

Set the trigger mode for synchronizing data acquisition with pulsed signals.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Mode	Trigger mode.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetTrigOutMode()

```
EXPORT int PwrSnsr_SetTrigOutMode (
                                    SessionID      Vi,
                                    const char *   Channel,
                                    int            Mode
                                  )
```

Sets the trigger out/mult io mode. Setting trigger mode overrides this command.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Mode	Trigger out/multi IO mode

Returns

Success (0) or error code.

◆PwrSnsr_SetTrigPosition()

```
EXPORT int PwrSnsr_SetTrigPosition (
                                     SessionID                  Vi,
                                     PwrSnsrTriggerPositionEnum Position
                                   )
```

Set the position of the trigger event on displayed sweep.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Position	Trigger position.

Returns

Success (0) or error code.

◆PwrSnsr_SetTrigSlope()

© 2019 WTG

EXPORT int PwrSnsr_SetTrigSlope (**SessionID** Vi, **PwrSnsrTriggerSlopeEnum** Slope)

Sets the trigger slope or polarity.

When set to positive, trigger events will be generated when a signal rising edge crosses the trigger level threshold. When negative, trigger events are generated on the falling edge of the pulse.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Slope	

Returns

Success (0) or error code.

◆ PwrSnsr_SetTrigSource()

EXPORT int PwrSnsr_SetTrigSource (**SessionID** Vi, **PwrSnsrTriggerSourceEnum** Source)

Get the signal the power meter monitors for a trigger. It can be channel external input, or independent.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Source	

Returns

Success (0) or error code.

© 2019 WTG

◆ PwrSnsr_SetTrigVernier()

EXPORT int PwrSnsr_SetTrigVernier (**SessionID** Vi, float Vernier)

Set the fine position of the trigger event on the power sweep.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Vernier	Trigger position -30.0 to 30.0 (0.0 = left, 5.0 = middle, 10.0 = Right).

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.5 Acquisition

Power Sensor Library 1.1.0

Functions

Acquisition

Functions

EXPORT int	**PwrSnsr_Abort** (**SessionID** Vi)
	Terminates any measurement in progress and resets the state of the trigger system. Note that Abort will leave the measurement in a stopped condition with all current measurements cleared. More...

© 2019 WTG

EXPORT int	**PwrSnsr_FetchExtendedWaveform** (**SessionID** Vi, const char *Channel, int WaveformArrayBufferSize, float WaveformArray[], int *WaveformArrayActualSize, int Count)
	When capture priority is enabled, returns up to 100000 points of trace data based on the current timebase starting at the current trigger delay point. More...
EXPORT int	**PwrSnsr_Clear** (**SessionID** Vi)
	Clear all data buffers. Clears averaging filters to empty. More...
EXPORT int	**PwrSnsr_InitiateAquisition** (**SessionID** Vi)
	Starts a single measurement cycle when INITiate:CONTinuous is set to OFF. More...
EXPORT int	**PwrSnsr_Status** (**SessionID** Vi, **PwrSnsrAcquisitionStatusEnum** *Val)
	Returns whether an acquisition is in progress, complete, or if the status is unknown. More...
EXPORT int	**PwrSnsr_SetInitiateContinuous** (**SessionID** Vi, int InitiateContinuous)
	Set the data acquisition mode for single or free-run measurements. More...
EXPORT int	**PwrSnsr_GetInitiateContinuous** (**SessionID** Vi, int *InitiateContinuous)
	Get the data acquisition mode for single or free-run measurements. More...
EXPORT int	**PwrSnsr_EnableCapturePriority** (**SessionID** Vi, const char *Channel, int Enabled)
	Sets the 55 series power meter to a buffered capture mode and disables real time processing. More...

© 2019 WTG

Detailed Description

Acquisition related functions

Function Documentation

◆ PwrSnsr_Abort()

EXPORT int PwrSnsr_Abort (**SessionID** Vi)

Terminates any measurement in progress and resets the state of the trigger system. Note that Abort will leave the measurement in a stopped condition with all current measurements cleared.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆ PwrSnsr_Clear()

EXPORT int PwrSnsr_Clear (**SessionID** Vi)

Clear all data buffers. Clears averaging filters to empty.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_EnableCapturePriority()

```
EXPORT int PwrSnsr_EnableCapturePriority (
                                          SessionID      Vi,
                                          const char *   Channel,
                                          int            Enabled
                                          )
```

Sets the 55 series power meter to a buffered capture mode and disables real time processing.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Enabled	If set to 1, enables buffered mode. If set to zero, disables capture priority(default).

Returns

Success (0) or error code.

◆PwrSnsr_FetchExtendedWaveform()

```
EXPORT int PwrSnsr_FetchExtendedWaveform (
                                          SessionID      Vi,
                                          const char *   Channel,
                                          int            WaveformArrayBufferSize,
                                          float          WaveformArray[],
                                          int *          WaveformArrayActualSize,
                                          int            Count
                                          )
```

When capture priority is enabled, returns up to 100000 points of trace data based on the current timebase starting at the current trigger delay point.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle

© 2019 WTG

	identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
WaveformArrayBufferSize	Number of elements in the WaveformArray buffer
WaveformArray	Waveform buffer.
WaveformArrayActualSize	Number of elements updated with data.
Count	Number of points to capture.

Returns

Success (0) or error code.

◆PwrSnsr_GetInitiateContinuous()

EXPORT int PwrSnsr_GetInitiateContinuous (**SessionID** Vi, int * InitiateContinuous)

Get the data acquisition mode for single or free-run measurements.

If INITiate:CONTinuous is set to ON, the instrument immediately begins taking measurements (Modulated, CW and Statistical Modes), or arms its trigger and takes a measurement each time a trigger occurs (Pulse Mode). If set to OFF, the measurement will begin (or be armed) as soon as the INITiate command is issued, and will stop once the measurement criteria (averaging, filtering or sample count) has been satisfied. Note that INITiate:IMMediate and READ commands are invalid when INITiate:CONTinuous is set to ON; however, by convention this situation does not result in a SCPI error.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
InitiateContinuous	Boolean. 0 for off or 1 for on.

Returns

Success (0) or error code.

◆PwrSnsr_InitiateAquisition()

© 2019 WTG

EXPORT int PwrSnsr_InitiateAquisition (**SessionID** Vi)

Starts a single measurement cycle when INITiate:CONTinuous is set to OFF.

In Modulated Mode, the measurement will complete once the power has been integrated for the full FILTer time. In Pulse Mode, enough trace sweeps must be triggered to satisfy the AVERaging setting. In Statistical Mode, acquisition stops once the terminal condition(s) are met. In each case, no reading will be returned until the measurement is complete. This command is not valid when INITiate:CONTinuous is ON, however, by convention this situation does not result in a SCPI error

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_SetInitiateContinuous()

EXPORT int PwrSnsr_SetInitiateContinuous (**SessionID** Vi, int InitiateContinuous)

Set the data acquisition mode for single or free-run measurements.

If INITiate:CONTinuous is set to ON, the instrument immediately begins taking measurements (Modulated, CW and Statistical Modes), or arms its trigger and takes a measurement each time a trigger occurs (Pulse Mode). If set to OFF, the measurement will begin (or be armed) as soon as the INITiate command is issued, and will stop once the measurement criteria (averaging, filtering or sample count) has been satisfied. Note that INITiate:IMMediate and READ commands are invalid when INITiate:CONTinuous is set to ON; however, by convention this situation does not result in a SCPI error.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle

© 2019 WTG

	identifies a particular instrument session.
InitiateContinuous	Boolean. 0 for off or 1 for on.

Returns

Success (0) or error code.

◆ PwrSnsr_Status()

EXPORT int PwrSnsr_Status (**SessionID** Vi, **PwrSnsrAcquisitionStatusEnum** * Val)

Returns whether an acquisition is in progress, complete, or if the status is unknown.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Val	Status out parameter.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.6 Channel Functions

Power Sensor Library 1.1.0

Functions

Channel Functions

Functions	
EXPORT int	**PwrSnsr_GetChannelByIndex** (**SessionID** Vi, int BuffSize, char Channel[], int Index)
	Gets the channel name by zero index. Note: SCPI commands use a one-based index.

© 2019 WTG

	More...
EXPORT int	**PwrSnsr_GetEnabled** (**SessionID** Vi, const char *Channel, int *Enabled)
	Get the measurement state of the selected channel. When the value is true, the channel performs measurements; when the value is false, the channel is disabled and no measurements are performed. More...
EXPORT int	**PwrSnsr_SetEnabled** (**SessionID** Vi, const char *Channel, int Enabled)
	Get the measurement state of the selected channel. When the value is true, the channel performs measurements; when the value is false, the channel is disabled and no measurements are performed. More...
EXPORT int	**PwrSnsr_GetSerialNumber** (**SessionID** Vi, const char *Channel, int SerialNumberBufferSize, char SerialNumber[])
	Gets the serial number of the sensor. More...
EXPORT int	**PwrSnsr_GetChannelCount** (**SessionID** Vi, int *Count)
	Get number of channels. More...
EXPORT int	**PwrSnsr_GetUnits** (**SessionID** Vi, const char *Channel, **PwrSnsrUnitsEnum** *Units)
	Get units for the selected channel. More...
EXPORT int	**PwrSnsr_SetUnits** (**SessionID** Vi, const char *Channel, **PwrSnsrUnitsEnum** Units)
	Set units for the selected channel. More...
EXPORT int	**PwrSnsr_GetCurrentTemp** (**SessionID** Vi, const char *Channel, double *CurrentTemp)
	Get current sensor internal temperature in degrees C. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetAverage** (**SessionID** Vi, const char *Channel, int *Average)
	Get the number of traces averaged together to form the measurement result on the selected channel. More...
EXPORT int	**PwrSnsr_SetAverage** (**SessionID** Vi, const char *Channel, int Average)
	Set the number of traces averaged together to form the measurement result on the selected channel. More...
EXPORT int	**PwrSnsr_GetBandwidth** (**SessionID** Vi, const char *Channel, **PwrSnsrBandwidthEnum** *Bandwidth)
	Get the sensor video bandwidth for the selected sensor. More...
EXPORT int	**PwrSnsr_SetBandwidth** (**SessionID** Vi, const char *Channel, **PwrSnsrBandwidthEnum** Bandwidth)
	Set the sensor video bandwidth for the selected sensor. More...
EXPORT int	**PwrSnsr_GetFilterState** (**SessionID** Vi, const char *Channel, **PwrSnsrFilterStateEnum** *FilterState)
	Get the current setting of the integration filter on the selected channel. More...
EXPORT int	**PwrSnsr_SetFilterState** (**SessionID** Vi, const char *Channel, **PwrSnsrFilterStateEnum** FilterState)
	Set the current setting of the integration filter on the selected channel. More...
EXPORT int	**PwrSnsr_GetFilterTime** (**SessionID** Vi, const char *Channel, float *FilterTime)
	Get the current length of the integration filter on the selected channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_SetFilterTime** (**SessionID** Vi, const char *Channel, float FilterTime)
	Set the current length of the integration filter on the selected channel. More...
EXPORT int	**PwrSnsr_GetIsAvgSensor** (**SessionID** Vi, const char *Channel, int *IsAvgSensor)
	Retruns true if sensor is average responding (not peak detecting). More...
EXPORT int	**PwrSnsr_GetDistal** (**SessionID** Vi, const char *Channel, float *Distal)
	Get the pulse amplitude percentage, which is used to define the end of a rising edge or beginning of a falling edge transition. More...
EXPORT int	**PwrSnsr_SetDistal** (**SessionID** Vi, const char *Channel, float Distal)
	Set the pulse amplitude percentage, which is used to define the end of a rising edge or beginning of a falling edge transition. More...
EXPORT int	**PwrSnsr_GetEndGate** (**SessionID** Vi, const char *Channel, float *EndGate)
	Get the point on a pulse, which is used to define the end of the pulse's active interval. More...
EXPORT int	**PwrSnsr_SetEndGate** (**SessionID** Vi, const char *Channel, float EndGate)
	Set the point on a pulse, which is used to define the end of the pulse's active interval. More...
EXPORT int	**PwrSnsr_GetMesial** (**SessionID** Vi, const char *Channel, float *Mesial)
	Get the pulse amplitude percentage, which is used to define the midpoint of a rising edge or end of a falling edge transition. More...
EXPORT int	**PwrSnsr_SetMesial** (**SessionID** Vi, const char *Channel, float Mesial)

© 2019 WTG

	Set the pulse amplitude percentage, which is used to define the midpoint of a rising edge or end of a falling edge transition. More...
EXPORT int	**PwrSnsr_GetProximal** (**SessionID** Vi, const char *Channel, float *Proximal)
	Get the pulse amplitude percentage, which is used to define the beginning of a rising edge or end of a falling edge transition. More...
EXPORT int	**PwrSnsr_SetProximal** (**SessionID** Vi, const char *Channel, float Proximal)
	Set the pulse amplitude percentage, which is used to define the beginning of a rising edge or end of a falling edge transition. More...
EXPORT int	**PwrSnsr_GetPulseUnits** (**SessionID** Vi, const char *Channel, **PwrSnsrPulseUnitsEnum** *Units)
	Get the units for entering the pulse distal, mesial and proximal levels. More...
EXPORT int	**PwrSnsr_SetPulseUnits** (**SessionID** Vi, const char *Channel, **PwrSnsrPulseUnitsEnum** **PwrSnsrPulseUnitsEnum**)
	Set the units for entering the pulse distal, mesial and proximal levels. More...
EXPORT int	**PwrSnsr_GetStartGate** (**SessionID** Vi, const char *Channel, float *StartGate)
	Get the point on a pulse, which is used to define the beginning of the pulse's active interval. More...
EXPORT int	**PwrSnsr_SetStartGate** (**SessionID** Vi, const char *Channel, float StartGate)
	Set the point on a pulse, which is used to define the beginning of the pulse's active interval. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetCalFactors** (**SessionID** Vi, const char *Channel, float *MaxFrequency, float *MinFrequency, int FrequencyListBufferSize, float FrequencyList[], int *FrequencyListActualSize, int CalFactorListBufferSize, float CalFactorList[], int *CalFactorListActualSize, **PwrSnsrBandwidthEnum** Bandwidth)
	Query information associated with calibration factors. More...
EXPORT int	**PwrSnsr_GetCalFactor** (**SessionID** Vi, const char *Channel, float *CalFactor)
	Get the frequency calibration factor currently in use on the selected channel. More...
EXPORT int	**PwrSnsr_SetCalFactor** (**SessionID** Vi, const char *Channel, float CalFactor)
	Set the frequency calibration factor currently in use on the selected channel. More...
EXPORT int	**PwrSnsr_GetFrequency** (**SessionID** Vi, const char *Channel, float *Frequency)
	Get the RF frequency for the current sensor. More...
EXPORT int	**PwrSnsr_SetFrequency** (**SessionID** Vi, const char *Channel, float Frequency)
	Set the RF frequency for the current sensor, and apply the appropriate frequency calibration factor from the sensor internal table. More...
EXPORT int	**PwrSnsr_GetOffsetdB** (**SessionID** Vi, const char *Channel, float *OffsetdB)
	Get a measurement offset in dB for the selected sensor. More...
EXPORT int	**PwrSnsr_SetOffsetdB** (**SessionID** Vi, const char *Channel, float OffsetdB)
	Set a measurement offset in dB for the selected sensor. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetTempComp** (**SessionID** Vi, const char *Channel, int *TempComp)
	Get the state of the peak sensor temperature compensation system. More...
EXPORT int	**PwrSnsr_SetTempComp** (**SessionID** Vi, const char *Channel, int TempComp)
	Set the state of the peak sensor temperature compensation system. More...
EXPORT int	**PwrSnsr_GetModel** (**SessionID** Vi, const char *Channel, int ModelBufferSize, char Model[])
	Gets the model of the meter connected to the specified channel. More...
EXPORT int	**PwrSnsr_GetPeakHoldDecay** (**SessionID** Vi, const char *Channel, int *EnvelopeAverage)
	Get the number of min/max traces averaged together to form the peak hold measurement results on the selected channel. More...
EXPORT int	**PwrSnsr_GetPeakHoldTracking** (**SessionID** Vi, const char *Channel, int *EnvelopeTracking)
	Returns whether peak hold decay automatically tracks trace averaging. If set to true, the peak hold decay and trace averaging values are the same. If set to false, peak hold decay is indepedent. More...
EXPORT int	**PwrSnsr_SetPeakHoldTracking** (**SessionID** Vi, const char *Channel, int EnvelopeTracking)
	Sets whether peak hold decay automatically tracks trace averaging. If set to true, the peak hold decay and trace averaging values are the same. If set to false, peak hold decay is indepedent. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetFirmwareVersion** (**SessionID** Vi, const char *Channel, int FirmwareVersionBufferSize, char FirmwareVersion[])
	Returns the firmware version of the power meter associated with this channel. More...
EXPORT int	**PwrSnsr_SetPeakHoldDecay** (**SessionID** Vi, const char *Channel, int PeakHoldDecay)
	Set the number of min/max traces averaged together to form the peak hold measurement results on the selected channel. More...

Detailed Description

Channel related functions

Function Documentation

◆ PwrSnsr_GetAverage()

EXPORT int PwrSnsr_GetAverage (

SessionID	Vi,
const char *	Channel,
int *	Average

)

Get the number of traces averaged together to form the measurement result on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Average	

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_GetBandwidth()

EXPORT int PwrSnsr_GetBandwidth (

SessionID	Vi,
const char *	Channel,
PwrSnsrBandwidthEnum *	Bandwidth

)

Get the sensor video bandwidth for the selected sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Bandwidth	

Returns

Success (0) or error code.

