Standard WSA Library

Generated by Doxygen 1.7.4

Thu Sep 8 2011 16:09:04

Contents

1 Introduction

The wsa_lib is a library with high level interfaces to a WSA device. It abstracts away the actual low level interface and communication through the connection of choice, and subsequently all the controls or commands to the WSA. It allows you to easily control the WSA4000 through standardized command syntax, such as SCPI, to get WSA status, set gain, set centre frequency, etc., and perform data acquisition.

The wsa_lib supports SCPI for control command syntax and VRT for packet.

1.1 How to use the library

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

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

wsa_descriptor (This structure stores WSA information)	**2**
wsa_device (A structure containing the components associate with each WSA device)	**4**
wsa_frame_header (This structure contains header information related to each frame read by wsa_get_frame())	**4**
wsa_resp (This structure contains the response information for each query)	**6**
wsa_socket (A structure containing the socket parameters used for creating TCP/IP connection for control and data acquisition)	**6**
wsa_time (This structure contains the time information. It is used for the time stamp in a frame header)	**6**

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 uint64_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.

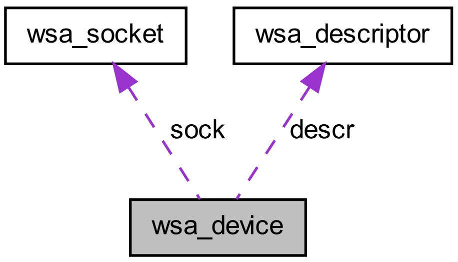
4.1.1.15 char rfe_version

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.

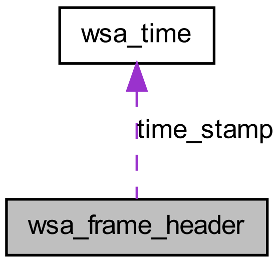
4.2.1.2 struct wsa_socket sock

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

- char prod_serial [20]
- uint64_t freq
- char gain [10]
- uint32_t sample_size
- struct wsa_time time_stamp

4.3.1 Field Documentation

4.3.1.1 uint64_t **freq**

The center frequency (Hz) to which the RF PLL is tuned.

4.3.1.2 char **gain**

The amplification in the radio front end at the time a WSA data frame is captured.

4.3.1.3 char **prod_serial**

WSA product version number.

4.3.1.4 uint32_t **sample_size**

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

4.3.1.5 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).

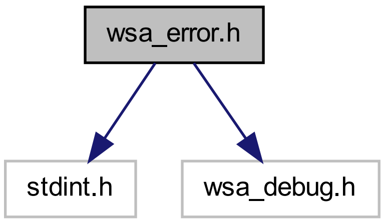
4.6.1.2 int32_t sec

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

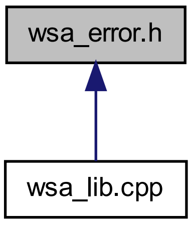
5 File Documentation

5.1 wsa_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_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_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)

5.1.1 Define Documentation

5.1.1.1 #define LNEG_NUM (-10000)

5.1.1.2 #define WSA_ERR_ANTENNASETFAILED (LNEG_NUM - 1602)

5.1.1.3 #define WSA_ERR_CALIBRATESETFAILED (LNEG_NUM - 1606)

5.1.1.4 #define WSA_ERR_CMDINVALID (LNEG_NUM - 1502)

5.1.1.5 #define WSA_ERR_CMDSENDFAILED (LNEG_NUM - 1501)

5.1.1.6 #define WSA_ERR_ETHERNETCONNECTFAILED (LNEG_NUM - 207)

5.1.1.7 #define WSA_ERR_ETHERNETINITFAILED (LNEG_NUM - 209)

5.1.1.8 #define WSA_ERR_ETHERNETNOTAVBL (LNEG_NUM - 206)

5.1.1.9 #define WSA_ERR_FILECREATEFAILED (LNEG_NUM - 1900)

5.1.1.10 #define WSA_ERR_FILEOPENFAILED (LNEG_NUM - 1901)

5.1.1.11 #define WSA_ERR_FILEREADFAILED (LNEG_NUM - 1902)

5.1.1.12 #define WSA_ERR_FILEWRITEFAILED (LNEG_NUM - 1903)

5.1.1.13 #define WSA_ERR_FILTERSETFAILED (LNEG_NUM - 1604)

5.1.1.14 #define WSA_ERR_FREQOUTOFBOUND (LNEG_NUM - 601)

5.1.1.15 #define WSA_ERR_FREQSETFAILED (LNEG_NUM - 603)

5.1.1.16 #define WSA_ERR_IFGAINSETFAILED (LNEG_NUM - 803)

5.1.1.17 #define WSA_ERR_INITFAILED (LNEG_NUM - 105)

5.1.1.18 #define WSA_ERR_INVADCCORRVALUE (LNEG_NUM - 106)

5.1.1.19 #define WSA_ERR_INVAMP (LNEG_NUM - 301)

5.1.1.20 #define WSA_ERR_INVANTENNAPORT (LNEG_NUM - 1601)

5.1.1.21 #define WSA_ERR_INVCALIBRATEMODE (LNEG_NUM - 1605)

5.1.1.22 #define WSA_ERR_INVDWELL (LNEG_NUM - 1208)

5.1.1.23 #define WSA_ERR_INVFILTERMODE (LNEG_NUM - 1603)

5.1.1.24 #define WSA_ERR_INVFREQRES (LNEG_NUM - 602)

5.1.1.25 #define WSA_ERR_INVIFGAIN (LNEG_NUM - 802)

5.1.1.26 #define WSA_ERR_INVINTFMETHOD (LNEG_NUM - 201)

