Standard WSA API Library

Generated by Doxygen 1.7.4

Thu Oct 6 2011 14:15:28

Contents

1 Introduction

This documentation, compiled using Doxygen, describes in details the wsa_api library. The wsa_api provides functions to set/get particular settings or acquire data from the WSA. The wsa_api encodes the commands into SCPI syntax scripts, which are sent to a WSA through the wsa_lib library. Subsequently, it decodes any responses or packet coming back from the WSA through the wsa_lib. Thus, the API helps to abstract away SCPI syntax from the user.

Data frames passing back from the wsa_lib are in VRT format. This API will extract the information and the actual data frames within the VRT packet and makes them available in structures and buffers for users.

1.1 Limitations in v1.0

The following features are not yet supported with the CLI:

- DC correction. Need Nikhil to clarify on that.
- IQ correction. Same as above.
- Automatic finding of a WSA box(s) on a network.
- Set sample sizes. 1024 size for now.
- Triggers.
- Gain calibrarion. TBD with triggers.
- USB interface method - might never be available.

1.2 How to use the library

The wsa_api is designed using mixed C/C++ languages. To use the library, you need to include the header file, wsa_api.h, in files that will use any of its functions to access a WSA, and a link to the wsa_api.lib.

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Data Structure Documentation

4.1 wsa_descriptor Struct Reference

This structure stores WSA information.

Data Fields

- char prod_name [50]
- char prod_serial [20]
- char prod_version [20]
- char rfe_name [50]
- char rfe_version [20]
- char fw_version [20]
- char intf_type [20]
- uint64_t inst_bw
- uint64_t max_sample_size

- uint64_t max_tune_freq
- uint64_t min_tune_freq
- uint64_t freq_resolution
- float max_if_gain
- float min_if_gain
- float abs_max_amp [NUM_RF_GAINS]

4.1.1 Field Documentation

4.1.1.1 float **abs_max_amp**

An array storing the absolute maximum RF input level in dBm for each RF gain setting of the RFE use. Operating a WSA device at these absolute maximums may cause damage to the device.

4.1.1.2 uint64_t **freq_resolution**

The frequency resolution in Hz that a WSA's centre frequency can be incremented.

4.1.1.3 char **fw_version**

The firmware version currently in the WSA.

4.1.1.4 uint64_t **inst_bw**

The WSA instantaneous bandwidth in Hz.

4.1.1.5 char **intf_type**

The interface method to a WSA. Available: "TCPIP" ("USB" TBD).

4.1.1.6 float **max_if_gain**

The maximum IF gain in dB that a WSA's RFE can be set.

4.1.1.7 uint32_t **max_sample_size**

The maximum number of continuous I and Q data samples the WSA can capture per frame.

4.1.1.8 uint64_t **max_tune_freq**

The maximum frequency in Hz that a WSA's RFE can be tuned to.

4.1.1.9 float **min_if_gain**

The minimum IF gain in dB that a WSA's RFE can be set.

4.1.1.10 uint64_t **min_tune_freq**

The minimum frequency in Hz that a WSA's RFE can be tuned to.

4.1.1.11 char prod_name

WSA product name.

4.1.1.12 char prod_serial

WSA product serial number.

4.1.1.13 char prod_version

WSA product version number.

4.1.1.14 char rfe_name

WSA product name.

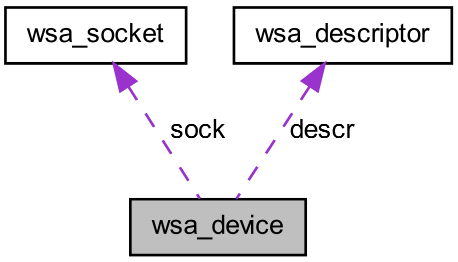
4.1.1.15 char rfe_version

WSA product version number.

4.2 wsa_device Struct Reference

A structure containing the components associate with each WSA device.

Collaboration diagram for wsa_device:



Data Fields

- struct wsa_descriptor descr
- struct wsa_socket sock

4.2.1 Field Documentation

4.2.1.1 struct wsa_descriptor descr

The information component of the WSA, stored in wsa_descriptor.

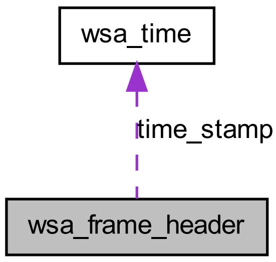
4.2.1.2 struct wsa_socket sock

The socket structure component of the WSA, used for TCPIP connection.

4.3 wsa_frame_header Struct Reference

This structure contains header information related to each frame read by wsa_get_-frame().

Collaboration diagram for wsa_frame_header:



Data Fields

- uint32_t sample_size
- struct wsa_time time_stamp

4.3.1 Field Documentation

4.3.1.1 uint32_t sample_size

Number of {I, Q} samples pairs per WSA data frame.

4.3.1.2 struct wsa_time time_stamp

The time when a data frame capture begins, stored in wsa_time structure.

4.4 wsa_resp Struct Reference

This structure contains the response information for each query.

Data Fields

- int64_t status
- char result [MAX_STR_LEN]

4.4.1 Field Documentation

4.4.1.1 char result

The resulted string responded to a query.

4.4.1.2 int32_t status

The status of the query. Positive number when success, negative when failed.

4.5 wsa_socket Struct Reference

A structure containing the socket parameters used for creating TCP/IP connection for control and data acquisition.

Data Fields

- SOCKET cmd
- SOCKET data

4.5.1 Field Documentation

4.5.1.1 SOCKET cmd

The command socket for command controls and queries. The string protocol used for this socket is HISLIP.

4.5.1.2 SOCKET data

The data socket used for streaming of data

4.6 wsa_time Struct Reference

This structure contains the time information. It is used for the time stamp in a frame header.

Data Fields

- int32_t sec
- uint32_t nsec

4.6.1 Field Documentation

4.6.1.1 int32_t nsec

Nanoseconds after the second (0 - 999 999 999).