◆ PwrSnsr_GetCalFactor()

EXPORT int PwrSnsr_GetCalFactor (

SessionID	Vi,
const char *	Channel,
float *	CalFactor

)

Get the frequency calibration factor currently in use on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CalFactor	

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_GetCalFactors()

```
EXPORT int PwrSnsr_GetCalFactors (
        SessionID                   Vi,
        const char *                Channel,
        float *                     MaxFrequency,
        float *                     MinFrequency,
        int                         FrequencyListBufferSize,
        float                       FrequencyList[],
        int *                       FrequencyListActualSize,
        int                         CalFactorListBufferSize,
        float                       CalFactorList[],
        int *                       CalFactorListActualSize,
        PwrSnsrBandwidthEnum        Bandwidth
)
```

Query information associated with calibration factors.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MaxFrequency	Maximum RF frequency measureable by this channel.
MinFrequency	Minimum RF frequency measureable by this channel.
FrequencyListBufferSize	Number of elements in FrequencyList.
FrequencyList	List of frequencies correlated to the cal factors.
FrequencyListActualSize	Actual number of elements returned in FrequencyList.
CalFactorListBufferSize	Number of elements in CalFactorList.
CalFactorList	List of cal factors correlated to the frequencies.

© 2019 WTG

CalFactorListActualSize	Actual number of elements returned in CalFactorList.
Bandwidth	Bandwidth of interest. Cal factors for low and high bandwidth are different.

Returns

Success (0) or error code.

◆PwrSnsr_GetChannelByIndex()

```
EXPORT int PwrSnsr_GetChannelByIndex (
    SessionID Vi,
    int BuffSize,
    char Channel[],
    int Index
)
```

Gets the channel name by zero index. Note: SCPI commands use a one-based index.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
BuffSize	Buffer size for Channel
Channel	Channel number buffer
Index	the index of the channel

Returns

Success (0) or error code.

◆PwrSnsr_GetChannelCount()

```
EXPORT int PwrSnsr_GetChannelCount (
    SessionID Vi,
    int * Count
)
```

Get number of channels.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Count	Number of channels

Returns

Success (0) or error code.

◆PwrSnsr_GetCurrentTemp()

EXPORT int PwrSnsr_GetCurrentTemp (**SessionID** Vi, const char * Channel, double * CurrentTemp)

Get current sensor internal temperature in degrees C.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CurrentTemp	

Returns

Success (0) or error code.

◆PwrSnsr_GetDistal()

EXPORT int PwrSnsr_GetDistal (**SessionID** Vi, const char * Channel, float * Distal)

Get the pulse amplitude percentage, which is used to define the end of a rising edge or beginning of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Distal	

© 2019 WTG

Returns

Success (0) or error code.

◆PwrSnsr_GetEnabled()

EXPORT int PwrSnsr_GetEnabled (

SessionID	Vi,
const char *	Channel,
int *	Enabled

)

Get the measurement state of the selected channel. When the value is true, the channel performs measurements; when the value is false, the channel is disabled and no measurements are performed.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Enabled	Boolean. 1 for enabled; 0 for disabled.

Returns

Success (0) or error code.

◆PwrSnsr_GetEndGate()

EXPORT int PwrSnsr_GetEndGate (

SessionID	Vi,
const char *	Channel,
float *	EndGate

)

Get the point on a pulse, which is used to define the end of the pulse's active interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
EndGate	

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_GetFilterState()

EXPORT int PwrSnsr_GetFilterState (**SessionID** Vi, const char * Channel, **PwrSnsrFilterStateEnum** * FilterState)

Get the current setting of the integration filter on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FilterState	State of the filter.

Returns

Success (0) or error code.

◆ PwrSnsr_GetFilterTime()

EXPORT int PwrSnsr_GetFilterTime (**SessionID** Vi, const char * Channel, float * FilterTime)

Get the current length of the integration filter on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FilterTime	Filter time in seconds.

© 2019 WTG

Returns

Success (0) or error code.

◆PwrSnsr_GetFirmwareVersion()

```
EXPORT int PwrSnsr_GetFirmwareVersion (
                                        SessionID      Vi,
                                        const char *   Channel,
                                        int            FirmwareVersionBufferSize,
                                        char           FirmwareVersion[]
                                      )
```

Returns the firmware version of the power meter associated with this channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FirmwareVersionBufferSize	Size of the FirmwareVersion buffer.
FirmwareVersion	Buffer to hold the firmware version.

Returns

Success (0) or error code.

◆PwrSnsr_GetFrequency()

```
EXPORT int PwrSnsr_GetFrequency (
                                  SessionID      Vi,
                                  const char *   Channel,
                                  float *        Frequency
                                )
```

Get the RF frequency for the current sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
Frequency	RF Frequency in Hz.

Returns

Success (0) or error code.

◆PwrSnsr_GetIsAvgSensor()

EXPORT int PwrSnsr_GetIsAvgSensor (**SessionID** Vi, const char * Channel, int * IsAvgSensor)

Retruns true if sensor is average responding (not peak detecting).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsAvgSensor	True if sensor is average responding.

Returns

Success (0) or error code.

◆PwrSnsr_GetMesial()

EXPORT int PwrSnsr_GetMesial (**SessionID** Vi, const char * Channel, float * Mesial)

Get the pulse amplitude percentage, which is used to define the midpoint of a rising edge or end of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
Mesial	

Returns

Success (0) or error code.

◆PwrSnsr_GetModel()

EXPORT int PwrSnsr_GetModel (**SessionID** Vi, const char * Channel, int ModelBufferSize, char Model[])

Gets the model of the meter connected to the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ModelBufferSize	Size of the buffer..
Model	Buffer where the model is read into.

Returns

Success (0) or error code.

◆PwrSnsr_GetOffsetdB()

EXPORT int PwrSnsr_GetOffsetdB (**SessionID** Vi, const char * Channel, float * OffsetdB)

Get a measurement offset in dB for the selected sensor.

This setting is used to compensate for external couplers, attenuators or amplifiers in the RF signal path ahead of the power sensor.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
OffsetdB	

Returns

Success (0) or error code.

◆PwrSnsr_GetPeakHoldDecay()

EXPORT int PwrSnsr_GetPeakHoldDecay (**SessionID** Vi, const char * Channel, int * EnvelopeAverage)

Get the number of min/max traces averaged together to form the peak hold measurement results on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
EnvelopeAverage	Out parameter value.

Returns

Success (0) or error code.

◆PwrSnsr_GetPeakHoldTracking()

EXPORT int PwrSnsr_GetPeakHoldTracking (**SessionID** Vi, const char * Channel, int * EnvelopeTracking)

© 2019 WTG

Returns whether peak hold decay automatically tracks trace averaging. If set to true, the peak hold decay and trace averaging values are the same. If set to false, peak hold decay is indepedent.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
EnvelopeTracking	Out boolean parameter value.

Returns

Success (0) or error code.

◆PwrSnsr_GetProximal()

```
EXPORT int PwrSnsr_GetProximal (
    SessionID Vi,
    const char * Channel,
    float * Proximal
)
```

Get the pulse amplitude percentage, which is used to define the beginning of a rising edge or end of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Proximal	

Returns

Success (0) or error code.

◆PwrSnsr_GetPulseUnits()

```
EXPORT int PwrSnsr_GetPulseUnits (
    SessionID Vi,
```


© 2019 WTG

	const char *	Channel,
	PwrSnsrPulseUnits Enum *	Units

)

Get the units for entering the pulse distal, mesial and proximal levels.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Units	PwrSnsrPulseUnitsEnum Pulse calculation units (watts or volts).

Returns

Success (0) or error code.

◆ PwrSnsr_GetSerialNumber()

EXPORT int PwrSnsr_GetSerialNumber (

SessionID	Vi,
const char *	Channel,
int	SerialNumberBufferSize,
char	SerialNumber[]

)

Gets the serial number of the sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SerialNumberBufferSize	Size in bytes of Serial number.
SerialNumber	Out parameter. ASCII string serial number.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetStartGate()

EXPORT int PwrSnsr_GetStartGate (**SessionID** Vi, const char * Channel, float * StartGate)

Get the point on a pulse, which is used to define the beginning of the pulse's active interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
StartGate	

Returns

Success (0) or error code.

◆PwrSnsr_GetTempComp()

EXPORT int PwrSnsr_GetTempComp (**SessionID** Vi, const char * Channel, int * TempComp)

Get the state of the peak sensor temperature compensation system.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TempComp	Boolean. 1 for on; 0 for off.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetUnits()

EXPORT int
PwrSnsr_GetUnits (**SessionID** Vi,
const char * Channel,
PwrSnsrUnitsEnum * Units
)

Get units for the selected channel.

Voltage is calculated with reference to the sensor input impedance. Note that for ratiometric results, logarithmic units will always return dBr (dB relative) while linear units return percent.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Units	

Returns

Success (0) or error code.

◆PwrSnsr_SetAverage()

EXPORT int
PwrSnsr_SetAverage (**SessionID** Vi,
const char * Channel,
int Average
)

Set the number of traces averaged together to form the measurement result on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Average	

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_SetBandwidth()

EXPORT int PwrSnsr_SetBandwidth	(	**SessionID**	Vi,
		const char *	Channel,
		PwrSnsrBandwidthEnum	Bandwidth
	)		

Set the sensor video bandwidth for the selected sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Bandwidth	

Returns

Success (0) or error code.

◆ PwrSnsr_SetCalFactor()

EXPORT int PwrSnsr_SetCalFactor	(	**SessionID**	Vi,
		const char *	Channel,
		float	CalFactor
	)		

Set the frequency calibration factor currently in use on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CalFactor	

© 2019 WTG

Returns

Success (0) or error code.

◆PwrSnsr_SetDistal()

EXPORT int
PwrSnsr_SetDistal (**SessionID** Vi,
const char * Channel,
float Distal
)

Set the pulse amplitude percentage, which is used to define the end of a rising edge or beginning of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Distal	

Returns

Success (0) or error code.

◆PwrSnsr_SetEnabled()

EXPORT int
PwrSnsr_SetEnabled (**SessionID** Vi,
const char * Channel,
int Enabled
)

Get the measurement state of the selected channel. When the value is true, the channel performs measurements; when the value is false, the channel is disabled and no measurements are performed.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

© 2019 WTG

Enabled	Boolean. 1 for enable; 0 for disable.

Returns

Success (0) or error code.

◆PwrSnsr_SetEndGate()

EXPORT int PwrSnsr_SetEndGate (

SessionID	Vi,
const char *	Channel,
float	EndGate

)

Set the point on a pulse, which is used to define the end of the pulse's active interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
EndGate	

Returns

Success (0) or error code.

◆PwrSnsr_SetFilterState()

EXPORT int PwrSnsr_SetFilterSta te (

SessionID	Vi,
const char *	Channel,
PwrSnsrFilterStateE num	FilterState

)

Set the current setting of the integration filter on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FilterState	State of the filter.

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_SetFilterTime()

EXPORT int PwrSnsr_SetFilterTime	(	**SessionID**	Vi,
		const char *	Channel,
		float	FilterTime
	)		

Set the current length of the integration filter on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FilterTime	Filter time in seconds.

Returns

Success (0) or error code.

◆ PwrSnsr_SetFrequency()

EXPORT int PwrSnsr_SetFrequency	(	**SessionID**	Vi,
		const char *	Channel,
		float	Frequency
	)		

Set the RF frequency for the current sensor, and apply the appropriate frequency calibration factor from the sensor internal table.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Frequency	RF Frequency in Hz.

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_SetMesial()

EXPORT int PwrSnsr_SetMesial (**SessionID** Vi, const char * Channel, float Mesial)

Set the pulse amplitude percentage, which is used to define the midpoint of a rising edge or end of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Mesial	

Returns

Success (0) or error code.

◆ PwrSnsr_SetOffsetdB()

EXPORT int PwrSnsr_SetOffsetdB (**SessionID** Vi, const char * Channel, float OffsetdB)

Set a measurement offset in dB for the selected sensor.

This setting is used to compensate for external couplers, attenuators or amplifiers in the RF signal path ahead of the power sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
OffsetdB	

Returns

Success (0) or error code.

◆PwrSnsr_SetPeakHoldDecay()

EXPORT int PwrSnsr_SetPeakHoldDecay	(	**SessionID**	Vi,
		const char *	Channel,
		int	PeakHoldDecay
	)		

Set the number of min/max traces averaged together to form the peak hold measurement results on the selected channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PeakHoldDecay	Peak hold decay value.

Returns

Success (0) or error code.

◆PwrSnsr_SetPeakHoldTracking()

EXPORT int PwrSnsr_SetPeakHoldTracking	(	**SessionID**	Vi,
		const char *	Channel,
		int	EnvelopeTracking
	)		

Sets whether peak hold decay automatically tracks trace averaging. If set to true, the peak hold decay and trace averaging values are the same. If set to false, peak hold decay is indepedent.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
EnvelopeTracking	Boolean value. True to set peak hold tracking on.

Returns

Success (0) or error code.

◆PwrSnsr_SetProximal()

EXPORT int PwrSnsr_SetProximal (

SessionID	Vi,
const char *	Channel,
float	Proximal

)

Set the pulse amplitude percentage, which is used to define the beginning of a rising edge or end of a falling edge transition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Proximal	

Returns

Success (0) or error code.

◆PwrSnsr_SetPulseUnits()

EXPORT int PwrSnsr_SetPulseUnits (

SessionID	Vi,
const char *	Channel,
PwrSnsrPulseUnitsEnum	PwrSnsrPulseUnitsEnum

)

Set the units for entering the pulse distal, mesial and proximal levels.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PwrSnsrPulseUnitsEnum	Pulse calculation units (watts or volts).

Returns

Success (0) or error code.

◆PwrSnsr_SetStartGate()

EXPORT int PwrSnsr_SetStartGate (**SessionID** Vi, const char * Channel, float StartGate)

Set the point on a pulse, which is used to define the beginning of the pulse's active interval.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
StartGate	

Returns

Success (0) or error code.

◆PwrSnsr_SetTempComp()

EXPORT int PwrSnsr_SetTempComp (**SessionID** Vi, const char * Channel, int TempComp)

Set the state of the peak sensor temperature compensation system.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TempComp	Boolean. 1 for on; 0 for off.

Returns

Success (0) or error code.

◆ PwrSnsr_SetUnits()

```
EXPORT int
PwrSnsr_SetUnits (
                  SessionID      Vi,
                  const char *   Channel,
                  PwrSnsrUnitsEnum Units
                  )
```

Set units for the selected channel.

Voltage is calculated with reference to the sensor input impedance. Note that for ratiometric results, logarithmic units will always return dBr (dB relative) while linear units return percent.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Units	

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.7 Time Base Functions

Power Sensor Library 1.1.0

Functions

© 2019 WTG

Time Base Functions

Functions

EXPORT int	**PwrSnsr_GetTimebase** (**SessionID** Vi, float *Timebase)
	Get the Pulse Mode timebase in seconds/division. (10 divisions = 1 trace) Value = 5e-9 to 10e-3 (or max timebase) sec in a 1-2-5 sequence,. More...
EXPORT int	**PwrSnsr_SetTimebase** (**SessionID** Vi, float Timebase)
	Set the Pulse Mode timebase in seconds/division. (10 divisions = 1 trace) Value = 5e-9 to 10e-3 sec (or max timebase) in a 1-2-5 sequence,. More...
EXPORT int	**PwrSnsr_SetTimespan** (**SessionID** Vi, float Timespan)
	Set the horizontal time span of the trace in pulse mode. Time span = 10* Time/Division. Value = 5e-8 to 100e-3 sec in a 1-2-5 sequence. More...
EXPORT int	**PwrSnsr_GetTimespan** (**SessionID** Vi, float *Timespan)
	Get the horizontal time span of the trace in pulse mode. Time span = 10* Time/Division. Value = 5e-8 to 100e-3 sec in a 1-2-5 sequence. More...
EXPORT int	**PwrSnsr_GetMaxTimebase** (**SessionID** Vi, float *MaxTimebase)
	Gets the maximum timebase setting available. More...

Detailed Description

© 2019 WTG

Time base and span functions

Function Documentation

◆PwrSnsr_GetMaxTimebase()

EXPORT int PwrSnsr_GetMaxTimebase (**SessionID** Vi, float * MaxTimebase)

Gets the maximum timebase setting available.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MaxTimebase	

Returns

Success (0) or error code.

◆PwrSnsr_GetTimebase()

EXPORT int PwrSnsr_GetTimebase (**SessionID** Vi, float * Timebase)

Get the Pulse Mode timebase in seconds/division. (10 divisions = 1 trace) Value = 5e-9 to 10e-3 (or max timebase) sec in a 1-2-5 sequence,.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Timebase	

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetTimespan()

EXPORT int PwrSnsr_GetTimespan (**SessionID** Vi, float * Timespan)

Get the horizontal time span of the trace in pulse mode. Time span = 10* Time/Division.

Value = 5e-8 to 100e-3 sec in a 1-2-5 sequence.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Timespan	

Returns

Success (0) or error code.

◆PwrSnsr_SetTimebase()

EXPORT int PwrSnsr_SetTimebase (**SessionID** Vi, float Timebase)

Set the Pulse Mode timebase in seconds/division. (10 divisions = 1 trace) Value = 5e-9 to

10e-3 sec (or max timebase) in a 1-2-5 sequence,.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Timebase	

Returns

Success (0) or error code.

◆PwrSnsr_SetTimespan()

© 2019 WTG

EXPORT int PwrSnsr_SetTimespan (**SessionID** Vi, float Timespan)

Set the horizontal time span of the trace in pulse mode. Time span = 10* Time/Division.