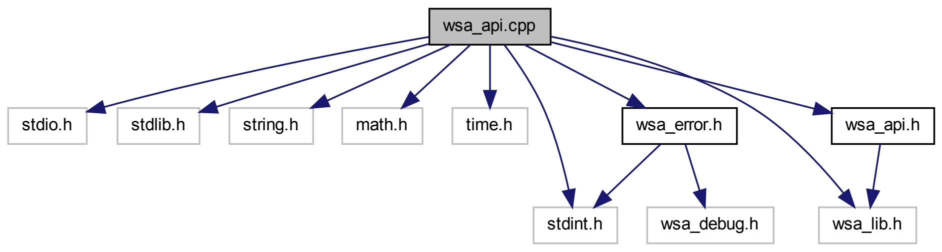
4.6.1.2 int32_t sec

The number of seconds elapsed since 00:00 hours, Jan 1, 1970 UTC.

5 File Documentation

5.1 wsa_api.cpp File Reference

Include dependency graph for wsa_api.cpp:



Defines

- #define MAX_ANT_PORT 2
- #define SCPI_OSR_CALI 0x0001
- #define SCPI_OSR_SETT 0x0002
- #define SCPI_OSR_SWE 0x0008
- #define SCPI_OSR_MEAS 0x0010
- #define SCPI_OSR_TRIG 0x0020
- #define SCPI_OSR_CORR 0x0080
- #define SCPI_OSR_DATA 0x0100

Functions

- int16_t wsa_verify_freq (struct wsa_device ∗dev, uint64_t freq)
- int16_t wsa_open (struct wsa_device ∗dev, char ∗intf_method)
- void wsa_close (struct wsa_device ∗dev)
- int16_t wsa_check_addr (char ∗ip_addr)
- int16_t wsa_list (char ∗∗wsa_list)
- int16_t wsa_is_connected (struct wsa_device ∗dev)
- int16_t wsa_set_command_file (struct wsa_device ∗dev, char ∗file_name)
- float wsa_get_abs_max_amp (struct wsa_device ∗dev, wsa_gain gain)
- int32_t wsa_read_pkt (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int16_t ∗i_buf, int16_t ∗q_buf, const uint32_t sample_size)
- int16_t wsa_set_sample_size (struct wsa_device ∗dev, int32_t sample_size)
- int32_t wsa_get_sample_size (struct wsa_device ∗dev)
- int64_t wsa_get_freq (struct wsa_device ∗dev)
- int16_t wsa_set_freq (struct wsa_device ∗dev, uint64_t cfreq)
- float wsa_get_gain_if (struct wsa_device ∗dev)
- int16_t wsa_set_gain_if (struct wsa_device ∗dev, float gain)
- wsa_gain wsa_get_gain_rf (struct wsa_device ∗dev)
- int16_t wsa_set_gain_rf (struct wsa_device ∗dev, wsa_gain gain)
- int16_t wsa_get_antenna (struct wsa_device ∗dev)
- int16_t wsa_set_antenna (struct wsa_device ∗dev, uint8_t port_num)
- int16_t wsa_get_bpf (struct wsa_device ∗dev)
- int16_t wsa_set_bpf (struct wsa_device ∗dev, uint8_t mode)
- int16_t wsa_query_cal_mode (struct wsa_device ∗dev)
- int16_t wsa_run_cal_mode (struct wsa_device ∗dev, uint8_t mode)

5.1.1 Define Documentation

5.1.1.1 #define MAX_ANT_PORT 2

5.1.1.2 #define SCPI_OSR_CALI 0x0001

5.1.1.3 #define SCPI_OSR_CORR 0x0080

5.1.1.4 #define SCPI_OSR_DATA 0x0100

5.1.1.5 #define SCPI_OSR_MEAS 0x0010

5.1.1.6 #define SCPI_OSR_SETT 0x0002

5.1.1.7 #define SCPI_OSR_SWE 0x0008

5.1.1.8 #define SCPI_OSR_TRIG 0x0020

5.1.2 Function Documentation

5.1.2.1 int16_t wsa_check_addr (char ∗ *ip_addr*)

Verify if the IP address or host name given is valid for the WSA.

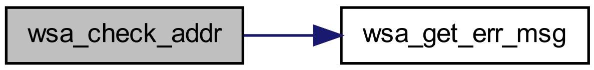
Parameters

ip_addr	- A char pointer to the IP address or host name to be verified.

Returns

1 if the IP is valid, or a negative number on error.

Here is the call graph for this function:



5.1.2.2 void wsa_close (struct **wsa_device** ∗ *dev*)

Closes the device handle if one is opened and stops any existing data capture.

Parameters

dev	- A pointer to a WSA device structure to be closed.

Returns

none

5.1.2.3 float wsa_get_abs_max_amp (struct **wsa_device** ∗ *dev,* **wsa_gain** *gain*)

Gets the absolute maximum RF input level (dBm) for the WSA at the given gain setting.

Operating the WSA device at the absolute maximum may cause damage to the device.

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain setting of **wsa_gain** type at which the absolute maximum amplitude input level is to be retrieved.

Returns

The absolute maximum RF input level in dBm or negative error number.

5.1.2.4 int16_t wsa_get_antenna (struct **wsa_device** ∗ *dev*)

Gets which antenna port is currently in used with the RFE board.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The antenna port number on success, or a negative number on error.

5.1.2.5 int16_t wsa_get_bpf (struct wsa_device * *dev*)

Gets the current mode of the RFE's preselect BPF stage.

Parameters

dev	- A pointer to the WSA device structure.

Returns

1 (on), 0 (off), or a negative number on error.

5.1.2.6 int64_t wsa_get_freq (struct wsa_device * *dev*)

Retrieves the center frequency that the WSA is running at.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The frequency in Hz, or a negative number on error.