Value = 5e-8 to 100e-3 sec in a 1-2-5 sequence.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Timespan	

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.8 Marker Functions

Power Sensor Library 1.1.0

Functions

Marker Functions

Functions	
EXPORT int	**PwrSnsr_GetMarkerTimePosition** (**SessionID** Vi, int MarkerNumber, float *TimePosition)
	Get the time (x-axis-position) of the selected marker relative to the trigger. More...
EXPORT int	**PwrSnsr_SetMarkerTimePosition** (**SessionID** Vi, int MarkerNumber, float TimePosition)
	Set the time (x-axis-position) of the selected marker relative to the trigger. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetMarkerPixelPosition** (**SessionID** Vi, int MarkerNumber, int *PixelPosition)
	Get the horizontal pixel position (X-axis-position) of the selected vertical marker. There are 501 pixel positions numbered from 0 to 500 inclusive. More...
EXPORT int	**PwrSnsr_SetMarkerPixelPosition** (**SessionID** Vi, int MarkerNumber, int PixelPosition)
	Set the horizontal pixel position (X-axis-position) of the selected vertical marker. There are 501 pixel positions numbered from 0 to 500 inclusive. More...
EXPORT int	**PwrSnsr_FetchArrayMarkerPower** (**SessionID** Vi, const char *Channel, float *AvgPower, **PwrSnsrCondCodeEnum** *AvgPowerCondCode, float *MaxPower, **PwrSnsrCondCodeEnum** *MaxPowerCondCode, float *MinPower, **PwrSnsrCondCodeEnum** *MinPowerCondCode, float *PkToAvgRatio, **PwrSnsrCondCodeEnum** *PkToAvgRatioCondCode, float *Marker1Power, **PwrSnsrCondCodeEnum** *Marker1PowerCondCode, float *Marker2Power, **PwrSnsrCondCodeEnum** *Marker2PowerCondCode, float *MarkerRatio, **PwrSnsrCondCodeEnum** *MarkerRatioCondCode)
	Returns an array of the current marker measurements for the specified channel. More...
EXPORT int	**PwrSnsr_FetchMarkerAverage** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Forthe specified marker, return the average power or voltage at the marker. The units are the same as the specified channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_FetchMarkerMax** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Forthe specified marker, return the maximum power or voltage at the marker. The units are the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchMarkerMin** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *IsValid, float *Val)
	Forthe specified marker, return the minimum power or voltage at the marker. The units are the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadArrayMarkerPower** (**SessionID** Vi, const char *Channel, float *AvgPower, **PwrSnsrCondCodeEnum** *AvgPowerCondCode, float *MaxPower, **PwrSnsrCondCodeEnum** *MaxPowerCondCode, float *MinPower, **PwrSnsrCondCodeEnum** *MinPowerCondCode, float *PkToAvgRatio, **PwrSnsrCondCodeEnum** *PkToAvgRatioCondCode, float *Marker1Power, **PwrSnsrCondCodeEnum** *Marker1PowerCondCode, float *Marker2Power, **PwrSnsrCondCodeEnum** *Marker2PowerCondCode, float *MarkerRatio, **PwrSnsrCondCodeEnum** *MarkerRatioCondCode)
	Returns an array of the current marker measurements for the specified channel. More...
EXPORT int	**PwrSnsr_ReadMarkerAverage** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Forthe specified marker, return the average power or voltage at the marker. The units are the same as the specified channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_ReadMarkerMax** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Forthe specified marker, return the maximum power or voltage at the marker. The units are the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadMarkerMin** (**SessionID** Vi, const char *Channel, int Marker, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Forthe specified marker, return the minimum power or voltage at the marker. The units are the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchIntervalAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the average power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchIntervalFilteredMin** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the minmum power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchIntervalFilteredMax** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the maximum filtered power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_FetchIntervalMax** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the maximum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchIntervalMin** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the minimum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchIntervalPkToAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the peak-to-average ratio of the power or voltage between marker 1 and marker 2. The units are dB for logarithmic channel units or percent for linear channel units. More...
EXPORT int	**PwrSnsr_ReadIntervalAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the average power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalFilteredMin** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the minmum power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_ReadIntervalFilteredMax** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the maximum filtered power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalMax** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the maximum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalMin** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the minimum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalPkToAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the peak-to-average ratio of the power or voltage between marker 1 and marker 2. The units are dB for logarithmic channel units or percent for linear channel units. More...
EXPORT int	**PwrSnsr_FetchIntervalMaxAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return maximum of the average power trace between MK1 and MK2. The units will be the same as the specified channel. More...

© 2019 WTG

EXPORT int	**PwrSnsr_FetchIntervalMinAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return minimum of the average power trace between MK1 and MK2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalMaxAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return maximum of the average power trace between MK1 and MK2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_ReadIntervalMinAvg** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return minimum of the average power trace between MK1 and MK2. The units will be the same as the specified channel. More...
EXPORT int	**PwrSnsr_FetchMarkerDelta** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the difference between MK1 and MK2. The units will be the same as marker units. More...
EXPORT int	**PwrSnsr_FetchMarkerRatio** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the ratio of MK1 to MK2. The units will be dB for logarithmic units or percent for linear units. More...
EXPORT int	**PwrSnsr_FetchMarkerRDelta** (**SessionID** Vi, const char *Channel,

© 2019 WTG

	PwrSnsrCondCodeEnum *CondCode, float *Val)
	Return the difference between MK2 and MK1. The units will be the same as marker units. More...
EXPORT int	**PwrSnsr_FetchMarkerRRatio** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the ratio of MK2 to MK1. The units will be dB for logarithmic units or percent for linear units. More...
EXPORT int	**PwrSnsr_ReadMarkerDelta** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the difference between MK1 and MK2. The units will be the same as marker units. More...
EXPORT int	**PwrSnsr_ReadMarkerRatio** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the ratio of MK1 to MK2. The units will be dB for logarithmic units or percent for linear units. More...
EXPORT int	**PwrSnsr_ReadMarkerRDelta** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the difference between MK2 and MK1. The units will be the same as marker units. More...
EXPORT int	**PwrSnsr_ReadMarkerRRatio** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Return the ratio of MK2 to MK1. The units will be dB for logarithmic units or percent for

© 2019 WTG

	linear units. More...

Detailed Description

Marker functions

Function Documentation

◆ PwrSnsr_FetchArrayMarkerPower()

EXPORT int PwrSnsr_FetchArrayMarkerPower (

SessionID	Vi,
const char *	Channel,
float *	AvgPower,
PwrSnsrCondCodeEnum *	AvgPowerCondCode,
float *	MaxPower,
PwrSnsrCondCodeEnum *	MaxPowerCondCode,
float *	MinPower,
PwrSnsrCondCodeEnum *	MinPowerCondCode,
float *	PkToAvgRatio,
PwrSnsrCondCodeEnum *	PkToAvgRatioCondCode,
float *	Marker1Power,
PwrSnsrCondCodeEnum *	Marker1PowerCondCode,
float *	Marker2Power,
PwrSnsrCondCodeEnum *	Marker2PowerCondCode,
float *	MarkerRatio,
PwrSnsrCondCodeEnum *	MarkerRatioCondCode

)

Returns an array of the current marker measurements for the specified channel.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
AvgPower	Average power between the markers.
AvgPowerCondCode	Condition code.
MaxPower	Maximum power between the markers.
MaxPowerCondCode	Condition code.
MinPower	Minimum power between the markers.
MinPowerCondCode	Condition code.
PkToAvgRatio	The ratio of peak to average power between the markers.
PkToAvgRatioCondCode	Condition code.
Marker1Power	The power at Marker 1.
Marker1PowerCondCode	Condition code.
Marker2Power	The power at Marker 2.
Marker2PowerCondCode	Condition code.
MarkerRatio	Ratio of power at Marker 1 and power at Marker 2.
MarkerRatioCondCode	Condition code.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchIntervalAvg()

```
EXPORT int PwrSnsr_FetchIntervalAvg (
                                      SessionID          Vi,
                                      const char *       Channel,
                                      PwrSnsrCondCode
                                      Enum *             CondCode,
                                      float *            Val
                                    )
```

Return the average power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalFilteredMax()

```
EXPORT int
PwrSnsr_FetchInterv
alFilteredMax          (  SessionID                Vi,
                          const char *             Channel,
                          PwrSnsrCondCode
                          Enum *                   CondCode,
                          float *                  Val
                       )
```

Return the maximum filtered power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalFilteredMin()

```
EXPORT int
PwrSnsr_FetchInterv
alFilteredMin          (  SessionID                Vi,
```


© 2019 WTG

		const char *	Channel,
		PwrSnsrCondCodeEnum *	CondCode,
		float *	Val
	)		

Return the minmum power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchIntervalMax()

EXPORT int PwrSnsr_FetchIntervalMax	(	**SessionID**	Vi,
		const char *	Channel,
		PwrSnsrCondCodeEnum *	CondCode,
		float *	Val
	)		

Return the maximum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

© 2019 WTG

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalMaxAvg()

EXPORT int PwrSnsr_FetchIntervalMaxAvg (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * CondCode, float * Val)

Return maximum of the average power trace between MK1 and MK2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalMin()

EXPORT int PwrSnsr_FetchIntervalMin (**SessionID** Vi, const char * Channel, **PwrSnsrCondCodeEnum** * CondCode, float * Val)

Return the minimum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalMinAvg()

```
EXPORT int
PwrSnsr_FetchIntervalMinAvg (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Return minimum of the average power trace between MK1 and MK2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchIntervalPkToAvg()

```
EXPORT int
PwrSnsr_FetchInterv (
    SessionID Vi,
```


© 2019 WTG

alPkToAvg

	const char *	Channel,
	PwrSnsrCondCodeEnum *	CondCode,
	float *	Val

)

Return the peak-to-average ratio of the power or voltage between marker 1 and marker 2.

The units are dB for logarithmic channel units or percent for linear channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchMarkerAverage()

EXPORT int PwrSnsr_FetchMarkerAverage (

SessionID	Vi,
const char *	Channel,
int	Marker,
PwrSnsrCondCodeEnum *	IsValid,
float *	Val

)

Forthe specified marker, return the average power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.

© 2019 WTG

IsValid	Condition code.
Val	Measurement value

Returns

Success (0) or error code.

◆PwrSnsr_FetchMarkerDelta()

```
EXPORT int PwrSnsr_FetchMarkerDelta (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Return the difference between MK1 and MK2. The units will be the same as marker units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchMarkerMax()

```
EXPORT int PwrSnsr_FetchMarkerMax (
    SessionID Vi,
    const char * Channel,
    int Marker,
    PwrSnsrCondCodeEnum * IsValid,
    float * Val
)
```


© 2019 WTG

Forthe specified marker, return the maximum power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.
IsValid	
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchMarkerMin()

```
EXPORT int PwrSnsr_FetchMarkerMin (
                                    SessionID       Vi,
                                    const char *    Channel,
                                    int             Marker,
                                    PwrSnsrCondCodeEnum * IsValid,
                                    float *         Val
                                  )
```

Forthe specified marker, return the minimum power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.
IsValid	
Val	measurement value.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_FetchMarkerRatio()

```
EXPORT int PwrSnsr_FetchMarkerRatio (
                                      SessionID Vi,
                                      const char * Channel,
                                      PwrSnsrCondCodeEnum * CondCode,
                                      float * Val
                                    )
```

Return the ratio of MK1 to MK2. The units will be dB for logarithmic units or percent for linear units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_FetchMarkerRDelta()

```
EXPORT int PwrSnsr_FetchMarkerRDelta (
                                       SessionID Vi,
                                       const char * Channel,
                                       PwrSnsrCondCodeEnum * CondCode,
                                       float * Val
                                     )
```

Return the difference between MK2 and MK1. The units will be the same as marker units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchMarkerRRatio()

EXPORT int PwrSnsr_FetchMarkerRRatio (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCodeEnum *	CondCode,
float *	Val

)

Return the ratio of MK2 to MK1. The units will be dB for logarithmic units or percent for linear units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_GetMarkerPixelPosition()

EXPORT int PwrSnsr_GetMarkerPixelPosition (

SessionID	Vi,
int	MarkerNumber,
int *	PixelPosition

)

© 2019 WTG

Get the horizontal pixel position (X-axis-position) of the selected vertical marker. There are 501 pixel positions numbered from 0 to 500 inclusive.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MarkerNumber	
PixelPosition	

Returns

Success (0) or error code.

◆PwrSnsr_GetMarkerTimePosition()

```
EXPORT int PwrSnsr_GetMarkerTimePosition (
    SessionID Vi,
    int MarkerNumber,
    float * TimePosition
)
```

Get the time (x-axis-position) of the selected marker relative to the trigger.

Note that time markers must be positioned within the time limits of the trace window in the graph display. If a time outside of the display limits is entered, the marker will be placed at the first or last time position as appropriate.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MarkerNumber	
TimePosition	

Returns

Success (0) or error code.

◆PwrSnsr_ReadArrayMarkerPower()

```
EXPORT int PwrSnsr_ReadArrayMarkerPower (
    SessionID Vi,
```


© 2019 WTG

const char *	Channel,
float *	AvgPower,
PwrSnsrCondCodeEnum *	AvgPowerCondCode,
float *	MaxPower,
PwrSnsrCondCodeEnum *	MaxPowerCondCode,
float *	MinPower,
PwrSnsrCondCodeEnum *	MinPowerCondCode,
float *	PkToAvgRatio,
PwrSnsrCondCodeEnum *	PkToAvgRatioCondCode,
float *	Marker1Power,
PwrSnsrCondCodeEnum *	Marker1PowerCondCode,
float *	Marker2Power,
PwrSnsrCondCodeEnum *	Marker2PowerCondCode,
float *	MarkerRatio,
PwrSnsrCondCodeEnum *	MarkerRatioCondCode

)

Returns an array of the current marker measurements for the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
AvgPower	Average power between the markers.
AvgPowerCondCode	Condition code.
MaxPower	Maximum power between the markers.
MaxPowerCondCode	Condition code.
MinPower	Minimum power between the markers.
MinPowerCondCode	Condition code.
PkToAvgRatio	The ratio of peak to average power between the markers.
PkToAvgRatioCondCode	Condition code.
Marker1Power	The power at Marker 1.
Marker1PowerCondCode	Condition code.

© 2019 WTG

Marker2Power	The power at Marker 2.
Marker2PowerCondCode	Condition code.
MarkerRatio	Ratio of power at Marker 1 and power at Marker 2.
MarkerRatioCondCode	Condition code.

Returns

Success (0) or error code.

◆PwrSnsr_ReadIntervalAvg()

```
EXPORT int PwrSnsr_ReadIntervalAvg (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Return the average power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadIntervalFilteredMax()

```
EXPORT int PwrSnsr_ReadIntervalFilteredMax (
    SessionID Vi,
    const char * Channel,
```


© 2019 WTG

		PwrSnsrCondCode Enum *	CondCode,
		float *	Val
)			

Return the maximum filtered power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadIntervalFilteredMin()

EXPORT int PwrSnsr_ReadIntervalFilteredMin	(	**SessionID**	Vi,
		const char *	Channel,
		PwrSnsrCondCode Enum *	CondCode,
		float *	Val
	)		

Return the minmum power or voltage in the time interval between marker 1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_ReadIntervalMax()

```
EXPORT int PwrSnsr_ReadIntervalMax (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Return the maximum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadIntervalMaxAvg()

```
EXPORT int PwrSnsr_ReadIntervalMaxAvg (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    float * Val
)
```

Return maximum of the average power trace between MK1 and MK2. The units will be the same as the specified channel.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadIntervalMin()

EXPORT int PwrSnsr_ReadIntervalMin (
SessionID Vi,
const char * Channel,
PwrSnsrCondCode Enum * CondCode,
float * Val
)

Return the minimum instantaneous power or voltage in the time interval between marker1 and marker 2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadIntervalMinAvg()

EXPORT int PwrSnsr_ReadIntervalMinAvg (
SessionID Vi,

© 2019 WTG

	const char *	Channel,
	PwrSnsrCondCodeEnum *	CondCode,
	float *	Val

)

Return minimum of the average power trace between MK1 and MK2. The units will be the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadIntervalPkToAvg()

EXPORT int PwrSnsr_ReadIntervalPkToAvg (

	SessionID	Vi,
	const char *	Channel,
	PwrSnsrCondCodeEnum *	CondCode,
	float *	Val

)

Return the peak-to-average ratio of the power or voltage between marker 1 and marker 2. The units are dB for logarithmic channel units or percent for linear channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_ReadMarkerAverage()

```
EXPORT int PwrSnsr_ReadMarkerAverage (
        SessionID Vi,
        const char * Channel,
        int Marker,
        PwrSnsrCondCodeEnum * CondCode,
        float * Val
)
```

Forthe specified marker, return the average power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value

Returns

Success (0) or error code.

◆ PwrSnsr_ReadMarkerDelta()

```
EXPORT int PwrSnsr_ReadMarkerDelta (
        SessionID Vi,
        const char * Channel,
        PwrSnsrCondCodeEnum * CondCode,
        float * Val
)
```


© 2019 WTG

Return the difference between MK1 and MK2. The units will be the same as marker units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadMarkerMax()

```
EXPORT int PwrSnsr_ReadMarkerMax (
    SessionID                   Vi,
    const char *                Channel,
    int                         Marker,
    PwrSnsrCondCodeEnum *       CondCode,
    float *                     Val
)
```

Forthe specified marker, return the maximum power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_ReadMarkerMin()

```
EXPORT int PwrSnsr_ReadMarkerMin (
                                  SessionID Vi,
                                  const char * Channel,
                                  int Marker,
                                  PwrSnsrCondCodeEnum * CondCode,
                                  float * Val
                                  )
```

Forthe specified marker, return the minimum power or voltage at the marker. The units are the same as the specified channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Marker	Marker number.
CondCode	Condition code for the measurement. Condition code.
Val	measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadMarkerRatio()

```
EXPORT int PwrSnsr_ReadMarkerRatio (
                                    SessionID Vi,
                                    const char * Channel,
                                    PwrSnsrCondCodeEnum * CondCode,
                                    float * Val
                                    )
```

Return the ratio of MK1 to MK2. The units will be dB for logarithmic units or percent for linear units.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadMarkerRDelta()

EXPORT int PwrSnsr_ReadMarkerRDelta (
SessionID Vi,
const char * Channel,
PwrSnsrCondCode Enum * CondCode,
float * Val
)

Return the difference between MK2 and MK1. The units will be the same as marker units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆PwrSnsr_ReadMarkerRRatio()

EXPORT int PwrSnsr_ReadMarkerRRatio (
SessionID Vi,
const char * Channel,

© 2019 WTG

		PwrSnsrCondCode Enum *	CondCode,
		float *	Val
	)		

Return the ratio of MK2 to MK1. The units will be dB for logarithmic units or percent for linear units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement. Condition code.
Val	Measurement value.

Returns

Success (0) or error code.

◆ PwrSnsr_SetMarkerPixelPosition()

EXPORT int PwrSnsr_SetMarkerPixelPosition	(	**SessionID**	Vi,
		int	MarkerNumber,
		int	PixelPosition
	)		

Set the horizontal pixel position (X-axis-position) of the selected vertical marker. There are 501 pixel positions numbered from 0 to 500 inclusive.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MarkerNumber	
PixelPosition	

Returns

Success (0) or error code.

© 2019 WTG

◆ PwrSnsr_SetMarkerTimePosition()

EXPORT int PwrSnsr_SetMarkerTimePosition	(	**SessionID** Vi, int MarkerNumber, float TimePosition	)

Set the time (x-axis-position) of the selected marker relative to the trigger.

Note that time markers must be positioned within the time limits of the trace window in the graph display. If a time outside of the display limits is entered, the marker will be placed at the first or last time position as appropriate.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MarkerNumber	
TimePosition	

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.9 Display Functions

Power Sensor Library 1.1.0

Functions

Display Functions

Functions

int **EXPORT**	PwrSnsr_GetVerticalCenter (**SessionID** Vi, const char *Channel, float *VerticalCenter)

© 2019 WTG

	Gets vertical center based on current units: arg = (range varies by units) More...
int **EXPORT**	**PwrSnsr_SetVerticalCenter** (**SessionID** Vi, const char *Channel, float VerticalCenter)
	Sets vertical center based on current units: arg = (range varies by units) More...
int **EXPORT**	**PwrSnsr_GetVerticalScale** (**SessionID** Vi, const char *Channel, float *VerticalScale)
	Gets vertical scale based on current units: arg = (range varies by units) More...
int **EXPORT**	**PwrSnsr_SetVerticalScale** (**SessionID** Vi, const char *Channel, float VerticalScale)
	Sets vertical scale based on current units: arg = (range varies by units) More...

Detailed Description

Display functions

Function Documentation

◆ PwrSnsr_GetVerticalCenter()

int **EXPORT** PwrSnsr_GetVerticalCenter (**SessionID** Vi, const char * Channel, float * VerticalCenter)

Gets vertical center based on current units: arg = (range varies by units)

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
VerticalCenter	Vertical center in units

Returns

Success (0) or error code.

◆PwrSnsr_GetVerticalScale()

```
int EXPORT
PwrSnsr_GetVertical
Scale                 (   SessionID          Vi,
                          const char *       Channel,
                          float *            VerticalScale

                      )
```

Gets vertical scale based on current units: arg = (range varies by units)

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
VerticalScale	Vertical scale in units

Returns

Success (0) or error code.

◆PwrSnsr_SetVerticalCenter()

```
int EXPORT
PwrSnsr_SetVertical
Center                (   SessionID          Vi,
                          const char *       Channel,
                          float              VerticalCenter

                      )
```

Sets vertical center based on current units: arg = (range varies by units)

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
VerticalCenter	Vertical center in units

Returns

Success (0) or error code.

◆ PwrSnsr_SetVerticalScale()

```
int EXPORT PwrSnsr_SetVerticalScale (
    SessionID Vi,
    const char * Channel,
    float VerticalScale
)
```

Sets vertical scale based on current units: arg = (range varies by units)

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
VerticalScale	Vertical scale in units

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.10 Statistical Mode

Power Sensor Library 1.1.0

Functions

Statistical Mode

Functions

© 2019 WTG

EXPORT int	**PwrSnsr_GetHorizontalOffset** (**SessionID** Vi, const char *Channel, double *HorizontalOffset)
	Get the statistical mode horizontal scale offset in dB. The offset value will appear at the leftmost edge of the scale with units dBr (decibels relative). More...
EXPORT int	**PwrSnsr_SetHorizontalOffset** (**SessionID** Vi, const char *Channel, double HorizontalOffset)
	Set the statistical mode horizontal scale offset in dB. The offset value will appear at the leftmost edge of the scale with units dBr (decibels relative). More...
EXPORT int	**PwrSnsr_GetHorizontalScale** (**SessionID** Vi, const char *Channel, double *HorizontalScale)
	Get the statistical mode horizontal scale in dB/Div. More...
EXPORT int	**PwrSnsr_SetHorizontalScale** (**SessionID** Vi, const char *Channel, double HorizontalScale)
	Set the statistical mode horizontal scale in dB/Div. More...
EXPORT int	**PwrSnsr_FetchCCDFTrace** (**SessionID** Vi, const char *Channel, int TraceBufferSize, float Trace[], int *TraceActualSize)
	Returns the points in the CCDF trace. More...
EXPORT int	**PwrSnsr_StatModeReset** (**SessionID** Vi, const char *Channel)
	Resets statistical capturing mode by clearing the buffers and restarting the aquisition timer. More...
EXPORT int	**PwrSnsr_FetchStatMeasurementArray** (**SessionID** Vi, const char *Channel, double *Pavg, **PwrSnsrCondCodeEnum** *PavgCond, double *Ppeak,

© 2019 WTG

	PwrSnsrCondCodeEnum *PpeakCond, double *Pmin, **PwrSnsrCondCodeEnum** *PminCond, double *PkToAvgRatio, **PwrSnsrCondCodeEnum** *PkToAvgRatioCond, double *CursorPwr, **PwrSnsrCondCodeEnum** *CursorPwrCond, double *CursorPct, **PwrSnsrCondCodeEnum** *CursorPctCond, double *SampleCount, **PwrSnsrCondCodeEnum** *SampleCountCond, double *SecondsRun, **PwrSnsrCondCodeEnum** *SecondsRunCond)
	Returns an array of the current automatic statistical measurements performed on a sample population. More...
EXPORT int	**PwrSnsr_FetchCCDFPower** (**SessionID** Vi, const char *Channel, double Percent, **PwrSnsrCondCodeEnum** *CondCode, double *Val)
	Return relative power (in dB) for a given percent on the CCDF plot. More...
EXPORT int	**PwrSnsr_FetchCCDFPercent** (**SessionID** Vi, const char *Channel, double Power, **PwrSnsrCondCodeEnum** *CondCode, double *Val)
	Return relative power (in dB) for a given percent on the CCDF plot. More...
EXPORT int	**PwrSnsr_GetCapture** (**SessionID** Vi, const char *Channel, int *Capture)
	Get whether statistical capture is enabled. More...
EXPORT int	**PwrSnsr_SetCapture** (**SessionID** Vi, const char *Channel, int Capture)
	Set whether statistical capture is enabled. More...
EXPORT int	**PwrSnsr_GetGating** (**SessionID** Vi, const char *Channel, **PwrSnsrStatGatingEnum** *Gating)

© 2019 WTG

	Get whether statistical capture is enabled. More...
EXPORT int	**PwrSnsr_SetGating** (**SessionID** Vi, const char *Channel, **PwrSnsrStatGatingEnum** Gating)
	Set whether the statical capture is gated by markers or free-running. More...
EXPORT int	**PwrSnsr_GetTermAction** (**SessionID** Vi, const char *Channel, **PwrSnsrTermActionEnum** *TermAction)
	Get the termination action for statistical capturing. More...
EXPORT int	**PwrSnsr_SetTermAction** (**SessionID** Vi, const char *Channel, **PwrSnsrTermActionEnum** TermAction)
	Set the termination action for statistical capturing. More...
EXPORT int	**PwrSnsr_GetTermCount** (**SessionID** Vi, const char *Channel, double *TermCount)
	Get the termination count for statistical capturing. After the sample count has been reached, the action determined by TermAction is taken. More...
EXPORT int	**PwrSnsr_SetTermCount** (**SessionID** Vi, const char *Channel, double TermCount)
	Set the termination count for statistical capturing. After the sample count has been reached, the action determined by TermAction is taken. More...
EXPORT int	**PwrSnsr_GetTermTime** (**SessionID** Vi, const char *Channel, int *TermTime)
	Get the termination time in seconds for statistical capturing. After the time has elapsed, the action determined by TermAction is taken. More...

© 2019 WTG

EXPORT int	**PwrSnsr_SetTermTime** (**SessionID** Vi, const char *Channel, int TermTime)
	Set the termination time in seconds (1 - 3600) for statistical capturing. After the time has elapsed, the action determined by TermAction is taken. More...
EXPORT int	**PwrSnsr_GetCCDFTraceCount** (**SessionID** Vi, const char *Channel, int *TraceCount)
	Get the number of points in the CCDF trace plot. More...
EXPORT int	**PwrSnsr_SetCCDFTraceCount** (**SessionID** Vi, const char *Channel, int TraceCount)
	Set the number of points (1 - 16384) in the CCDF trace plot. More...
EXPORT int	**PwrSnsr_FetchCursorPercent** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, double *Val)
	Returns the percent CCDF at the cursor. More...
EXPORT int	**PwrSnsr_FetchCursorPower** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, double *Val)
	Returns the power CCDF in dB at the cursor. More...
EXPORT int	**PwrSnsr_GetPercentPosition** (**SessionID** Vi, const char *Channel, double *PercentPosition)
	Get the cursor percent on the CCDF plot. More...
EXPORT int	**PwrSnsr_SetPercentPosition** (**SessionID** Vi, const char *Channel, double PercentPosition)

© 2019 WTG

	Set the cursor percent on the CCDF plot. More...
EXPORT int	**PwrSnsr_SetPowerPosition** (**SessionID** Vi, const char *Channel, double PowerPosition)
	Set the cursor power in dB on the CCDF plot. More...
EXPORT int	**PwrSnsr_GetPowerPosition** (**SessionID** Vi, const char *Channel, double *PowerPosition)
	Get the cursor power in dB on the CCDF plot. More...
EXPORT int	**PwrSnsr_GetAcqStatusArray** (**SessionID** Vi, const char *Channel, int *SweepLength, double *SampleRate, double *SweepRate, double *SweepTime, double *StartTime, int *StatusWord)
	Returns data about the status of the acquisition system. More...
EXPORT int	**PwrSnsr_GetDiagStatusArray** (**SessionID** Vi, const char *Channel, float *DetectorTemp, float *CpuTemp, float *MioVoltage, float *VccInt10, float *VccAux18, float *Vcc50, float *Vcc25, float *Vcc33)
	Returns diagnostic data. More...

Detailed Description

Statmode functions

Function Documentation

◆ PwrSnsr_FetchCCDFPercent()

© 2019 WTG

EXPORT int PwrSnsr_FetchCCDFPercent (**SessionID** Vi, const char * Channel, double Power, **PwrSnsrCondCodeEnum** * CondCode, double * Val)

Return relative power (in dB) for a given percent on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Power	Relative power in dB
CondCode	Condition code for the measurement.
Val	Percent measurement at power.

Returns

Success (0) or error code.

◆ PwrSnsr_FetchCCDFPower()

EXPORT int PwrSnsr_FetchCCDFPower (**SessionID** Vi, const char * Channel, double Percent, **PwrSnsrCondCodeEnum** * CondCode, double * Val)

Return relative power (in dB) for a given percent on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

© 2019 WTG

Percent	Statistical percent to retrieve power from.
CondCode	Condition code for the measurement.
Val	relative power at percent.

Returns

Success (0) or error code.

◆PwrSnsr_FetchCCDFTrace()

```
EXPORT int PwrSnsr_FetchCCDFTrace (
    SessionID       Vi,
    const char *    Channel,
    int             TraceBufferSize,
    float           Trace[],
    int *           TraceActualSize
)
```

Returns the points in the CCDF trace.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TraceBufferSize	
Trace	
TraceActualSize	

Returns

Success (0) or error code.

◆PwrSnsr_FetchCursorPercent()

```
EXPORT int PwrSnsr_FetchCursorPercent (
    SessionID                   Vi,
    const char *                Channel,
    PwrSnsrCondCodeEnum *       CondCode,
    double *                    Val
)
```


© 2019 WTG

Returns the percent CCDF at the cursor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	

Returns

Success (0) or error code.

◆PwrSnsr_FetchCursorPower()

```
EXPORT int PwrSnsr_FetchCursorPower (
    SessionID Vi,
    const char * Channel,
    PwrSnsrCondCodeEnum * CondCode,
    double * Val
)
```

Returns the power CCDF in dB at the cursor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	

Returns

Success (0) or error code.

◆PwrSnsr_FetchStatMeasurementArray()

```
EXPORT int PwrSnsr_FetchStatMeasurementArray (
    SessionID Vi,
```


© 2019 WTG

```
const char *                    Channel,
double *                        Pavg,
PwrSnsrCondCodeEnum *           PavgCond,
double *                        Ppeak,
PwrSnsrCondCodeEnum *           PpeakCond,
double *                        Pmin,
PwrSnsrCondCodeEnum *           PminCond,
double *                        PkToAvgRatio,
PwrSnsrCondCodeEnum *           PkToAvgRatioCond,
double *                        CursorPwr,
PwrSnsrCondCodeEnum *           CursorPwrCond,
double *                        CursorPct,
PwrSnsrCondCodeEnum *           CursorPctCond,
double *                        SampleCount,
PwrSnsrCondCodeEnum *           SampleCountCond,
double *                        SecondsRun,
PwrSnsrCondCodeEnum *           SecondsRunCond
)
```

Returns an array of the current automatic statistical measurements performed on a sample population.

Measurements performed are: long term average, peak and minimum amplitude, peak-to-average ratio, amplitude at the CCDF percent cursor, statistical percent at the CCDF power cursor, and the sample population size in samples. Note the peak-to-average ratio is returned in dB for logarithmic channel units, and percent for all other channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Pavg	Long term average power in channel units.

© 2019 WTG

PavgCond	Condition code.
Ppeak	Peak power in channel units.
PpeakCond	Condition code.
Pmin	Minimum power in channel units.
PminCond	Condition code.
PkToAvgRatio	Peak-to-average power in percent or dB.
PkToAvgRatioCond	Condition code.
CursorPwr	Power at the cursor in channel units.
CursorPwrCond	Condition code.
CursorPct	Statistical percent at the cursor.
CursorPctCond	Condition code.
SampleCount	Population size in samples.
SampleCountCond	Condition code.
SecondsRun	Number of seconds the measurement has run.
SecondsRunCond	Condition code.

Returns

Success (0) or error code.

◆ PwrSnsr_GetAcqStatusArray()

```
EXPORT int PwrSnsr_GetAcqStatusArray (
                                      SessionID      Vi,
                                      const char *   Channel,
                                      int *          SweepLength,
                                      double *       SampleRate,
                                      double *       SweepRate,
                                      double *       SweepTime,
                                      double *       StartTime,
                                      int *          StatusWord
                                      )
```

Returns data about the status of the acquisition system.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

© 2019 WTG

SweepLength	Returns the number of points in the trace.
SampleRate	Returns the sample rate.
SweepRate	Returns the number of triggered sweeps per second.
SweepTime	Returns the sweep time for the trace.
StartTime	Returns the start time relative to the trigger.
StatusWord	Internal use - acquisition system status word.

Returns

Success (0) or error code.

◆ PwrSnsr_GetCapture()

EXPORT int PwrSnsr_GetCapture (

SessionID	Vi,
const char *	Channel,
int *	Capture

)

Get whether statistical capture is enabled.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Capture	

Returns

Success (0) or error code.

◆ PwrSnsr_GetCCDFTraceCount()

EXPORT int PwrSnsr_GetCCDFTraceCount (

SessionID	Vi,
const char *	Channel,
int *	TraceCount

)

Get the number of points in the CCDF trace plot.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TraceCount	

Returns

Success (0) or error code.

◆ PwrSnsr_GetDiagStatusArray()

```
EXPORT int PwrSnsr_GetDiagStatusArray (
        SessionID Vi,
        const char * Channel,
        float * DetectorTemp,
        float * CpuTemp,
        float * MioVoltage,
        float * VccInt10,
        float * VccAux18,
        float * Vcc50,
        float * Vcc25,
        float * Vcc33
)
```

Returns diagnostic data.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
DetectorTemp	Temperature in degrees C at the RF detector.
CpuTemp	Temperature of the CPU in degrees C.
MioVoltage	Voltage at the Multi I/O port.
VccInt10	Vcc 10 voltage.
VccAux18	Vcc Aux 18 voltage.
Vcc50	Vcc 50 voltage.
Vcc25	Vcc 25 voltage.
Vcc33	Vcc 33 voltage.

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_GetGating()

EXPORT int PwrSnsr_GetGating (**SessionID** Vi, const char * Channel, **PwrSnsrStatGatingEnum** * Gating)

Get whether statistical capture is enabled.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1". whether the statical capture is gated by markers or free-running.
Gating	

Returns

Success (0) or error code.

◆ PwrSnsr_GetHorizontalOffset()

EXPORT int PwrSnsr_GetHorizontalOffset (**SessionID** Vi, const char * Channel, double * HorizontalOffset)

Get the statistical mode horizontal scale offset in dB. The offset value will appear at the leftmost edge of the scale with units dBr (decibels relative).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
HorizontalOffset	

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_GetHorizontalScale()

EXPORT int PwrSnsr_GetHorizontalScale (**SessionID** Vi, const char * Channel, double * HorizontalScale)

Get the statistical mode horizontal scale in dB/Div.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
HorizontalScale	

Returns

Success (0) or error code.

◆ PwrSnsr_GetPercentPosition()

EXPORT int PwrSnsr_GetPercentPosition (**SessionID** Vi, const char * Channel, double * PercentPosition)

Get the cursor percent on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1". Channel Channel number. For single instruments, set this to 1.
PercentPosition	

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_GetPowerPosition()

EXPORT int PwrSnsr_GetPowerPosition (

SessionID	Vi,
const char *	Channel,
double *	PowerPosition

)

Get the cursor power in dB on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PowerPosition	

Returns

Success (0) or error code.