5.1.2.7 float wsa_get_gain_if (struct wsa_device * *dev*)

Gets the current IF gain value of the RFE in dB.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The gain value in dB, or a large negative number on error.

5.1.2.8 wsa_gain wsa_get_gain_rf (struct wsa_device * *dev*)

Gets the current quantized RF front end gain setting of the RFE.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The gain setting of wsa_gain type, or a negative number on error.

5.1.2.9 int32_t wsa_get_sample_size (struct **wsa_device** ∗ *dev*)

Gets the number of samples per frame.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The sample size if success, or a negative number on error.

5.1.2.10 int16_t wsa_is_connected (struct **wsa_device** ∗ *dev*)

Indicates if the WSA is still connected to the PC.

Parameters

dev	- A pointer to the WSA device structure to be verified for the connection.

Returns

1 if it is connected, 0 if not connected, or a negative number if errors.

5.1.2.11 int16_t wsa_list (char ∗∗ *wsa_list*)

Count and print out the IPs of connected WSAs to the network? or the PC??? For now, will list the IPs for any of the connected devices to a PC?

Parameters

wsa_list	- A double char pointer to store (WSA???) IP addresses connected to a network???.

Returns

Number of connected WSAs (or IPs for now) on success, or a negative number on error.

5.1.2.12 int16_t wsa_open (struct **wsa_device** ∗ *dev,* char ∗ *intf_method*)

Establishes a connection of choice specified by the interface method to the WSA.

At success, the handle remains open for future access by other library methods until wsa_close() is called. When unsuccessful, the WSA will be closed automatically and an error is returned.

Parameters

dev	- A pointer to the WSA device structure to be opened.
intf_method	- A char pointer to store the interface method to the WSA. Possible methods: • With LAN, use: "TCPIP::<Ip address of the WSA>::HISLIP" • With USB, use: "USB" (check if supported with the WSA version used).

Returns

0 on success, or a negative number on error.

Errors:

Situations that will generate an error are:

- the interface method does not exist for the WSA product version.
- the WSA is not detected (has not been connected or powered up).
-

5.1.2.13 int16_t wsa_query_cal_mode (struct wsa_device * *dev*)

Gets the current mode of the RFE's internal anti-aliasing LPF.

Parameters

dev	- A pointer to the WSA device structure.

Returns

1 (on), 0 (off), or a negative number on error. Checks if the RFE's internal calibration has finished or not.

Parameters

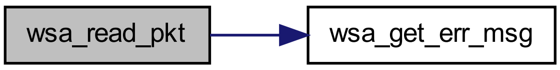
dev	- A pointer to the WSA device structure.

Returns

1 if the calibration is still running or 0 if completed, or a negative number on error.

5.1.2.14 int32_t wsa_read_pkt (struct wsa_device ∗ *dev*, struct wsa_frame_header ∗ *header*, int16_t ∗ *i_buf*, int16_t ∗ *q_buf*, const uint32_t *sample_size*)

Here is the call graph for this function:



5.1.2.15 int16_t wsa_run_cal_mode (struct wsa_device ∗ *dev*, uint8_t *mode*)

Runs the RFE'S internal calibration mode or cancel it.

While the calibration mode is running, no other commands should be running until the calibration is finished by using wsa_query_cal_mode(), or could be cancelled

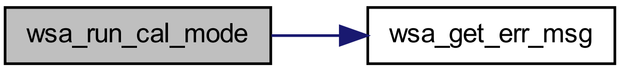
Parameters

dev	- A pointer to the WSA device structure.
mode	- An integer mode of selection: 1 - Run, 0 - Cancel.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.1.2.16 int16_t wsa_set_antenna (struct wsa_device ∗ *dev*, uint8_t *port_num*)

Sets the antenna port to be used for the RFE board.

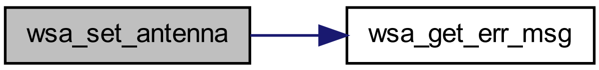
Parameters

dev	- A pointer to the WSA device structure.
port_num	- An integer port number to used. Available ports: 1, 2. Or see product datasheet for ports availability. **Note:** When calibration mode is enabled through wsa_run_cal_mode(), these antenna ports will not be available. The seletected port will resume when the calibration mode is set to off.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.1.2.17 int16_t wsa_set_bpf (struct wsa_device * *dev*, uint8_t *mode*)

Sets the RFE's preselect band pass filter (BPF) stage on or off (bypassing).

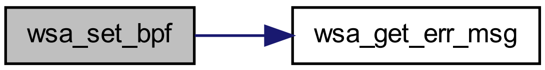
Parameters

dev	- A pointer to the WSA device structure.
mode	- An integer mode of selection: 0 - Off, 1 - On.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.1.2.18 int16_t wsa_set_command_file (struct wsa_device ∗ *dev,* char ∗ *file_name*)

Read command line(s) stored in the given **file_name** and set each line to the WSA.

Remarks

- Assuming each command line is for a single function followed by a new line.
- Currently read only SCPI commands. Other types of commands, TBD.

Parameters

dev	- A pointer to the WSA device structure.
file_name	- A pointer to the file name

Returns

Number of command lines at success, or a negative error number.

5.1.2.19 int16_t wsa_set_freq (struct wsa_device ∗ *dev,* uint64_t *cfreq*)

Sets the WSA to the desired center frequency, **cfreq**.

Remarks

wsa_set_freq() will return error if trigger mode is already running. See the **descr** component of **wsa_dev** structure for maximum/minimum frequency values.

Parameters

dev	- A pointer to the WSA device structure.
cfreq	- The center frequency to set, in Hz

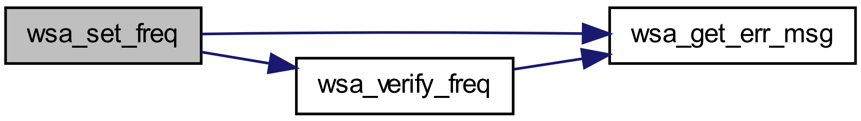
Returns

0 on success, or a negative number on error.

Errors:

- Frequency out of range.
- Set frequency when WSA is in trigger mode.
- Incorrect frequency resolution (check with data sheet).

Here is the call graph for this function:



5.1.2.20 int16_t wsa_set_gain_if (struct wsa_device * *dev*, float *gain*)

Sets the gain value in dB for the variable IF gain stages of the RFE, which is additive to the primary RF quantized gain stages (wsa_set_gain_rf()).

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain level in dB.

Remarks

See the **descr** component of **wsa_dev** structure for maximum/minimum IF gain values. ???

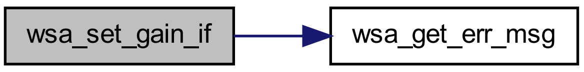
Returns

0 on success, or a negative number on error.

Errors:

- Gain level out of range.

Here is the call graph for this function:



5.1.2.21 int16_t wsa_set_gain_rf (struct **wsa_device** ∗ *dev*, **wsa_gain** *gain*)

Sets the quantized **gain** (sensitivity) level for the RFE of the WSA.

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain setting of type wsa_gain to set for WSA. Valid gain settings are: • WSA_GAIN_HIGH • WSA_GAIN_MEDIUM • WSA_GAIN_LOW • WSA_GAIN_VLOW

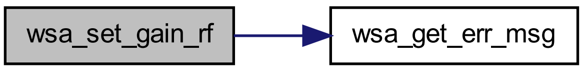
Returns

0 on success, or a negative number on error.

Errors:

- Gain setting not allow.

Here is the call graph for this function:



5.1.2.22 int16_t wsa_set_sample_size (struct **wsa_device** ∗ *dev*, int32_t *sample_size*)

Sets the number of samples per frame to be received

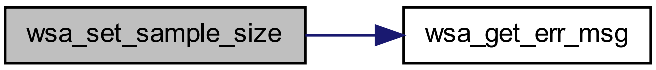
Parameters

dev	- A pointer to the WSA device structure.
sample_size	- The sample size to set.

Returns

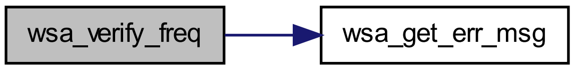
0 if success, or a negative number on error.

Here is the call graph for this function:



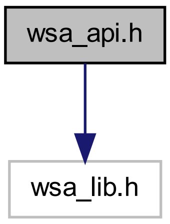
5.1.2.23 int16_t wsa_verify_freq (struct **wsa_device** ∗ *dev*, uint64_t *freq*)

Here is the call graph for this function:

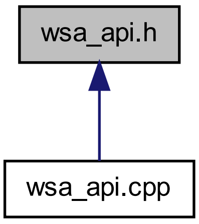


5.2 wsa_api.h File Reference

Include dependency graph for wsa_api.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct wsa_descriptor

  This structure stores WSA information.
- struct wsa_time

  This structure contains the time information. It is used for the time stamp in a frame header.
- struct wsa_frame_header

  This structure contains header information related to each frame read by wsa_get_-frame().
- struct wsa_socket

  A structure containing the socket parameters used for creating TCP/IP connection for control and data acquisition.
- struct wsa_device

  A structure containing the components associate with each WSA device.
- struct wsa_resp

  This structure contains the response information for each query.

Enumerations

- enum wsa_gain { WSA_GAIN_HIGH = 1, WSA_GAIN_MEDIUM, WSA_GAIN_-LOW, WSA_GAIN_VLOW }

Functions

- int16_t wsa_open (struct wsa_device ∗dev, char ∗intf_method)
- void wsa_close (struct wsa_device ∗dev)
- int16_t wsa_check_addr (char ∗intf_method)
- int16_t wsa_list (char ∗∗wsa_list)

- int16_t wsa_is_connected (struct wsa_device ∗dev)
- int16_t wsa_set_command_file (struct wsa_device ∗dev, char ∗file_name)
- float wsa_get_abs_max_amp (struct wsa_device ∗dev, wsa_gain gain)
- int32_t wsa_read_pkt (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int16_t ∗i_buf, int16_t ∗q_buf, const uint32_t sample_size)
- int16_t wsa_set_sample_size (struct wsa_device ∗dev, int32_t sample_size)
- int32_t wsa_get_sample_size (struct wsa_device ∗dev)
- int64_t wsa_get_freq (struct wsa_device ∗dev)
- int16_t wsa_set_freq (struct wsa_device ∗dev, uint64_t cfreq)
- float wsa_get_gain_if (struct wsa_device ∗dev)
- int16_t wsa_set_gain_if (struct wsa_device ∗dev, float gain)
- wsa_gain wsa_get_gain_rf (struct wsa_device ∗dev)
- int16_t wsa_set_gain_rf (struct wsa_device ∗dev, wsa_gain gain)
- int16_t wsa_get_antenna (struct wsa_device ∗dev)
- int16_t wsa_set_antenna (struct wsa_device ∗dev, uint8_t port_num)
- int16_t wsa_get_bpf (struct wsa_device ∗dev)
- int16_t wsa_set_bpf (struct wsa_device ∗dev, uint8_t mode)
- int16_t wsa_query_cal_mode (struct wsa_device ∗dev)
- int16_t wsa_run_cal_mode (struct wsa_device ∗dev, uint8_t mode)

5.2.1 Enumeration Type Documentation

5.2.1.1 enum wsa_gain

Defines the RF quantized gain settings available for the radio front end (RFE) of the WSA.

Enumerator:

WSA_GAIN_HIGH	High RF amplification. Value 1.
WSA_GAIN_MEDIUM	Medium RF amplification.
WSA_GAIN_LOW	Low RF amplification.
WSA_GAIN_VLOW	Very low RF amplification.