◆ PwrSnsr_GetTermAction()

EXPORT int PwrSnsr_GetTermAction (

SessionID	Vi,
const char *	Channel,
PwrSnsrTermActionEnum *	TermAction

)

Get the termination action for statistical capturing.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TermAction	

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_GetTermCount()

EXPORT int PwrSnsr_GetTermCount	(	**SessionID**	Vi,
		const char *	Channel,
		double *	TermCount
	)		

Get the termination count for statistical capturing. After the sample count has been reached, the action determined by TermAction is taken.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TermCount	

Returns

Success (0) or error code.

◆ PwrSnsr_GetTermTime()

EXPORT int PwrSnsr_GetTermTime	(	**SessionID**	Vi,
		const char *	Channel,
		int *	TermTime
	)		

Get the termination time in seconds for statistical capturing. After the time has elapsed, the action determined by TermAction is taken.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

© 2019 WTG

TermTime

Returns

Success (0) or error code.

◆PwrSnsr_SetCapture()

EXPORT int PwrSnsr_SetCapture (**SessionID** Vi, const char * Channel, int Capture)

Set whether statistical capture is enabled.

Parameters

Vi The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Channel Channel number. For single instruments, set this to "CH1".

Capture

Returns

Success (0) or error code.

◆PwrSnsr_SetCCDFTraceCount()

EXPORT int PwrSnsr_SetCCDFTraceCount (**SessionID** Vi, const char * Channel, int TraceCount)

Set the number of points (1 - 16384) in the CCDF trace plot.

Parameters

Vi The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Channel Channel number. For single instruments, set this to "CH1".

TraceCount

Returns

© 2019 WTG

Success (0) or error code.

◆PwrSnsr_SetGating()

EXPORT int PwrSnsr_SetGating (**SessionID** Vi, const char * Channel, **PwrSnsrStatGatingEnum** Gating)

Set whether the statical capture is gated by markers or free-running.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Gating	

Returns

Success (0) or error code.

◆PwrSnsr_SetHorizontalOffset()

EXPORT int PwrSnsr_SetHorizontalOffset (**SessionID** Vi, const char * Channel, double HorizontalOffset)

Set the statistical mode horizontal scale offset in dB. The offset value will appear at the leftmost edge of the scale with units dBr (decibels relative).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
HorizontalOffset	

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_SetHorizontalScale()

```
EXPORT int PwrSnsr_SetHorizontalScale (
    SessionID Vi,
    const char * Channel,
    double HorizontalScale
)
```

Set the statistical mode horizontal scale in dB/Div.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
HorizontalScale	

Returns

Success (0) or error code.

◆ PwrSnsr_SetPercentPosition()

```
EXPORT int PwrSnsr_SetPercentPosition (
    SessionID Vi,
    const char * Channel,
    double PercentPosition
)
```

Set the cursor percent on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PercentPosition	

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetPowerPosition()

EXPORT int PwrSnsr_SetPowerPosition (**SessionID** Vi, const char * Channel, double PowerPosition)

Set the cursor power in dB on the CCDF plot.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PowerPosition	

Returns

Success (0) or error code.

◆PwrSnsr_SetTermAction()

EXPORT int PwrSnsr_SetTermAction (**SessionID** Vi, const char * Channel, **PwrSnsrTermActionEnum** TermAction)

Set the termination action for statistical capturing.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TermAction	

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetTermCount()

EXPORT int PwrSnsr_SetTermCount (**SessionID** Vi, const char * Channel, double TermCount)

Set the termination count for statistical capturing. After the sample count has been reached, the action determined by TermAction is taken.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TermCount	

Returns

Success (0) or error code.

◆PwrSnsr_SetTermTime()

EXPORT int PwrSnsr_SetTermTime (**SessionID** Vi, const char * Channel, int TermTime)

Set the termination time in seconds (1 - 3600) for statistical capturing. After the time has elapsed, the action determined by TermAction is taken.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TermTime	

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_StatModeReset()

EXPORT int PwrSnsr_StatModeReset (**SessionID** Vi, const char * Channel)

Resets statistical capturing mode by clearing the buffers and restarting the aquisition timer.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.11 Sensor Info

Power Sensor Library 1.1.0

Functions

Sensor Info

Functions

EXPORT int	**PwrSnsr_GetManufactureDate** (**SessionID** Vi, const char *Channel, int ManufactureDateBufferSize, char ManufactureDate[])
	Date the sensor was manufactured in the following format YYYYmmDD. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetImpedance** (**SessionID** Vi, const char *Channel, float *Impedance)
	Input impedance of the sensor. More...
EXPORT int	**PwrSnsr_GetPeakPowerMax** (**SessionID** Vi, const char *Channel, float *PeakPowerMax)
	Maximum power level the sensor can measure. More...
EXPORT int	**PwrSnsr_GetPeakPowerMin** (**SessionID** Vi, const char *Channel, float *PeakPowerMin)
	Minimum power level the sensor can measure. More...
EXPORT int	**PwrSnsr_GetAttenuation** (**SessionID** Vi, const char *Channel, float *Attenuation)
	Attenuation in dB for the sensor. More...
EXPORT int	**PwrSnsr_GetFactoryCalDate** (**SessionID** Vi, const char *Channel, int FactoryCalDateBufferSize, char FactoryCalDate[])
	The date (YYYYmmDD) the last time the sensor was calibrated at the factory. More...
EXPORT int	**PwrSnsr_GetMinimumTrig** (**SessionID** Vi, const char *Channel, float *MinimumTrig)
	Minimum internal trigger level in dBm. More...
EXPORT int	**PwrSnsr_GetMinFreqHighBandwidth** (**SessionID** Vi, const char *Channel, float *MinFreqHighBandwidth)
	Minimum frequency of RF the sensor can measure in high bandwidth. More...
EXPORT int	**PwrSnsr_GetMaxFreqHighBandwidth** (**SessionID** Vi, const char *Channel, float *MaxFreqHighBandwidth)
	Maximum frequency carrier the sensor can measure in high bandwidth. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetMinFreqLowBandwidth** (**SessionID** Vi, const char *Channel, float *MinFreqLowBandwidth)
	Minimum frequency carrier the sensor can measure in low bandwidth. More...
EXPORT int	**PwrSnsr_GetMaxFreqLowBandwidth** (**SessionID** Vi, const char *Channel, float *MaxFreqLowBandwidth)
	Maximum frequency carrier the sensor can measure in low bandwidth. More...
EXPORT int	**PwrSnsr_GetFpgaVersion** (**SessionID** Vi, const char *Channel, int ValBufferSize, char Val[])
	Get the sensor FPGA version. More...

Detailed Description

Sensor info functions

Function Documentation

◆ PwrSnsr_GetAttenuation()

EXPORT int PwrSnsr_GetAttenuation (

SessionID	Vi,
const char *	Channel,
float *	Attenuation

)

Attenuation in dB for the sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
Attenuation	

Returns

Success (0) or error code.

◆PwrSnsr_GetFactoryCalDate()

EXPORT int PwrSnsr_GetFactoryCalDate	(	**SessionID**	Vi,
		const char *	Channel,
		int	FactoryCalDateBufferSize,
		char	FactoryCalDate[]
	)		

The date (YYYYmmDD) the last time the sensor was calibrated at the factory.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
FactoryCalDateBufferSize	Size of FactoryCalDate in bytes.
FactoryCalDate	

Returns

Success (0) or error code.

◆PwrSnsr_GetFpgaVersion()

EXPORT int PwrSnsr_GetFpgaVersion	(	**SessionID**	Vi,
		const char *	Channel,
		int	ValBufferSize,
		char	Val[]
	)		

Get the sensor FPGA version.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size pf Val in bytes
Val	Buffer for staoring the version

Returns

Success (0) or error code.

◆PwrSnsr_GetImpedance()

EXPORT int PwrSnsr_GetImpedance (

SessionID	Vi,
const char *	Channel,
float *	Impedance

)

Input impedance of the sensor.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Impedance	

Returns

Success (0) or error code.

◆PwrSnsr_GetManufactureDate()

EXPORT int PwrSnsr_GetManufactureDate (

SessionID	Vi,
const char *	Channel,
int	ManufactureDateBufferSize,
char	ManufactureDate[]

)

Date the sensor was manufactured in the following format YYYYmmDD.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ManufactureDateBufferSize	Size of ManufactureDate in bytes.
ManufactureDate	Return value.

Returns

Success (0) or error code.

◆PwrSnsr_GetMaxFreqHighBandwidth()

EXPORT int PwrSnsr_GetMaxFreqHighBandwidth (

SessionID	Vi,
const char *	Channel,
float *	MaxFreqHighBandwidth

)

Maximum frequency carrier the sensor can measure in high bandwidth.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MaxFreqHighBandwidth	

Returns

Success (0) or error code.

◆PwrSnsr_GetMaxFreqLowBandwidth()

EXPORT int PwrSnsr_GetMaxFreqLowBandwidth (

SessionID	Vi,
const char *	Channel,
float *	MaxFreqLowBandwidth

)

© 2019 WTG

Maximum frequency carrier the sensor can measure in low bandwidth.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MaxFreqLowBandwidth	

Returns

Success (0) or error code.

◆ PwrSnsr_GetMinFreqHighBandwidth()

```
EXPORT int PwrSnsr_GetMinFreqHighBandwidth (
        SessionID       Vi,
        const char *    Channel,
        float *         MinFreqHighBandwidth
)
```

Minimum frequency of RF the sensor can measure in high bandwidth.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MinFreqHighBandwidth	

Returns

Success (0) or error code.

◆ PwrSnsr_GetMinFreqLowBandwidth()

```
EXPORT int PwrSnsr_GetMinFreqLowBandwidth (
        SessionID       Vi,
        const char *    Channel,
        float *         MinFreqLowBandwidth
)
```


© 2019 WTG

Minimum frequency carrier the sensor can measure in low bandwidth.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MinFreqLowBandwidth	

Returns

Success (0) or error code.

◆PwrSnsr_GetMinimumTrig()

```
EXPORT int
PwrSnsr_GetMinimu
mTrig                (
                       SessionID      Vi,
                       const char *   Channel,
                       float *        MinimumTrig

                     )
```

Minimum internal trigger level in dBm.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
MinimumTrig	

Returns

Success (0) or error code.

◆PwrSnsr_GetPeakPowerMax()

```
EXPORT int
PwrSnsr_GetPeakPo
werMax               (
                       SessionID      Vi,
                       const char *   Channel,
                       float *        PeakPowerMax

                     )
```

Maximum power level the sensor can measure.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PeakPowerMax	

Returns

Success (0) or error code.

◆ PwrSnsr_GetPeakPowerMin()

EXPORT int PwrSnsr_GetPeakPowerMin (**SessionID** Vi, const char * Channel, float * PeakPowerMin)

Minimum power level the sensor can measure.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PeakPowerMin	

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.12 User Calibration

Power Sensor Library 1.1.0

Functions

© 2019 WTG

User Calibration

Functions

EXPORT int	**PwrSnsr_SaveUserCal** (**SessionID** Vi, const char *Channel)
	Instructs power meter to save the value of fixed cal, zero, and skew values. More...
EXPORT int	**PwrSnsr_ClearUserCal** (**SessionID** Vi, const char *Channel)
	Resets the value of fixed cal, zero, and skew to factory defaults. More...
EXPORT int	**PwrSnsr_GetExternalSkew** (**SessionID** Vi, const char *Channel, float *External)
	Gets the skew in seconds for the external trigger. More...
EXPORT int	**PwrSnsr_SetExternalSkew** (**SessionID** Vi, const char *Channel, float External)
	Sets the skew in seconds for the external trigger. More...
EXPORT int	**PwrSnsr_GetSlaveSkew** (**SessionID** Vi, const char *Channel, float *SlaveSkew)
	Gets the skew in seconds for the slave trigger. More...
EXPORT int	**PwrSnsr_SetSlaveSkew** (**SessionID** Vi, const char *Channel, float SlaveSkew)
	Sets the skew in seconds for the slave trigger. More...
EXPORT int	**PwrSnsr_GetInternalSkew** (**SessionID** Vi, const char *Channel, float *InternalSkew)
	Gets the skew in seconds for the internal trigger. More...

© 2019 WTG

EXPORT int	**PwrSnsr_SetInternalSkew** (**SessionID** Vi, const char *Channel, float InternalSkew)
	Sets the skew in seconds for the internal trigger. More...
EXPORT int	**PwrSnsr_Zero** (**SessionID** Vi, const char *Channel)
	Performs a zero offset null adjustment. More...
EXPORT int	**PwrSnsr_ZeroQuery** (**SessionID** Vi, const char *Channel, int *Val)
	Performs a zero offset null adjustment and returns true if successful. More...

Detailed Description

User calibration functions

Function Documentation

◆ PwrSnsr_ClearUserCal()

EXPORT int PwrSnsr_ClearUserCal (**SessionID** Vi, const char * Channel)

Resets the value of fixed cal, zero, and skew to factory defaults.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_GetExternalSkew()

```
EXPORT int PwrSnsr_GetExternalSkew (
    SessionID Vi,
    const char * Channel,
    float * External
)
```

Gets the skew in seconds for the external trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
External	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

◆ PwrSnsr_GetInternalSkew()

```
EXPORT int PwrSnsr_GetInternalSkew (
    SessionID Vi,
    const char * Channel,
    float * InternalSkew
)
```

Gets the skew in seconds for the internal trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
InternalSkew	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetSlaveSkew()

```
EXPORT int
PwrSnsr_GetSlaveSk
ew                      (   SessionID       Vi,
                            const char *    Channel,
                            float *         SlaveSkew
                        )
```

Gets the skew in seconds for the slave trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SlaveSkew	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

◆PwrSnsr_SaveUserCal()

```
EXPORT int
PwrSnsr_SaveUserC
al                      (   SessionID       Vi,
                            const char *    Channel
                        )
```

Instructs power meter to save the value of fixed cal, zero, and skew values.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

Returns

Success (0) or error code.

◆PwrSnsr_SetExternalSkew()

© 2019 WTG

EXPORT int PwrSnsr_SetExternalSkew (**SessionID** Vi, const char * Channel, float External)

Sets the skew in seconds for the external trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
External	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

◆ PwrSnsr_SetInternalSkew()

EXPORT int PwrSnsr_SetInternalSkew (**SessionID** Vi, const char * Channel, float InternalSkew)

Sets the skew in seconds for the internal trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
InternalSkew	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

◆ PwrSnsr_SetSlaveSkew()

© 2019 WTG

EXPORT int PwrSnsr_SetSlaveSkew (**SessionID** Vi, const char * Channel, float SlaveSkew)

Sets the skew in seconds for the slave trigger.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SlaveSkew	Trigger skew in seconds (-1e-6 to 1e-6).

Returns

Success (0) or error code.

◆PwrSnsr_Zero()

EXPORT int PwrSnsr_Zero (**SessionID** Vi, const char * Channel)

Performs a zero offset null adjustment.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

Returns

Success (0) or error code.

◆PwrSnsr_ZeroQuery()

EXPORT int PwrSnsr_ZeroQuery (**SessionID** Vi, const char * Channel, int * Val

© 2019 WTG

)

Performs a zero offset null adjustment and returns true if successful.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Val	Boolean value for operation success or failure.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.13 Trace Functions

Power Sensor Library 1.1.0

Functions

Trace Functions

Functions

EXPORT int	**PwrSnsr_GetTimePerPoint** (**SessionID** Vi, const char *Channel, float *TimePerPoint)
	Get time spacing for each waveform point in seconds. More...
EXPORT int	**PwrSnsr_GetSweepTime** (**SessionID** Vi, const char *Channel, float *SweepTime)
	Get sweep time for the trace in seconds. More...
EXPORT int	**PwrSnsr_GetChanTraceCount** (**SessionID** Vi, const char *Channel, int *TraceCount)

© 2019 WTG

	Get the number of points in the CCDF trace plot. More...
EXPORT int	**PwrSnsr_GetTraceStartTime** (**SessionID** Vi, const char *Channel, float *TraceStartTime)
	Get time offset (start time) of the trace in seconds. May be negative, indicating pre-trigger information. More...
EXPORT int	**PwrSnsr_FetchDistal** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the actual detected power of the distal level in the current channel units. More...
EXPORT int	**PwrSnsr_FetchMesial** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the actual detected power of the mesial level in the current channel units. More...
EXPORT int	**PwrSnsr_FetchProximal** (**SessionID** Vi, const char *Channel, **PwrSnsrCondCodeEnum** *CondCode, float *Val)
	Returns the actual detected power of the proximal level in the current channel units. More...

Detailed Description

Trace functions

Function Documentation

© 2019 WTG

◆PwrSnsr_FetchDistal()

EXPORT int
PwrSnsr_FetchDistal (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	CondCode,
float *	Val

)

Returns the actual detected power of the distal level in the current channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	Detected power of the distal level in the current channel units.

Returns

Success (0) or error code.

◆PwrSnsr_FetchMesial()

EXPORT int
PwrSnsr_FetchMesial (

SessionID	Vi,
const char *	Channel,
PwrSnsrCondCode Enum *	CondCode,
float *	Val

)

Returns the actual detected power of the mesial level in the current channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.

© 2019 WTG

Val	Detected power of the mesial level in the current channel units.

Returns

Success (0) or error code.

◆PwrSnsr_FetchProximal()

EXPORT int PwrSnsr_FetchProximal (**SessionID** Vi, const char * Channel, **PwrSnsrCondCode Enum** * CondCode, float * Val)

Returns the actual detected power of the proximal level in the current channel units.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
CondCode	Condition code for the measurement.
Val	Detected power of the proximal level in the current channel units.

Returns

Success (0) or error code.

◆PwrSnsr_GetChanTraceCount()

EXPORT int PwrSnsr_GetChanTraceCount (**SessionID** Vi, const char * Channel, int * TraceCount)

Get the number of points in the CCDF trace plot.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TraceCount	The number of points in the CCDF trace plot.