5.2.2 Function Documentation

5.2.2.1 int16_t wsa_check_addr (char ∗ *ip_addr*)

Verify if the IP address or host name given is valid for the WSA.

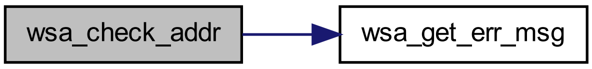
Parameters

ip_addr	- A char pointer to the IP address or host name to be verified.

Returns

1 if the IP is valid, or a negative number on error.

Here is the call graph for this function:



5.2.2.2 void wsa_close (struct **wsa_device** * *dev*)

Closes the device handle if one is opened and stops any existing data capture.

Parameters

dev	- A pointer to a WSA device structure to be closed.

Returns

none

5.2.2.3 float wsa_get_abs_max_amp (struct **wsa_device** * *dev*, **wsa_gain** *gain*)

Gets the absolute maximum RF input level (dBm) for the WSA at the given gain setting.

Operating the WSA device at the absolute maximum may cause damage to the device.

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain setting of **wsa_gain** type at which the absolute maximum amplitude input level is to be retrieved.

Returns

The absolute maximum RF input level in dBm or negative error number.

5.2.2.4 int16_t wsa_get_antenna (struct **wsa_device** * *dev*)

Gets which antenna port is currently in used with the RFE board.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The antenna port number on success, or a negative number on error.

5.2.2.5 int16_t wsa_get_bpf (struct wsa_device * *dev*)

Gets the current mode of the RFE's preselect BPF stage.

Parameters

dev	- A pointer to the WSA device structure.

Returns

1 (on), 0 (off), or a negative number on error.

5.2.2.6 int64_t wsa_get_freq (struct wsa_device * *dev*)

Retrieves the center frequency that the WSA is running at.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The frequency in Hz, or a negative number on error.

5.2.2.7 float wsa_get_gain_if (struct wsa_device * *dev*)

Gets the current IF gain value of the RFE in dB.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The gain value in dB, or a large negative number on error.

5.2.2.8 wsa_gain wsa_get_gain_rf (struct wsa_device * *dev*)

Gets the current quantized RF front end gain setting of the RFE.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The gain setting of wsa_gain type, or a negative number on error.

5.2.2.9 int32_t wsa_get_sample_size (struct wsa_device * *dev*)

Gets the number of samples per frame.

Parameters

dev	- A pointer to the WSA device structure.

Returns

The sample size if success, or a negative number on error.

5.2.2.10 int16_t wsa_is_connected (struct wsa_device * *dev*)

Indicates if the WSA is still connected to the PC.

Parameters

dev	- A pointer to the WSA device structure to be verified for the connection.

Returns

1 if it is connected, 0 if not connected, or a negative number if errors.

5.2.2.11 int16_t wsa_list (char ** *wsa_list*)

Count and print out the IPs of connected WSAs to the network? or the PC??? For now, will list the IPs for any of the connected devices to a PC?

Parameters

wsa_list	- A double char pointer to store (WSA???) IP addresses connected to a network???.

Returns

Number of connected WSAs (or IPs for now) on success, or a negative number on error.

5.2.2.12 int16_t wsa_open (struct wsa_device * *dev*, char * *intf_method*)

Establishes a connection of choice specified by the interface method to the WSA.

At success, the handle remains open for future access by other library methods until wsa_close() is called. When unsuccessful, the WSA will be closed automatically and an error is returned.

Parameters

dev	- A pointer to the WSA device structure to be opened.
intf_method	- A char pointer to store the interface method to the WSA. Possible methods: • With LAN, use: "TCPIP::<Ip address of the WSA>::HISLIP" • With USB, use: "USB" (check if supported with the WSA version used).

Returns

0 on success, or a negative number on error.

Errors:

Situations that will generate an error are:

- the interface method does not exist for the WSA product version.
- the WSA is not detected (has not been connected or powered up).
-

5.2.2.13 int16_t wsa_query_cal_mode (struct wsa_device ∗ *dev*)

Gets the current mode of the RFE's internal anti-aliasing LPF.

Parameters

dev	- A pointer to the WSA device structure.

Returns

1 (on), 0 (off), or a negative number on error. Checks if the RFE's internal calibration has finished or not.

Parameters

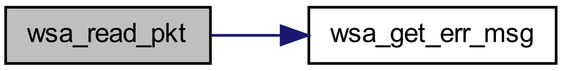
dev	- A pointer to the WSA device structure.

Returns

1 if the calibration is still running or 0 if completed, or a negative number on error.

5.2.2.14 int32_t wsa_read_pkt (struct wsa_device ∗ *dev,* struct wsa_frame_header ∗ *header,* int16_t ∗ *i_buf,* int16_t ∗ *q_buf,* const uint32_t *sample_size*)

Here is the call graph for this function:



5.2.2.15 int16_t wsa_run_cal_mode (struct wsa_device ∗ *dev,* uint8_t *mode*)

Runs the RFE'S internal calibration mode or cancel it.

While the calibration mode is running, no other commands should be running until the calibration is finished by using wsa_query_cal_mode(), or could be cancelled

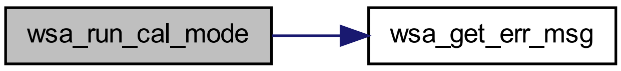
Parameters

dev	- A pointer to the WSA device structure.
mode	- An integer mode of selection: 1 - Run, 0 - Cancel.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.2.2.16 int16_t wsa_set_antenna (struct wsa_device * *dev*, uint8_t *port_num*)

Sets the antenna port to be used for the RFE board.

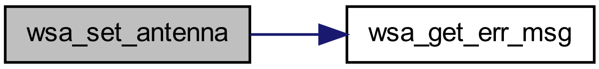
Parameters

dev	- A pointer to the WSA device structure.
port_num	- An integer port number to used. Available ports: 1, 2. Or see product datasheet for ports availability. **Note:** When calibration mode is enabled through wsa_run_cal_mode(), these antenna ports will not be available. The seletected port will resume when the calibration mode is set to off.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.2.2.17 int16_t wsa_set_bpf (struct wsa_device * *dev*, uint8_t *mode*)

Sets the RFE's preselect band pass filter (BPF) stage on or off (bypassing).

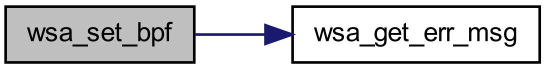
Parameters

dev	- A pointer to the WSA device structure.
mode	- An integer mode of selection: 0 - Off, 1 - On.

Returns

0 on success, or a negative number on error.

Here is the call graph for this function:



5.2.2.18 int16_t wsa_set_command_file (struct wsa_device * *dev*, char * *file_name*)

Read command line(s) stored in the given **file_name** and set each line to the WSA.

Remarks

- Assuming each command line is for a single function followed by a new line.
- Currently read only SCPI commands. Other types of commands, TBD.

Parameters

dev	- A pointer to the WSA device structure.
file_name	- A pointer to the file name

Returns

Number of command lines at success, or a negative error number.

5.2.2.19 int16_t wsa_set_freq (struct wsa_device ∗ *dev,* uint64_t *cfreq*)

Sets the WSA to the desired center frequency, **cfreq**.

Remarks

wsa_set_freq() will return error if trigger mode is already running. See the **descr** component of **wsa_dev** structure for maximum/minimum frequency values.

Parameters

dev	- A pointer to the WSA device structure.
cfreq	- The center frequency to set, in Hz

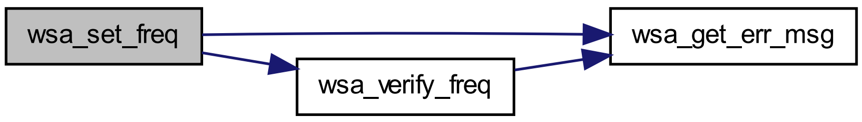
Returns

0 on success, or a negative number on error.

Errors:

- Frequency out of range.
- Set frequency when WSA is in trigger mode.
- Incorrect frequency resolution (check with data sheet).

Here is the call graph for this function:



5.2.2.20 int16_t wsa_set_gain_if (struct wsa_device ∗ *dev,* float *gain*)

Sets the gain value in dB for the variable IF gain stages of the RFE, which is additive to the primary RF quantized gain stages (wsa_set_gain_rf()).

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain level in dB.

Remarks

See the **descr** component of **wsa_dev** structure for maximum/minimum IF gain values. ???

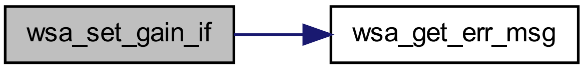
Returns

0 on success, or a negative number on error.

Errors:

- Gain level out of range.

Here is the call graph for this function:



5.2.2.21 int16_t wsa_set_gain_rf (struct **wsa_device** ∗ *dev,* **wsa_gain** *gain*)

Sets the quantized **gain** (sensitivity) level for the RFE of the WSA.

Parameters

dev	- A pointer to the WSA device structure.
gain	- The gain setting of type wsa_gain to set for WSA. Valid gain settings are: • WSA_GAIN_HIGH • WSA_GAIN_MEDIUM • WSA_GAIN_LOW • WSA_GAIN_VLOW

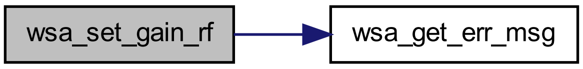
Returns

0 on success, or a negative number on error.

Errors:

- Gain setting not allow.

Here is the call graph for this function:



5.2.2.22 int16_t wsa_set_sample_size (struct wsa_device * *dev*, int32_t *sample_size*)

Sets the number of samples per frame to be received

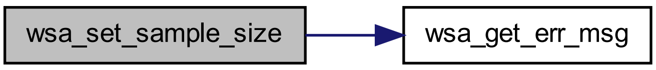
Parameters

dev	- A pointer to the WSA device structure.
sample_size	- The sample size to set.

Returns

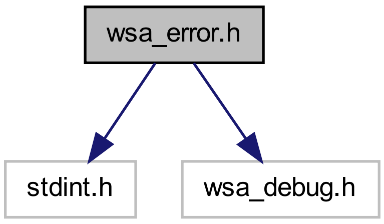
0 if success, or a negative number on error.

Here is the call graph for this function:

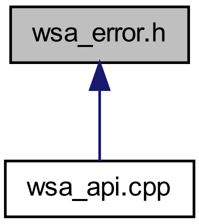


5.3 wsa_error.h File Reference

Include dependency graph for wsa_error.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define LNEG_NUM (-10000)
- #define WSA_ERR_NOWSA (LNEG_NUM - 1)
- #define WSA_ERR_INVIPADDRESS (LNEG_NUM - 2)
- #define WSA_ERR_NOCTRLPIPE (LNEG_NUM - 3)
- #define WSA_ERR_UNKNOWNPRODSER (LNEG_NUM - 4)
- #define WSA_ERR_UNKNOWNPRODVSN (LNEG_NUM - 5)
- #define WSA_ERR_UNKNOWNFWRVSN (LNEG_NUM - 6)
- #define WSA_ERR_UNKNOWNRFEVSN (LNEG_NUM - 7)
- #define WSA_ERR_PRODOBSOLETE (LNEG_NUM - 8)