Returns

Success (0) or error code.

◆PwrSnsr_GetSweepTime()

EXPORT int PwrSnsr_GetSweepTime (

SessionID	Vi,
const char *	Channel,
float *	SweepTime

)

Get sweep time for the trace in seconds.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SweepTime	Sweep time for the trace in seconds.

Returns

Success (0) or error code.

◆PwrSnsr_GetTimePerPoint()

EXPORT int PwrSnsr_GetTimePerPoint (

SessionID	Vi,
const char *	Channel,
float *	TimePerPoint

)

Get time spacing for each waveform point in seconds.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TimePerPoint	Time spacing for each waveform point in seconds.

Returns

Success (0) or error code.

◆ PwrSnsr_GetTraceStartTime()

EXPORT int PwrSnsr_GetTraceStartTime (**SessionID** Vi, const char * Channel, float * TraceStartTime)

Get time offset (start time) of the trace in seconds. May be negative, indicating pre-trigger information.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
TraceStartTime	Time offset (start time) of the trace in seconds. May be negative, indicating pre-trigger information.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.14 Multiple Pulse

Power Sensor Library 1.1.0

Data Structures | Typedefs | Functions

© 2019 WTG

Multiple Pulse

Data Structures

struct	**PulseInfo**
	Data structure containing pulse information. More...

Typedefs

typedef struct **PulseInfo**	**PulseInfo**
	Data structure containing pulse information. More...

Functions

EXPORT int	**PwrSnsr_FetchAllMultiPulse** (**SessionID** Vi, const char *Channel, int PulseInfosSize, **PulseInfo** PulseInfos[], int *PulseInfosActualSize)
	Return all previously acquired multiple pulse measurements. The elements in the PulseInfos array correspond to pulses on the current trace from left to right (ascending time order). More...

Detailed Description

MultiplePulse functions

Typedef Documentation

◆ PulseInfo

© 2019 WTG

typedef struct **PulseInfo PulseInfo**

Data structure containing pulse information.

Function Documentation

◆PwrSnsr_FetchAllMultiPulse()

EXPORT int PwrSnsr_FetchAllMultiPulse	(	**SessionID**	Vi,
		const char *	Channel,
		int	PulseInfosSize,
		PulseInfo	PulseInfos[],
		int *	PulseInfosActualSize
	)		

Return all previously acquired multiple pulse measurements. The elements in the PulseInfos array correspond to pulses on the current trace from left to right (ascending time order).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PulseInfosSize	Number of elements in PulseInfos array.
PulseInfos	Array to fill with multi pulse information.
PulseInfosActualSize	Actual number of valid elements in PulseInfos array.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.14.1 PulseInfo

Power Sensor Library 1.1.0

Data Fields

© 2019 WTG

PulseInfo Struct Reference

Multiple Pulse

Data structure containing pulse information. More...

Data Fields

float	**Width**
float	**Peak**
float	**Min**
float	**PulseAvg**
float	**Position**
float	**RiseProximal**
float	**RiseDistal**
float	**RiseTime**
float	**FallProximal**
float	**FallDistal**
float	**FallTime**

Detailed Description

Data structure containing pulse information.

© 2019 WTG

Field Documentation

◆FallDistal

float FallDistal

Position in time for the distal crossing on the falling edge of the pulse.

◆FallProximal

float FallProximal

Position in time for the proximal crossing on the falling edge of the pulse.

◆FallTime

float FallTime

Fall time of the pulse.

◆Min

float Min

Minimum instanaeous power measurement.

◆Peak

float Peak

Peak (max instanaeous) power measurement.

◆Position

float Position

Time position corresponding to the mesial crossing of the rising edge for the pulse.

◆PulseAvg

float PulseAvg

Average power measurement for the pulse.

© 2019 WTG

◆ RiseDistal

float RiseDistal

Position in time for the distal crossing on the rising edge of the pulse.

◆ RiseProximal

float RiseProximal

Position in time for the proximal crossing on the rising edge of the pulse.

◆ RiseTime

float RiseTime

Rise time of the pulse.

◆ Width

float Width

Pulse width is defined as the interval between the first and second signal crossings of the mesial line.

The documentation for this struct was generated from the following file:

- **PwrSnsrLib.h**

Generated by doxygen 1.8.15

1.14.2 PulseInfo

Power Sensor Library 1.1.0

Data Structures | Typedefs | Functions

Multiple Pulse

Data Structures	
struct	**PulseInfo**

© 2019 WTG

	Data structure containing pulse information. More...

Typedefs

typedef struct **PulseInfo**	**PulseInfo**
	Data structure containing pulse information. More...

Functions

EXPORT int	**PwrSnsr_FetchAllMultiPulse** (**SessionID** Vi, const char *Channel, int PulseInfosSize, **PulseInfo** PulseInfos[], int *PulseInfosActualSize)
	Return all previously acquired multiple pulse measurements. The elements in the PulseInfos array correspond to pulses on the current trace from left to right (ascending time order). More...

Detailed Description

MultiplePulse functions

Typedef Documentation

◆ PulseInfo

typedef struct **PulseInfo PulseInfo**

Data structure containing pulse information.

Function Documentation

© 2019 WTG

◆ PwrSnsr_FetchAllMultiPulse()

```
EXPORT int PwrSnsr_FetchAllMultiPulse (
                                        SessionID       Vi,
                                        const char *    Channel,
                                        int             PulseInfosSize,
                                        PulseInfo       PulseInfos[],
                                        int *           PulseInfosActualSize
                                      )
```

Return all previously acquired multiple pulse measurements. The elements in the PulseInfos array correspond to pulses on the current trace from left to right (ascending time order).

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PulseInfosSize	Number of elements in PulseInfos array.
PulseInfos	Array to fill with multi pulse information.
PulseInfosActualSize	Actual number of valid elements in PulseInfos array.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.15 Memory Channels

Power Sensor Library 1.1.0

Functions

Memory Channels

Functions

© 2019 WTG

EXPORT int	**PwrSnsr_SaveToMemoryChannel** (**SessionID** Vi, const char *MemChan, const char *ChannelName)
	Saves the given channel to a memory channel. If the memory channel does not exist, a new one is created. More...
EXPORT int	**PwrSnsr_GetMemChanArchive** (**SessionID** Vi, const char *MemChan, int ValBufferSize, char Val[])
	Returns an XML document containing settings and readings obtained using the SaveToMemoryChannel method. More...
EXPORT int	**PwrSnsr_LoadMemChanFromArchive** (**SessionID** Vi, const char *MemChan, const char *ArchiveContent)
	Loads the named memory channel using the given archive. If the memory channel does not exist, one is created. More...

Detailed Description

Memory Channel functions

Function Documentation

◆ PwrSnsr_GetMemChanArchive()

EXPORT int PwrSnsr_GetMemChanArchive (

SessionID	Vi,
const char *	MemChan,
int	ValBufferSize,
char	Val[]

)

Returns an XML document containing settings and readings obtained using the SaveToMemoryChannel method.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MemChan	The name of the memory channel to get the archive from.
ValBufferSize	
Val	XML document containing settings and readings.

Returns

Success (0) or error code.

◆PwrSnsr_LoadMemChanFromArchive()

EXPORT int PwrSnsr_LoadMemChanFromArchive (

SessionID	Vi,
const char *	MemChan,
const char *	ArchiveContent

)

Loads the named memory channel using the given archive. If the memory channel does not exist, one is created.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MemChan	Memory channel name. Must have the form MEM1...n, where n is the number of measurement channels. In single channel configurations, this parameter should always be "MEM1".
ArchiveContent	An xml document containing settings and readings obtained using the SaveToMemoryChannel method. An archive can be obtained using the GetMemChanArchive method.

Returns

Success (0) or error code.

◆PwrSnsr_SaveToMemoryChannel()

© 2019 WTG

EXPORT int PwrSnsr_SaveToMemoryChannel	(	**SessionID**	Vi,
		const char *	MemChan,
		const char *	ChannelName
	)		

Saves the given channel to a memory channel. If the memory channel does not exist, a new one is created.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MemChan	Memory channel name. Must have the form MEM1...n, where n is the number of measurement channels. In single channel configurations, this parameter should always be "MEM1".
ChannelName	The channel name to copy from.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.16 Modulated Measurements

Power Sensor Library 1.1.0

Functions

Modulated Measurements

Functions

EXPORT int	**PwrSnsr_GetIsAvailable** (**SessionID** Vi, const char *Channel, int *IsAvailable)
	Returns true if modulated/CW measurement system is available. Will always return false if measurement buffer is enabled. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetIsRunning** (**SessionID** Vi, const char *Channel, int *IsRunning)
	Returns true if modulated/CW measuremnts are actively running. More...
EXPORT int	**PwrSnsr_GetReadingPeriod** (**SessionID** Vi, const char *Channel, float *ReadingPeriod)
	Returns the period (rate) in seconds per new filtered reading. More...

Detailed Description

Modulated measurement functions

Function Documentation

◆ PwrSnsr_GetIsAvailable()

EXPORT int PwrSnsr_GetIsAvailable	(	**SessionID**	Vi,
		const char *	Channel,
		int *	IsAvailable
	)		

Returns true if modulated/CW measurement system is available. Will always return false if measurement buffer is enabled.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsAvailable	True if modulated/CW measurement system is available. Will always return false if measurement buffer is enabled.

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_GetIsRunning()

EXPORT int PwrSnsr_GetIsRunning (**SessionID** Vi, const char * Channel, int * IsRunning)

Returns true if modulated/CW measuremnts are actively running.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
IsRunning	True if modulated/CW measuremnts are actively running.

Returns

Success (0) or error code.

◆ PwrSnsr_GetReadingPeriod()

EXPORT int PwrSnsr_GetReadingPeriod (**SessionID** Vi, const char * Channel, float * ReadingPeriod)

Returns the period (rate) in seconds per new filtered reading.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ReadingPeriod	The period (rate) in seconds per new filtered reading.

© 2019 WTG

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.17 Measurement Buffer

Power Sensor Library 1.1.0

Functions

Measurement Buffer

Functions	
EXPORT int	**PwrSnsr_GetBufferedAverageMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Get the average power measurements that were captured during the last call to AcquireMeasurements. More...
EXPORT int	**PwrSnsr_AcquireMeasurements** (**SessionID** Vi, double Timeout, int Count, **PwrSnsrMeasBuffStopReasonEnum** *StopReason, int *Val)
	Initiates new acquisition from the measurement buffer system (if acquisition is in the stopped state). Blocks until the number of measurements for each enabled channel is equal to count, or a time out has occurred. More...
EXPORT int	**PwrSnsr_GetMaxMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Get the maximum power measurements that were captured during the last call to AcquireMeasurements. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetMinMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Get the minimum power measurements that were captured during the last call to AcquireMeasurements. More...
EXPORT int	**PwrSnsr_GetDuration** (**SessionID** Vi, float *Duration)
	Get the time duration samples are captured during each timed mode acquisiton. More...
EXPORT int	**PwrSnsr_SetDuration** (**SessionID** Vi, float Duration)
	Set the duration samples are captured during each timed mode acquisition. More...
EXPORT int	**PwrSnsr_GetSequenceNumbers** (**SessionID** Vi, const char *Channel, int ValBufferSize, unsigned int Val[], int *ValActualSize)
	Get the sequence number entries that were captured during the last call to AcquireMeasurements. More...
EXPORT int	**PwrSnsr_GetStartTimes** (**SessionID** Vi, const char *Channel, int ValBufferSize, double Val[], int *ValActualSize)
	Get the start time entries in seconds that were captured during the last call to AcquireMeasurements. More...
EXPORT int	**PwrSnsr_GetDurations** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Get the duration entries in seconds that were captured during the last call to AcquireMeasurements. More...
EXPORT int	**PwrSnsr_StartAcquisition** (**SessionID** Vi)
	Starts measurement buffer acquisition. This method allows the user to send a command to the power meter to begin buffering

© 2019 WTG

	measurements without waiting for all measurements to be completed. Alternately, you can call the AcquireReadings method to start buffering measurements and wait for them to be read from the meter. More...
EXPORT int	**PwrSnsr_StopAcquisition** (**SessionID** Vi)
	Sends a command to stop the measurement buffer from acquiring readings. More...
EXPORT int	**PwrSnsr_ClearBuffer** (**SessionID** Vi)
	Sends a command to the power meter to clear all buffered readings. This method does not clear cached measurements accessible through GetAverageMeasurements, etc. More...
EXPORT int	**PwrSnsr_ClearMeasurements** (**SessionID** Vi)
	Clears cached average, min, max, duration, start time, and sequence number measurements. More...
EXPORT int	**PwrSnsr_GetMeasurementsAvailable** (**SessionID** Vi, const char *Channel, int *Val)
	Get the number of measurement entries available that were captured during AcquireMeasurements(). More...
EXPORT int	**PwrSnsr_SetPeriod** (**SessionID** Vi, float Period)
	Set the period each timed mode acquisition (measurement buffer) is started. More...
EXPORT int	**PwrSnsr_GetPeriod** (**SessionID** Vi, float *Period)
	Get the period each timed mode acquisition (measurement buffer) is started. More...
EXPORT int	**PwrSnsr_GetRdgsEnableFlag** (**SessionID** Vi, int *Flag)

© 2019 WTG

	Get the flag indicating which measurement buffer arrays will be read when calling PwrSnsr_AcquireMeasurements. More...
EXPORT int	**PwrSnsr_SetRdgsEnableFlag** (**SessionID** Vi, int Flag)
	Set the flag indicating which measurement buffer arrays will be read when calling PwrSnsr_AcquireMeasurements. More...
EXPORT int	**PwrSnsr_GetGateMode** (**SessionID** Vi, **PwrSnsrMeasBuffGateEnum** *GateMode)
	Each Measurement Buffer Entry is controlled by a buffer gate that defines the start and end of the entry time interval. The gate signal may be internally or externally generated in several different ways. More...
EXPORT int	**PwrSnsr_SetGateMode** (**SessionID** Vi, **PwrSnsrMeasBuffGateEnum** GateMode)
	Each Measurement Buffer Entry is controlled by a buffer gate that defines the start and end of the entry time interval. More...
EXPORT int	**PwrSnsr_GetStartMode** (**SessionID** Vi, **PwrSnsrMeasBuffStartModeEnum** *StartMode)
	Get the mode used to start acquisition of buffer entries. More...
EXPORT int	**PwrSnsr_SetStartMode** (**SessionID** Vi, **PwrSnsrMeasBuffStartModeEnum** StartMode)
	Set the mode used to start acquisition of buffer entries. More...
EXPORT int	**PwrSnsr_AdvanceReadIndex** (**SessionID** Vi)
	Send a command to the meter to notify it the user is done reading and to advance the read index. More...

© 2019 WTG

EXPORT int	**PwrSnsr_QueryAverageMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Query the power meter for all buffered average power measurements. More...
EXPORT int	**PwrSnsr_QueryStartTimes** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Query the power meter for all buffered start times in seconds. More...
EXPORT int	**PwrSnsr_QuerySequenceNumbers** (**SessionID** Vi, const char *Channel, int ValBufferSize, unsigned int Val[], int *ValActualSize)
	Query the power meter for all buffered sequence numbers. More...
EXPORT int	**PwrSnsr_QueryDurations** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Query the power meter for all buffered measurement durations in seconds. More...
EXPORT int	**PwrSnsr_QueryMaxMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Query the power meter for all buffered maximum power measurements. More...
EXPORT int	**PwrSnsr_QueryMinMeasurements** (**SessionID** Vi, const char *Channel, int ValBufferSize, float Val[], int *ValActualSize)
	Query the power meter for all buffered minimum power measurements. More...
EXPORT int	**PwrSnsr_GetWriteProtection** (**SessionID** Vi, int *WriteProtection)
	Get whether the measurement buffer is set to overwrite members that have not been read by the user. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetTimedOut** (**SessionID** Vi, int *TimedOut)
	Check if the last measurement buffer session timed out. More...
EXPORT int	**PwrSnsr_GetSessionCount** (**SessionID** Vi, int *SessionCount)
	Get the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements. More...
EXPORT int	**PwrSnsr_SetSessionCount** (**SessionID** Vi, int SessionCount)
	Set the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements. More...
EXPORT int	**PwrSnsr_SetSessionTimeout** (**SessionID** Vi, float Seconds)
	Set the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements. More...
EXPORT int	**PwrSnsr_GetReturnCount** (**SessionID** Vi, int *ReturnCount)
	Get the return count for each measurement query. More...
EXPORT int	**PwrSnsr_SetReturnCount** (**SessionID** Vi, int ReturnCount)
	Set the return count for each measurement query. More...
EXPORT int	**PwrSnsr_SetWriteProtection** (**SessionID** Vi, int WriteProtection)
	Set whether to allow the measurement buffer to overwrite entries that have not been read by the user. More...

© 2019 WTG

EXPORT int	**PwrSnsr_GetOverRan** (**SessionID** Vi, int *OverRan)
	Get flag indicating whether the power meter's internal buffer filled up before being emptied. More...
EXPORT int	**PwrSnsr_GetBufferedMeasurementsAvailable** (**SessionID** Vi, int *MeasurementsAvailable)
	Gets the number of measurements available in the power meter's internal buffer. Note: The number of readings that have been acquired may be more or less. More...
EXPORT int	**PwrSnsr_GetMeasBuffEnabled** (**SessionID** Vi, int *MeasBuffEnabled)
	Get whether the measurement buffer has been enabled. More...
EXPORT int	**PwrSnsr_SetMeasBuffEnabled** (**SessionID** Vi, int MeasBuffEnabled)
	Enable or disable the measurement buffer. Disabling the measurement buffer enables modulated/CW measurements. Conversely, enabling it disables modulated/CW measurements. More...
EXPORT int	**PwrSnsr_ResetContinuousCapture** (**SessionID** Vi)
	Sets a flag indicating to restart continuous capture. This method allows the user to restart continuous acquisition. Has no effect if ContinuousCapture is set to false. More...
EXPORT int	**PwrSnsr_GetEndDelay** (**SessionID** Vi, float *EndDelay)
	Get delay time added to the detected end of a burst for analysis. Typically negative. Typically used to exclude the falling edge of a burst. More...

© 2019 WTG

EXPORT int	**PwrSnsr_SetEndDelay** (**SessionID** Vi, float EndDelay)
	Set delay time added to the detected end of a burst for analysis. Typically negative. Typically used to exclude the falling edge of a burst. More...
EXPORT int	**PwrSnsr_GetStartQual** (**SessionID** Vi, float *StartQual)
	Get the minimum amount of time power remains above the trigger point to be counted as the beginning of a burst. More...
EXPORT int	**PwrSnsr_SetStartQual** (**SessionID** Vi, float StartQual)
	Set the minimum amount of time power remains above the trigger point to be counted as the beginning of a burst. More...
EXPORT int	**PwrSnsr_GetStartDelay** (**SessionID** Vi, float *StartDelay)
	Get delay time added to the detected beginning of a burst for analysis. Typically used to exclude the rising edge of a burst. More...
EXPORT int	**PwrSnsr_SetStartDelay** (**SessionID** Vi, float StartDelay)
	Set delay time added to the detected beginning of a burst for analysis. Typically used to exclude the rising edge of a burst. More...
EXPORT int	**PwrSnsr_GetEndQual** (**SessionID** Vi, float *EndQual)
	Get the minimum amount of time power remains below the trigger point to be counted as the end of a burst. More...
EXPORT int	**PwrSnsr_SetEndQual** (**SessionID** Vi, float EndQual)
	Set the minimum amount of time power remains below the trigger point to be counted

© 2019 WTG

	as the end of a burst. More...
EXPORT int	**PwrSnsr_SetContinuousCapture** (**SessionID** Vi, int ContinuousCapture)
	Set whether AcquireMeasurements will stop the measurement buffer session or continue capturing measurement buffer entries after being called. More...
EXPORT int	**PwrSnsr_GetContinuousCapture** (**SessionID** Vi, int *ContinuousCapture)
	Get whether AcquireMeasurements will stop the measurement buffer session or continue capturing measurement buffer entries after being called. More...

Detailed Description

Measurement Buffer functions

Function Documentation

◆ PwrSnsr_AcquireMeasurements()

EXPORT int PwrSnsr_AcquireMeasurements (

SessionID	Vi,
double	Timeout,
int	Count,
PwrSnsrMeasBuffStopReasonEnum *	StopReason,
int *	Val

)

Initiates new acquisition from the measurement buffer system (if acquisition is in the stopped state). Blocks until the number of measurements for each enabled channel is equal to count, or a time out has occurred.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Timeout	Maximum time in seconds to continue acquiring samples. Negative values will be treated as infinite.
Count	Number of samples to acquire.
StopReason	Reason aqcuisition stopped.
Val	Number of samples acquired.

Returns

Success (0) or error code.

◆PwrSnsr_AdvanceReadIndex()

EXPORT int PwrSnsr_AdvanceReadIndex (**SessionID** Vi)

Send a command to the meter to notify it the user is done reading and to advance the read index.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_ClearBuffer()

EXPORT int PwrSnsr_ClearBuffer (**SessionID** Vi)

Sends a command to the power meter to clear all buffered readings. This method does not clear cached measurements accessible through GetAverageMeasurements, etc.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

© 2019 WTG

Success (0) or error code.

◆PwrSnsr_ClearMeasurements()

EXPORT int PwrSnsr_ClearMeasurements (**SessionID** Vi)

Clears cached average, min, max, duration, start time, and sequence number measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_GetBufferedAverageMeasurements()

EXPORT int PwrSnsr_GetBufferedAverageMeasurements (

SessionID	Vi,
const char *	Channel,
int	ValBufferSize,
float	Val[],
int *	ValActualSize

)

Get the average power measurements that were captured during the last call to AcquireMeasurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Buffer size of Val.
Val	Array of average measurements.
ValActualSize	Actual size of Val.

© 2019 WTG

Returns

Success (0) or error code.

◆PwrSnsr_GetBufferedMeasurementsAvailable()

EXPORT int PwrSnsr_GetBufferedMeasurementsAvailable (**SessionID** Vi, int * MeasurementsAvailable)

Gets the number of measurements available in the power meter's internal buffer. Note: The number of readings that have been acquired may be more or less.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MeasurementsAvailable	The number of measurements available in the power meter's internal buffer. Note: The number of readings that have been acquired may be more or less.

Returns

Success (0) or error code.

◆PwrSnsr_GetContinuousCapture()

EXPORT int PwrSnsr_GetContinuousCapture (**SessionID** Vi, int * ContinuousCapture)

Get whether AcquireMeasurements will stop the measurement buffer session or continue capturing measurement buffer entries after being called.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ContinuousCapture	True if AcquireMeasurements will stop the measurement buffer session or continue

© 2019 WTG

capturing measurement buffer entries after being called.

Returns

Success (0) or error code.

◆PwrSnsr_GetDuration()

```
EXPORT int
PwrSnsr_GetDuration (
                        SessionID       Vi,
                        float *         Duration
                    )
```

Get the time duration samples are captured during each timed mode acquisiton.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Duration	The duration in seconds samples are captured during each timed mode acquisiton.

Returns

Success (0) or error code.

◆PwrSnsr_GetDurations()

```
EXPORT int
PwrSnsr_GetDurations (
                        SessionID       Vi,
                        const char *    Channel,
                        int             ValBufferSize,
                        float           Val[],
                        int *           ValActualSize
                      )
```

Get the duration entries in seconds that were captured during the last call to AcquireMeasurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".

© 2019 WTG

ValBufferSize	Size of the buffer.
Val	Array of measurement durations in seconds.
ValActualSize	Actual size of the returned buffer.

Returns

Success (0) or error code.

◆PwrSnsr_GetEndDelay()

EXPORT int PwrSnsr_GetEndDelay (**SessionID** Vi, float * EndDelay)

Get delay time added to the detected end of a burst for analysis. Typically negative. Typically used to exclude the falling edge of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
EndDelay	The delay time added to the detected end of a burst for analysis.

Returns

Success (0) or error code.

◆PwrSnsr_GetEndQual()

EXPORT int PwrSnsr_GetEndQual (**SessionID** Vi, float * EndQual)

Get the minimum amount of time power remains below the trigger point to be counted as the end of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
EndQual	The minimum amount of time power remains below the trigger point to be counted as the

© 2019 WTG

end of a burst.

Returns

Success (0) or error code.

◆PwrSnsr_GetGateMode()

EXPORT int PwrSnsr_GetGateMode	(	**SessionID**	Vi,
		PwrSnsrMeasBuffGateEnum *	GateMode
	)		

Each Measurement Buffer Entry is controlled by a buffer gate that defines the start and end of the entry time interval. The gate signal may be internally or externally generated in several different ways.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
GateMode	Buffer gate mode that defines the start and end of the entry time interval.

Returns

Success (0) or error code.

◆PwrSnsr_GetMaxMeasurements()

EXPORT int PwrSnsr_GetMaxMeasurements	(	**SessionID**	Vi,
		const char *	Channel,
		int	ValBufferSize,
		float	Val[],
		int *	ValActualSize
	)		

Get the maximum power measurements that were captured during the last call to AcquireMeasurements.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of max measurements.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆ PwrSnsr_GetMeasBuffEnabled()

EXPORT int PwrSnsr_GetMeasBuffEnabled (**SessionID** Vi, int * MeasBuffEnabled)

Get whether the measurement buffer has been enabled.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MeasBuffEnabled	True if measurement buffer is enabled.

Returns

Success (0) or error code.

◆ PwrSnsr_GetMeasurementsAvailable()

EXPORT int PwrSnsr_GetMeasurementsAvailable (**SessionID** Vi, const char * Channel, int * Val)

Get the number of measurement entries available that were captured during AcquireMeasurements().

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Val	Number of measurement entries available.

Returns

Success (0) or error code.

◆PwrSnsr_GetMinMeasurements()

EXPORT int PwrSnsr_GetMinMeasurements (

SessionID	Vi,
const char *	Channel,
int	ValBufferSize,
float	Val[],
int *	ValActualSize

)

Get the minimum power measurements that were captured during the last call to AcquireMeasurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of min measurements.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆PwrSnsr_GetOverRan()

EXPORT int PwrSnsr_GetOverRan (

SessionID	Vi,
int *	OverRan

© 2019 WTG

)

Get flag indicating whether the power meter's internal buffer filled up before being emptied.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
OverRan	True if the power meter's internal buffer filled up before being emptied.

Returns

Success (0) or error code.

◆PwrSnsr_GetPeriod()

EXPORT int PwrSnsr_GetPeriod (**SessionID** Vi, float * Period)

Get the period each timed mode acquisition (measurement buffer) is started.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Period	The period in seconds each timed mode acquisition is started.

Returns

Success (0) or error code.

◆PwrSnsr_GetRdgsEnableFlag()

EXPORT int PwrSnsr_GetRdgsEnableFlag (**SessionID** Vi, int * Flag)

Get the flag indicating which measurement buffer arrays will be read when calling PwrSnsr_AcquireMeasurements.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Flag	Bit masked value indicating which measurement arrays will be queried (see PwrSnsrRdgsEnableFlag).

Returns

Success (0) or error code.

◆ PwrSnsr_GetReturnCount()

EXPORT int PwrSnsr_GetReturnCount (**SessionID** Vi, int * ReturnCount)

Get the return count for each measurement query.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ReturnCount	The return count for each measurement query.

Returns

Success (0) or error code.

◆ PwrSnsr_GetSequenceNumbers()

EXPORT int PwrSnsr_GetSequenceNumbers (**SessionID** Vi, const char * Channel, int ValBufferSize, unsigned int Val[], int * ValActualSize)

Get the sequence number entries that were captured during the last call to AcquireMeasurements.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of sequence numbers.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆ PwrSnsr_GetSessionCount()

EXPORT int PwrSnsr_GetSessionCount (**SessionID** Vi, int * SessionCount)

Get the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
SessionCount	Get the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements.

Returns

Success (0) or error code.

◆ PwrSnsr_GetStartDelay()

EXPORT int PwrSnsr_GetStartDelay (**SessionID** Vi, float * StartDelay)

© 2019 WTG

Get delay time added to the detected beginning of a burst for analysis. Typically used to exclude the rising edge of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartDelay	Delay time in seconds added to the detected beginning of a burst for analysis.

Returns

Success (0) or error code.

◆PwrSnsr_GetStartMode()

EXPORT int PwrSnsr_GetStartMode (**SessionID** Vi, **PwrSnsrMeasBuffStartModeEnum** * StartMode)

Get the mode used to start acquisition of buffer entries.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartMode	Mode used to start acquisition of buffer entries.

Returns

Success (0) or error code.

◆PwrSnsr_GetStartQual()

EXPORT int PwrSnsr_GetStartQual (**SessionID** Vi, float * StartQual)

Get the minimum amount of time power remains above the trigger point to be counted as the beginning of a burst.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartQual	The minimum amount of time power remains above the trigger point to be counted as the beginning of a burst.

Returns

Success (0) or error code.

◆ PwrSnsr_GetStartTimes()

EXPORT int PwrSnsr_GetStartTimes (

SessionID	Vi,
const char *	Channel,
int	ValBufferSize,
double	Val[],
int *	ValActualSize

)

Get the start time entries in seconds that were captured during the last call to AcquireMeasurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of start times.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆ PwrSnsr_GetTimedOut()

EXPORT int PwrSnsr_GetTimedOut (

SessionID	Vi,

© 2019 WTG

int * TimedOut

)

Check if the last measurement buffer session timed out.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
TimedOut	True if the last measurement buffer session timed out.

Returns

Success (0) or error code.

◆PwrSnsr_GetWriteProtection()

EXPORT int PwrSnsr_GetWriteProtection (**SessionID** Vi,
int * WriteProtection
)

Get whether the measurement buffer is set to overwrite members that have not been read by the user.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
WriteProtection	Returns trueif the measurement buffer is allowed to overwrite members that have not been read by the user.

Returns

Success (0) or error code.

◆PwrSnsr_QueryAverageMeasurements()

EXPORT int PwrSnsr_QueryAverageMeasurements (**SessionID** Vi,
const char * Channel,
int ValBufferSize,
float Val[],

© 2019 WTG

int * ValActualSize

)

Query the power meter for all buffered average power measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer in elements.
Val	Array of average power measurements.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆ PwrSnsr_QueryDurations()

EXPORT int PwrSnsr_QueryDurations (**SessionID** Vi,
const char * Channel,
int ValBufferSize,
float Val[],
int * ValActualSize
)

Query the power meter for all buffered measurement durations in seconds.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of buffered measurement durations.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_QueryMaxMeasurements()

```
EXPORT int PwrSnsr_QueryMaxMeasurements (
                                          SessionID Vi,
                                          const char * Channel,
                                          int ValBufferSize,
                                          float Val[],
                                          int * ValActualSize
                                        )
```

Query the power meter for all buffered maximum power measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of max measurements.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆PwrSnsr_QueryMinMeasurements()

```
EXPORT int PwrSnsr_QueryMinMeasurements (
                                          SessionID Vi,
                                          const char * Channel,
                                          int ValBufferSize,
                                          float Val[],
                                          int * ValActualSize
                                        )
```

Query the power meter for all buffered minimum power measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of min measurements.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆PwrSnsr_QuerySequenceNumbers()

EXPORT int PwrSnsr_QuerySequenceNumbers (

SessionID	Vi,
const char *	Channel,
int	ValBufferSize,
unsigned int	Val[],
int *	ValActualSize

)

Query the power meter for all buffered sequence numbers.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of sequence numbers.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆PwrSnsr_QueryStartTimes()

EXPORT int PwrSnsr_QueryStartTimes (

SessionID	Vi,
const char *	Channel,
int	ValBufferSize,
float	Val[],

© 2019 WTG

int * ValActualSize

)

Query the power meter for all buffered start times in seconds.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
ValBufferSize	Size of the buffer.
Val	Array of start times in seconds.
ValActualSize	Actual size of the returned array in elements.

Returns

Success (0) or error code.

◆ PwrSnsr_ResetContinuousCapture()

EXPORT int PwrSnsr_ResetContinuousCapture (**SessionID** Vi)

Sets a flag indicating to restart continuous capture. This method allows the user to restart continuous acquisition. Has no effect if ContinuousCapture is set to false.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆ PwrSnsr_SetContinuousCapture()

EXPORT int PwrSnsr_SetContinuousCapture (**SessionID** Vi, int ContinuousCapture)

© 2019 WTG

Set whether AcquireMeasurements will stop the measurement buffer session or continue capturing measurement buffer entries after being called.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ContinuousCapture	True to set whether AcquireMeasurements will stop the measurement buffer session or continue capturing measurement buffer entries after being called.

Returns

Success (0) or error code.

◆PwrSnsr_SetDuration()

EXPORT int PwrSnsr_SetDuration (**SessionID** Vi, float Duration)

Set the duration samples are captured during each timed mode acquisition.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Duration	The duration samples are captured during each timed mode acquisition in seconds.

Returns

Success (0) or error code.

◆PwrSnsr_SetEndDelay()

EXPORT int PwrSnsr_SetEndDelay (**SessionID** Vi, float EndDelay)

Set delay time added to the detected end of a burst for analysis. Typically negative. Typically used to exclude the falling edge of a burst.

© 2019 WTG

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
EndDelay	Delay time added to the detected end of a burst for analysis.

Returns

Success (0) or error code.

◆PwrSnsr_SetEndQual()

EXPORT int PwrSnsr_SetEndQual (**SessionID** Vi, float EndQual)

Set the minimum amount of time power remains below the trigger point to be counted as the end of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
EndQual	The minimum amount of time power remains below the trigger point to be counted as the end of a burst.

Returns

Success (0) or error code.

◆PwrSnsr_SetGateMode()

EXPORT int PwrSnsr_SetGateMode (**SessionID** Vi, **PwrSnsrMeasBuffGateEnum** GateMode)

Each Measurement Buffer Entry is controlled by a buffer gate that defines the start and end of the entry time interval.

Parameters

© 2019 WTG

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
GateMode	Buffer gate mode that defines the start and end of the entry time interval.

Returns

Success (0) or error code.

◆ PwrSnsr_SetMeasBuffEnabled()

EXPORT int PwrSnsr_SetMeasBuffEnabled (**SessionID** Vi, int MeasBuffEnabled)

Enable or disable the measurement buffer. Disabling the measurement buffer enables modulated/CW measurements. Conversely, enabling it disables modulated/CW measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
MeasBuffEnabled	True to enable measurement buffer, false to disable.

Returns

Success (0) or error code.

◆ PwrSnsr_SetPeriod()

EXPORT int PwrSnsr_SetPeriod (**SessionID** Vi, float Period)

Set the period each timed mode acquisition (measurement buffer) is started.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

© 2019 WTG

Period	The period in seconds each timed mode acquisition is started.

Returns

Success (0) or error code.

◆PwrSnsr_SetRdgsEnableFlag()

EXPORT int PwrSnsr_SetRdgsEnableFlag (**SessionID** int Vi, Flag)

Set the flag indicating which measurement buffer arrays will be read when calling PwrSnsr_AcquireMeasurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Flag	Bit masked value indicating which measurement arrays will be queried (see PwrSnsrRdgsEnableFlag).

Returns

Success (0) or error code.

◆PwrSnsr_SetReturnCount()

EXPORT int PwrSnsr_SetReturnCount (**SessionID** int Vi, ReturnCount)

Set the return count for each measurement query.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
ReturnCount	The return count for each measurement query.

Returns

© 2019 WTG

Success (0) or error code.

◆ PwrSnsr_SetSessionCount()

EXPORT int PwrSnsr_SetSessionCount (**SessionID** Vi, int SessionCount)

Set the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
SessionCount	Set the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements.