- #define WSA_ERR_QUERYNORESP (LNEG_NUM - 9)
- #define WSA_ERR_WSANOTRDY (LNEG_NUM - 101)
- #define WSA_ERR_WSAINUSE (LNEG_NUM - 102)
- #define WSA_ERR_SETFAILED (LNEG_NUM - 103)
- #define WSA_ERR_OPENFAILED (LNEG_NUM - 104)
- #define WSA_ERR_INITFAILED (LNEG_NUM - 105)
- #define WSA_ERR_INVADCCORRVALUE (LNEG_NUM - 106)
- #define WSA_ERR_INVINTFMETHOD (LNEG_NUM - 201)
- #define WSA_ERR_INVIPHOSTADDRESS (LNEG_NUM - 202)
- #define WSA_ERR_USBNOTAVBL (LNEG_NUM - 203)
- #define WSA_ERR_USBOPENFAILED (LNEG_NUM - 204)
- #define WSA_ERR_USBINITFAILED (LNEG_NUM - 205)
- #define WSA_ERR_ETHERNETNOTAVBL (LNEG_NUM - 206)
- #define WSA_ERR_ETHERNETCONNECTFAILED (LNEG_NUM - 207)
- #define WSA_ERR_ETHERNETINITFAILED (LNEG_NUM - 209)
- #define WSA_ERR_WINSOCKSTARTUPFAILED (LNEG_NUM - 210)
- #define WSA_ERR_SOCKETSETFUPFAILED (LNEG_NUM - 211)
- #define WSA_ERR_INVAMP (LNEG_NUM - 301)
- #define WSA_ERR_NODATABUS (LNEG_NUM - 401)
- #define WSA_ERR_READFRAMEFAILED (LNEG_NUM - 402)
- #define WSA_ERR_INVSAMPLESIZE (LNEG_NUM - 403)
- #define WSA_ERR_SIZESETFAILED (LNEG_NUM - 404)
- #define WSA_ERR_NOTIQFRAME (LNEG_NUM - 405)
- #define WSA_ERR_FREQOUTOFBOUND (LNEG_NUM - 601)
- #define WSA_ERR_INVFREQRES (LNEG_NUM - 602)
- #define WSA_ERR_FREQSETFAILED (LNEG_NUM - 603)
- #define WSA_ERR_PLLLOCKFAILED (LNEG_NUM - 604)
- #define WSA_ERR_INVRFGAIN (LNEG_NUM - 801)
- #define WSA_ERR_INVIFGAIN (LNEG_NUM - 802)
- #define WSA_ERR_IFGAINSETFAILED (LNEG_NUM - 803)
- #define WSA_ERR_RFGAINSETFAILED (LNEG_NUM - 804)
- #define WSA_ERR_INVRUNMODE (LNEG_NUM - 1001)
- #define WSA_ERR_INVTRIGID (LNEG_NUM - 1201)
- #define WSA_ERR_INVSTOPFREQ (LNEG_NUM - 1202)
- #define WSA_ERR_STARTOOB (LNEG_NUM - 1203)
- #define WSA_ERR_STOPOOB (LNEG_NUM - 1204)
- #define WSA_ERR_INVSTARTRES (LNEG_NUM - 1205)
- #define WSA_ERR_INVSTOPRES (LNEG_NUM - 1206)
- #define WSA_ERR_INVTRIGRANGE (LNEG_NUM - 1207)
- #define WSA_ERR_INVDWELL (LNEG_NUM - 1208)
- #define WSA_ERR_INVNUMFRAMES (LNEG_NUM - 1209)
- #define WSA_ERR_CMDSENDFAILED (LNEG_NUM - 1501)
- #define WSA_ERR_CMDINVALID (LNEG_NUM - 1502)
- #define WSA_ERR_INVANTENNAPORT (LNEG_NUM - 1601)
- #define WSA_ERR_ANTENNASETFAILED (LNEG_NUM - 1602)
- #define WSA_ERR_INVFILTERMODE (LNEG_NUM - 1603)

- #define WSA_ERR_FILTERSETFAILED (LNEG_NUM - 1604)
- #define WSA_ERR_INVCALIBRATEMODE (LNEG_NUM - 1605)
- #define WSA_ERR_CALIBRATESETFAILED (LNEG_NUM - 1606)
- #define WSA_ERR_FILECREATEFAILED (LNEG_NUM - 1900)
- #define WSA_ERR_FILEOPENFAILED (LNEG_NUM - 1901)
- #define WSA_ERR_FILEREADFAILED (LNEG_NUM - 1902)
- #define WSA_ERR_FILEWRITEFAILED (LNEG_NUM - 1903)
- #define WSA_ERR_INVNUMBER (LNEG_NUM - 2000)
- #define WSA_ERR_INVREGADDR (LNEG_NUM - 2001)
- #define WSA_ERR_MALLOCFAILED (LNEG_NUM - 2002)
- #define WSA_ERR_UNKNOWN_ERROR (LNEG_NUM - 2003)

Functions

- const char ∗ wsa_get_err_msg (int16_t err_id)

5.3.1 Define Documentation

5.3.1.1 #define LNEG_NUM (-10000)

5.3.1.2 #define WSA_ERR_ANTENNASETFAILED (LNEG_NUM - 1602)

5.3.1.3 #define WSA_ERR_CALIBRATESETFAILED (LNEG_NUM - 1606)

5.3.1.4 #define WSA_ERR_CMDINVALID (LNEG_NUM - 1502)

5.3.1.5 #define WSA_ERR_CMDSENDFAILED (LNEG_NUM - 1501)

5.3.1.6 #define WSA_ERR_ETHERNETCONNECTFAILED (LNEG_NUM - 207)

5.3.1.7 #define WSA_ERR_ETHERNETINITFAILED (LNEG_NUM - 209)

5.3.1.8 #define WSA_ERR_ETHERNETNOTAVBL (LNEG_NUM - 206)

5.3.1.9 #define WSA_ERR_FILECREATEFAILED (LNEG_NUM - 1900)

5.3.1.10 #define WSA_ERR_FILEOPENFAILED (LNEG_NUM - 1901)

5.3.1.11 #define WSA_ERR_FILEREADFAILED (LNEG_NUM - 1902)

5.3.1.12 #define WSA_ERR_FILEWRITEFAILED (LNEG_NUM - 1903)

5.3.1.13 #define WSA_ERR_FILTERSETFAILED (LNEG_NUM - 1604)

5.3.1.14 #define WSA_ERR_FREQOUTOFBOUND (LNEG_NUM - 601)

5.3.1.15 #define WSA_ERR_FREQSETFAILED (LNEG_NUM - 603)

5.3.1.16 #define WSA_ERR_IFGAINSETFAILED (LNEG_NUM - 803)

5.3.1.17 #define WSA_ERR_INITFAILED (LNEG_NUM - 105)

5.3.1.18 #define WSA_ERR_INVADCCORRVALUE (LNEG_NUM - 106)

5.3.1.19 #define WSA_ERR_INVAMP (LNEG_NUM - 301)

5.3.1.20 #define WSA_ERR_INVANTENNAPORT (LNEG_NUM - 1601)

5.3.1.21 #define WSA_ERR_INVCALIBRATEMODE (LNEG_NUM - 1605)

5.3.1.22 #define WSA_ERR_INVDWELL (LNEG_NUM - 1208)

5.3.1.23 #define WSA_ERR_INVFILTERMODE (LNEG_NUM - 1603)

5.3.1.24 #define WSA_ERR_INVFREQRES (LNEG_NUM - 602)

5.3.1.25 #define WSA_ERR_INVIFGAIN (LNEG_NUM - 802)

5.3.1.26 #define WSA_ERR_INVINTFMETHOD (LNEG_NUM - 201)

5.3.1.27 #define WSA_ERR_INVIPADDRESS (LNEG_NUM - 2)

5.3.1.28 #define WSA_ERR_INVIPHOSTADDRESS (LNEG_NUM - 202)

5.3.1.29 #define WSA_ERR_INVNUMBER (LNEG_NUM - 2000)

5.3.1.30 #define WSA_ERR_INVNUMFRAMES (LNEG_NUM - 1209)

5.3.1.31 #define WSA_ERR_INVREGADDR (LNEG_NUM - 2001)

5.3.1.32 #define WSA_ERR_INVRFGAIN (LNEG_NUM - 801)

5.3.1.33 #define WSA_ERR_INVRUNMODE (LNEG_NUM - 1001)

5.3.1.34 #define WSA_ERR_INVSAMPLESIZE (LNEG_NUM - 403)

5.3.1.35 #define WSA_ERR_INVSTARTRES (LNEG_NUM - 1205)

5.3.1.36 #define WSA_ERR_INVSTOPFREQ (LNEG_NUM - 1202)

5.3.1.37 #define WSA_ERR_INVSTOPRES (LNEG_NUM - 1206)

5.3.1.38 #define WSA_ERR_INVTRIGID (LNEG_NUM - 1201)

5.3.1.39 #define WSA_ERR_INVTRIGRANGE (LNEG_NUM - 1207)

5.3.1.40 #define WSA_ERR_MALLOCFAILED (LNEG_NUM - 2002)

5.3.1.41 #define WSA_ERR_NOCTRLPIPE (LNEG_NUM - 3)

5.3.1.42 #define WSA_ERR_NODATABUS (LNEG_NUM - 401)

5.3.1.43 #define WSA_ERR_NOTIQFRAME (LNEG_NUM - 405)

5.3.1.44 #define WSA_ERR_NOWSA (LNEG_NUM - 1)

5.3.1.45 #define WSA_ERR_OPENFAILED (LNEG_NUM - 104)

5.3.1.46 #define WSA_ERR_PLLLOCKFAILED (LNEG_NUM - 604)

5.3.1.47 #define WSA_ERR_PRODOBSOLETE (LNEG_NUM - 8)

5.3.1.48 #define WSA_ERR_QUERYNORESP (LNEG_NUM - 9)

5.3.1.49 #define WSA_ERR_READFRAMEFAILED (LNEG_NUM - 402)

5.3.1.50 #define WSA_ERR_RFGAINSETFAILED (LNEG_NUM - 804)

5.3.1.51 #define WSA_ERR_SETFAILED (LNEG_NUM - 103)

5.3.1.52 #define WSA_ERR_SIZESETFAILED (LNEG_NUM - 404)

5.3.1.53 #define WSA_ERR_SOCKETSETFUPFAILED (LNEG_NUM - 211)

5.3.1.54 #define WSA_ERR_STARTOOB (LNEG_NUM - 1203)

5.3.1.55 #define WSA_ERR_STOPOOB (LNEG_NUM - 1204)

5.3.1.56 #define WSA_ERR_UNKNOWN_ERROR (LNEG_NUM - 2003)

5.3.1.57 #define WSA_ERR_UNKNOWNFWRVSN (LNEG_NUM - 6)

5.3.1.58 #define WSA_ERR_UNKNOWNPRODSER (LNEG_NUM - 4)

5.3.1.59 #define WSA_ERR_UNKNOWNPRODVSN (LNEG_NUM - 5)

5.3.1.60 #define WSA_ERR_UNKNOWNRFEVSN (LNEG_NUM - 7)

5.3.1.61 #define WSA_ERR_USBINITFAILED (LNEG_NUM - 205)

5.3.1.62 #define WSA_ERR_USBNOTAVBL (LNEG_NUM - 203)

5.3.1.63 #define WSA_ERR_USBOPENFAILED (LNEG_NUM - 204)

5.3.1.64 #define WSA_ERR_WINSOCKSTARTUPFAILED (LNEG_NUM - 210)

5.3.1.65 #define WSA_ERR_WSAINUSE (LNEG_NUM - 102)

5.3.1.66 #define WSA_ERR_WSANOTRDY (LNEG_NUM - 101)

5.3.2 Function Documentation

5.3.2.1 const char* wsa_get_err_msg (int16_t *err_id*)

5.4 wsa_lib.txt File Reference

Contain some code documents for wsa_lib.h.

5.4.1 Detailed Description

Index