Returns

Success (0) or error code.

◆ PwrSnsr_SetSessionTimeout()

EXPORT int PwrSnsr_SetSessionTimeout (**SessionID** Vi, float Seconds)

Set the count of elements for this measurement buffer session. Set to 0 for the meter to continuously acquire measurements.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Seconds	Set the time out value. Values less than or equal to 0 will be treated as infinite. Valid range : 0.001 to 1000

Returns

© 2019 WTG

Success (0) or error code.

◆PwrSnsr_SetStartDelay()

EXPORT int PwrSnsr_SetStartDelay	(	**SessionID**	Vi,
		float	StartDelay
	)		

Set delay time added to the detected beginning of a burst for analysis. Typically used to exclude the rising edge of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartDelay	Delay time in seconds added to the detected beginning of a burst for analysis.

Returns

Success (0) or error code.

◆PwrSnsr_SetStartMode()

EXPORT int PwrSnsr_SetStartMode	(	**SessionID**	Vi,
		PwrSnsrMeasBuffStartModeEnum	StartMode
	)		

Set the mode used to start acquisition of buffer entries.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartMode	Mode used to start acquisition of buffer entries.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetStartQual()

EXPORT int PwrSnsr_SetStartQual (**SessionID** Vi, float StartQual)

Set the minimum amount of time power remains above the trigger point to be counted as the beginning of a burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
StartQual	The minimum amount of time power remains above the trigger point to be counted as the beginning of a burst.

Returns

Success (0) or error code.

◆PwrSnsr_SetWriteProtection()

EXPORT int PwrSnsr_SetWriteProtection (**SessionID** Vi, int WriteProtection)

Set whether to allow the measurement buffer to overwrite entries that have not been read by the user.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
WriteProtection	Set false to allow the measurement buffer to overwrite entries that have not been read by the user.

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_StartAcquisition()

EXPORT int PwrSnsr_StartAcquisition (**SessionID** Vi)

Starts measurement buffer acquisition. This method allows the user to send a command to the power meter to begin buffering measurements without waiting for all measurements to be completed. Alternately, you can call the AcquireReadings method to start buffering measurements and wait for them to be read from the meter.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

◆PwrSnsr_StopAcquisition()

EXPORT int PwrSnsr_StopAcquisition (**SessionID** Vi)

Sends a command to stop the measurement buffer from acquiring readings.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.18 Sensor RawIO

Power Sensor Library 1.1.0

Functions

© 2019 WTG

Sensor RawIO

Functions

EXPORT int	**PwrSnsr_Write** (**SessionID** Vi, const char *Channel, int DataBufferSize, unsigned char Data[])
	Write a byte array to the meter. More...
EXPORT int	**PwrSnsr_ReadByteArray** (**SessionID** Vi, const char *Channel, int Count, int ValBufferSize, unsigned char Val[], int *ValActualSize)
	Reads byte array from the meter. More...
EXPORT int	**PwrSnsr_ReadControl** (**SessionID** Vi, const char *Channel, int Count, int ValBufferSize, unsigned char Val[], int *ValActualSize)
	Reads a control transfer on the USB. More...

Detailed Description

Sensor RawIO functions

Function Documentation

◆ PwrSnsr_ReadByteArray()

EXPORT int PwrSnsr_ReadByteArray (

SessionID	Vi,
const char *	Channel,
int	Count,
int	ValBufferSize,

© 2019 WTG

	unsigned char	Val[],
	int *	ValActualSize

)

Reads byte array from the meter.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Count	Maximum count of bytes to return.
ValBufferSize	Size of the buffer.
Val	Byte array from the USB.
ValActualSize	Actual size of the returned array in bytes.

Returns

Success (0) or error code.

◆ PwrSnsr_ReadControl()

EXPORT int PwrSnsr_ReadControl (

SessionID	Vi,
const char *	Channel,
int	Count,
int	ValBufferSize,
unsigned char	Val[],
int *	ValActualSize

)

Reads a control transfer on the USB.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Count	Maximum count to return.
ValBufferSize	Size of the buffer.
Val	Byte array from a USB control transfer.
ValActualSize	Actual size of the returned array in bytes.

© 2019 WTG

Returns

Success (0) or error code.

◆ PwrSnsr_Write()

EXPORT int PwrSnsr_Write	(	**SessionID**	Vi,
		const char *	Channel,
		int	DataBufferSize,
		unsigned char	Data[]
	)		

Write a byte array to the meter.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_init function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
DataBufferSize	Size of the buffer in bytes.
Data	Data to send.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

1.19 License Functions

Power Sensor Library 1.1.0

Functions

License Functions

Functions

EXPORT int	**PwrSnsr_GetDongleSerialNumber** (long *val)

© 2019 WTG

	Get the hardware license serial number. More...
EXPORT int	**PwrSnsr_GetExpirationDate** (int *Date)
	Get the hardware license expiration date. More...
EXPORT int	**PwrSnsr_GetNumberOfCals** (long *val)
	Get the number of calibrations left on the license. More...
EXPORT int	**PwrSnsr_IsLicenseDongleConnected** (int *val)
	Get whether the hardware license dongle is connected. More...

Detailed Description

License functions (Windows Only)

Function Documentation

◆ PwrSnsr_GetDongleSerialNumber()

EXPORT int PwrSnsr_GetDongleSerialNumber (long * val)

Get the hardware license serial number.

Parameters

val	Serial number of the license dongle

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_GetExpirationDate()

EXPORT int PwrSnsr_GetExpirationDate (int * Date)

Get the hardware license expiration date.

Parameters

Date	expiration date in the format YYYYMMDD

Returns

Success (0) or error code.

◆PwrSnsr_GetNumberOfCals()

EXPORT int PwrSnsr_GetNumberOfCals (long * val)

Get the number of calibrations left on the license.

Parameters

val	Number of cals left.

Returns

Success (0) or error code.

◆PwrSnsr_IsLicenseDongleConnected()

EXPORT int PwrSnsr_IsLicenseDongleConnected (int * val)

Get whether the hardware license dongle is connected.

Parameters

val	Boolean. 1 for connected or 0 for not connected.

Returns

Success (0) or error code.

© 2019 WTG

Generated by doxygen 1.8.15

1.20 Sensor Simulation

Power Sensor Library 1.1.0

Typedefs | Enumerations | Functions

Sensor Simulation

Typedefs

typedef enum **PwrSnsrSignalUnits**	**PwrSnsrSignalUnits**
typedef enum **PwrSnsrSimSignalType**	**PwrSnsrSimSignalType**

Enumerations

enum	**PwrSnsrSignalUnits** { **PwrSnsrSignalUnitsdBm** = 0, **PwrSnsrSignalUnitsWatts** = 1 }
enum	**PwrSnsrSimSignalType** { **PwrSnsrSimSignalPeriodic** = 0, **PwrSnsrSimSignalBurst** = 1 }

Functions

EXPORT int	**PwrSnsr_OpenSimMeter** (int NumChans, char *ResourceBuff, int ResourceBuffSize, **SessionID** *Vi)
	Open a simulated power meter session. More...
EXPORT int	**PwrSnsr_SetSimSignalAmplitude** (**SessionID** Vi, const char *Channel, float

© 2019 WTG

	Amplitude, **PwrSnsrSignalUnits** Units)
	Set the amplitude for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_GetSimSignalAmplitude** (**SessionID** Vi, const char *Channel, float *Amplitude, **PwrSnsrSignalUnits** Units)
	Get the amplitude for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_SetSimSignalModulation** (**SessionID** Vi, const char *Channel, float Percent)
	Set the percent modulation for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_GetSimSignalModulation** (**SessionID** Vi, const char *Channel, float *Percent)
	Get the percent modulation for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_SetSimSignalCompression** (**SessionID** Vi, const char *Channel, float Percent)
	Set the percent compression for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_GetSimSignalCompression** (**SessionID** Vi, const char *Channel, float *Percent)
	Get the percent compression for the signal on a simulation channel. More...
EXPORT int	**PwrSnsr_SetSimSignalType** (**SessionID** Vi, const char *Channel, **PwrSnsrSimSignalType** SignalType)
	Set the simulated channel signal type. More...
EXPORT int	**PwrSnsr_GetSimSignalType** (**SessionID** Vi, const char *Channel, **PwrSnsrSimSignalType** *SignalType)

© 2019 WTG

	Get the simulated channel signal type. More...
EXPORT int	**PwrSnsr_SetSimSignalPRF** (**SessionID** Vi, const char *Channel, float PRF)
	Set the simulated signal PRF. Valid for Periodic and Burst. More...
EXPORT int	**PwrSnsr_GetSimSignalPRF** (**SessionID** Vi, const char *Channel, float *PRF)
	Get the simulated signal PRF. Valid for Periodic and Burst. More...
EXPORT int	**PwrSnsr_SetSimSignalDuty** (**SessionID** Vi, const char *Channel, float Duty)
	Set the simulated signal duty cycle in percent. Affects Periodic only. More...
EXPORT int	**PwrSnsr_GetSimSignalDuty** (**SessionID** Vi, const char *Channel, float *Duty)
	Get the simulated signal duty cycle in percent. Affects Periodic only. More...

Detailed Description

Sensor simulation functions

Typedef Documentation

◆ PwrSnsrSignalUnits

typedef enum **PwrSnsrSignalUnits PwrSnsrSignalUnits**

Unit selector for watts or dBm.

◆ PwrSnsrSimSignalType

typedef enum **PwrSnsrSimSignalType PwrSnsrSimSignalType**

© 2019 WTG

Simulated signal type.

Enumeration Type Documentation

◆ PwrSnsrSignalUnits

enum **PwrSnsrSignalUnits**

Unit selector for watts or dBm.

Enumerator	
PwrSnsrSignalUnitsdBm	dBm.
PwrSnsrSignalUnitsWatts	Watts.

◆ PwrSnsrSimSignalType

enum **PwrSnsrSimSignalType**

Simulated signal type.

Enumerator	
PwrSnsrSimSignalPeriodic	Periodic waveform defined by PRF and duty cycle.
PwrSnsrSimSignalBurst	IFF-like burst signal.

Function Documentation

◆ PwrSnsr_GetSimSignalAmplitude()

EXPORT int PwrSnsr_GetSimSignalAmplitude (

SessionID	Vi,
const char *	Channel,
float *	Amplitude,
PwrSnsrSignalUnits	Units

© 2019 WTG

)

Get the amplitude for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Amplitude	The simulated signal level (return value).
Units	Units of amplitude in dBm or Watts

Returns

Success (0) or error code.

◆PwrSnsr_GetSimSignalCompression()

EXPORT int PwrSnsr_GetSimSignalCompression (**SessionID** Vi, const char * Channel, float * Percent)

Get the percent compression for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Percent	The percent compression (return value).

Returns

Success (0) or error code.

◆PwrSnsr_GetSimSignalDuty()

EXPORT int PwrSnsr_GetSimSignalDuty (**SessionID** Vi,

© 2019 WTG

		const char *	Channel,
		float *	Duty
	)		

Get the simulated signal duty cycle in percent. Affects Periodic only.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Duty	Duty cycle in percent (return value).

Returns

Success (0) or error code.

◆PwrSnsr_GetSimSignalModulation()

EXPORT int PwrSnsr_GetSimSignalModulation	(	**SessionID**	Vi,
		const char *	Channel,
		float *	Percent
	)		

Get the percent modulation for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Percent	The percent modulation (return value).

Returns

Success (0) or error code.

◆PwrSnsr_GetSimSignalPRF()

EXPORT int PwrSnsr_GetSimSignalPRF	(	**SessionID**	Vi,

© 2019 WTG

	const char *	Channel,
	float *	PRF

)

Get the simulated signal PRF. Valid for Periodic and Burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PRF	Pulse repetition frequency in Hz (return value).

Returns

Success (0) or error code.

◆ PwrSnsr_GetSimSignalType()

EXPORT int PwrSnsr_GetSimSignalType (

SessionID		Vi,
const char *		Channel,
PwrSnsrSimSignalType *		SignalType

)

Get the simulated channel signal type.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SignalType	The simulated signal type (return value).

Returns

Success (0) or error code.

◆ PwrSnsr_OpenSimMeter()

© 2019 WTG

EXPORT int PwrSnsr_OpenSimMeter (int NumChans, char * ResourceBuff, int ResourceBuffSize, **SessionID** * Vi)

Open a simulated power meter session.

Parameters

NumChans	Number of channels
ResourceBuff	Buffer to read back the session name, can be NULL
ResourceBuffSize	size of ResourceBuff in characters
Vi	SessionID handle (out parameter)

Returns

Success (0) or error code.

◆ PwrSnsr_SetSimSignalAmplitude()

EXPORT int PwrSnsr_SetSimSignalAmplitude (**SessionID** Vi, const char * Channel, float Amplitude, **PwrSnsrSignalUnits** Units)

Set the amplitude for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Amplitude	The simulated signal level
Units	Units of amplitude in dBm or Watts

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetSimSignalCompression()

```
EXPORT int PwrSnsr_SetSimSignalCompression (
    SessionID Vi,
    const char * Channel,
    float Percent
)
```

Set the percent compression for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Percent	The percent compression.

Returns

Success (0) or error code.

◆PwrSnsr_SetSimSignalDuty()

```
EXPORT int PwrSnsr_SetSimSignalDuty (
    SessionID Vi,
    const char * Channel,
    float Duty
)
```

Set the simulated signal duty cycle in percent. Affects Periodic only.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Duty	Duty cycle in percent (< 99.0).

Returns

Success (0) or error code.

© 2019 WTG

◆PwrSnsr_SetSimSignalModulation()

EXPORT int PwrSnsr_SetSimSignalModulation	(	**SessionID**	Vi,
		const char *	Channel,
		float	Percent
	)		

Set the percent modulation for the signal on a simulation channel.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
Percent	The percent modulation.

Returns

Success (0) or error code.

◆PwrSnsr_SetSimSignalPRF()

EXPORT int PwrSnsr_SetSimSignalPRF	(	**SessionID**	Vi,
		const char *	Channel,
		float	PRF
	)		

Set the simulated signal PRF. Valid for Periodic and Burst.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
PRF	Pulse repetition frequency in Hz.

Returns

Success (0) or error code.

© 2019 WTG

◆ PwrSnsr_SetSimSignalType()

EXPORT int PwrSnsr_SetSimSignalType	(	**SessionID**	Vi,
		const char *	Channel,
		PwrSnsrSimSignalType	SignalType
	)		

Set the simulated channel signal type.

Parameters

Vi	The SessionID handle that you obtain from the PwrSnsr_OpenSimMeter function. The handle identifies a particular instrument session.
Channel	Channel number. For single instruments, set this to "CH1".
SignalType	The simulated signal type.

Returns

Success (0) or error code.

Generated by doxygen 1.8.15

© 2019 WTG

Data Structures

2 Data Structures

Power Sensor Library 1.1.0

Data Structures

Here are the data structures with brief descriptions:

C **PulseInfo**	Data structure containing pulse information

Generated by doxygen 1.8.15

2. .1 PulseInfo

Power Sensor Library 1.1.0

Data Fields

PulseInfo Struct Reference

Multiple Pulse

Data structure containing pulse information. More...

Data Fields

float	**Width**
float	**Peak**
float	**Min**
float	**PulseAvg**
float	**Position**
float	**RiseProximal**
float	**RiseDistal**

© 2019 WTG

float	**RiseTime**
float	**FallProximal**
float	**FallDistal**
float	**FallTime**

Detailed Description

Data structure containing pulse information.

Field Documentation

◆ FallDistal

float FallDistal

Position in time for the distal crossing on the falling edge of the pulse.

◆ FallProximal

float FallProximal

Position in time for the proximal crossing on the falling edge of the pulse.

◆ FallTime

float FallTime

Fall time of the pulse.

◆ Min

float Min

Minimum instanaeous power measurement.

© 2019 WTG

◆Peak

float Peak

Peak (max instanaeous) power measurement.

◆Position

float Position

Time position corresponding to the mesial crossing of the rising edge for the pulse.

◆PulseAvg

float PulseAvg

Average power measurement for the pulse.

◆RiseDistal

float RiseDistal

Position in time for the distal crossing on the rising edge of the pulse.

◆RiseProximal

float RiseProximal

Position in time for the proximal crossing on the rising edge of the pulse.

◆RiseTime

float RiseTime

Rise time of the pulse.

◆Width

float Width

Pulse width is defined as the interval between the first and second signal crossings of the mesial line.

The documentation for this struct was generated from the following file:

© 2019 WTG

- **PwrSnsrLib.h**

Generated by doxygen 1.8.15

2.2 Data Structure Index

Power Sensor Library 1.1.0

Data Structure Index

	p

p	
PulseInfo	

	p

Generated by doxygen 1.8.15

2.3 Data Fields

Power Sensor Library 1.1.0

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- FallDistal : **PulseInfo**
- FallProximal : **PulseInfo**
- FallTime : **PulseInfo**
- Min : **PulseInfo**
- Peak : **PulseInfo**
- Position : **PulseInfo**
- PulseAvg : **PulseInfo**

© 2019 WTG

- RiseDistal : **PulseInfo**
- RiseProximal : **PulseInfo**
- RiseTime : **PulseInfo**
- Width : **PulseInfo**

Generated by doxygen 1.8.15

2.3.2 Variables

Power Sensor Library 1.1.0

- FallDistal : **PulseInfo**
- FallProximal : **PulseInfo**
- FallTime : **PulseInfo**
- Min : **PulseInfo**
- Peak : **PulseInfo**
- Position : **PulseInfo**
- PulseAvg : **PulseInfo**
- RiseDistal : **PulseInfo**
- RiseProximal : **PulseInfo**
- RiseTime : **PulseInfo**
- Width : **PulseInfo**

Generated by doxygen 1.8.15

© 2019 WTG

- A -

- C -

- D -

- L -

- M -

- P -

- S -

- T -

- U -

© 2019 WTG