5.1.1.27 #define WSA_ERR_INVIPADDRESS (LNEG_NUM - 2)

5.1.1.28 #define WSA_ERR_INVIPHOSTADDRESS (LNEG_NUM - 202)

5.1.1.29 #define WSA_ERR_INVNUMBER (LNEG_NUM - 2000)

5.1.1.30 #define WSA_ERR_INVNUMFRAMES (LNEG_NUM - 1209)

5.1.1.31 #define WSA_ERR_INVREGADDR (LNEG_NUM - 2001)

5.1.1.32 #define WSA_ERR_INVRFGAIN (LNEG_NUM - 801)

5.1.1.33 #define WSA_ERR_INVRUNMODE (LNEG_NUM - 1001)

5.1.1.34 #define WSA_ERR_INVSAMPLESIZE (LNEG_NUM - 403)

5.1.1.35 #define WSA_ERR_INVSTARTRES (LNEG_NUM - 1205)

5.1.1.36 #define WSA_ERR_INVSTOPFREQ (LNEG_NUM - 1202)

5.1.1.37 #define WSA_ERR_INVSTOPRES (LNEG_NUM - 1206)

5.1.1.38 #define WSA_ERR_INVTRIGID (LNEG_NUM - 1201)

5.1.1.39 #define WSA_ERR_INVTRIGRANGE (LNEG_NUM - 1207)

5.1.1.40 #define WSA_ERR_MALLOCFAILED (LNEG_NUM - 2002)

5.1.1.41 #define WSA_ERR_NOCTRLPIPE (LNEG_NUM - 3)

5.1.1.42 #define WSA_ERR_NODATABUS (LNEG_NUM - 401)

5.1.1.43 #define WSA_ERR_NOWSA (LNEG_NUM - 1)

5.1.1.44 #define WSA_ERR_OPENFAILED (LNEG_NUM - 104)

5.1.1.45 #define WSA_ERR_PLLLOCKFAILED (LNEG_NUM - 604)

5.1.1.46 #define WSA_ERR_PRODOBSOLETE (LNEG_NUM - 8)

5.1.1.47 #define WSA_ERR_READFRAMEFAILED (LNEG_NUM - 402)

5.1.1.48 #define WSA_ERR_RFGAINSETFAILED (LNEG_NUM - 804)

5.1.1.49 #define WSA_ERR_SETFAILED (LNEG_NUM - 103)

5.1.1.50 #define WSA_ERR_SOCKETSETFUPFAILED (LNEG_NUM - 211)

5.1.1.51 #define WSA_ERR_STARTOOB (LNEG_NUM - 1203)

5.1.1.52 #define WSA_ERR_STOPOOB (LNEG_NUM - 1204)

5.1.1.53 #define WSA_ERR_UNKNOWN_ERROR (LNEG_NUM - 2003)

5.1.1.54 #define WSA_ERR_UNKNOWNFWRVSN (LNEG_NUM - 6)

5.1.1.55 #define WSA_ERR_UNKNOWNPRODSER (LNEG_NUM - 4)

5.1.1.56 #define WSA_ERR_UNKNOWNPRODVSN (LNEG_NUM - 5)

5.1.1.57 #define WSA_ERR_UNKNOWNRFEVSN (LNEG_NUM - 7)

5.1.1.58 #define WSA_ERR_USBINITFAILED (LNEG_NUM - 205)

5.1.1.59 #define WSA_ERR_USBNOTAVBL (LNEG_NUM - 203)

5.1.1.60 #define WSA_ERR_USBOPENFAILED (LNEG_NUM - 204)

5.1.1.61 #define WSA_ERR_WINSOCKSTARTUPFAILED (LNEG_NUM - 210)

5.1.1.62 #define WSA_ERR_WSAINUSE (LNEG_NUM - 102)

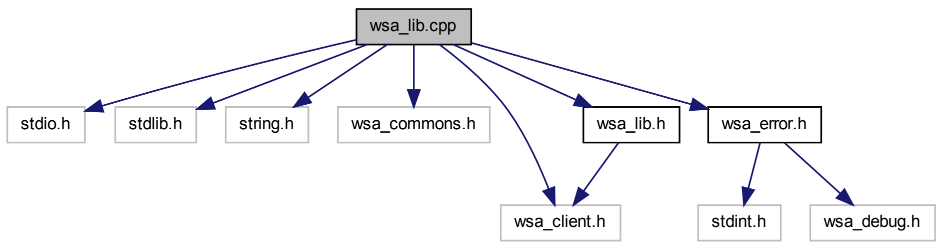
5.1.1.63 #define WSA_ERR_WSANOTRDY (LNEG_NUM - 101)

5.1.2 Function Documentation

5.1.2.1 const char* wsa_get_err_msg (int16_t *err_id*)

5.2 wsa_lib.cpp File Reference

Include dependency graph for wsa_lib.cpp:



Defines

- #define MAX_FILE_LINES 300
- #define SEP_CHARS "\n\r"

Functions

- int16_t wsa_tokenize_file (FILE ∗fptr, char ∗cmd_str[ ])
- int16_t wsa_dev_init (struct wsa_device ∗dev)
- int16_t wsa_connect (struct wsa_device ∗dev, char ∗cmd_syntax, char ∗intf_method)
- int16_t wsa_disconnect (struct wsa_device ∗dev)
- int16_t wsa_list_devs (char ∗∗wsa_list)
- int16_t wsa_send_command (struct wsa_device ∗dev, char ∗command)
- int16_t wsa_send_command_file (struct wsa_device ∗dev, char ∗file_name)
- struct wsa_resp wsa_send_query (struct wsa_device ∗dev, char ∗command)
- int16_t wsa_query_error (struct wsa_device ∗dev)
- int64_t wsa_get_frame (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int32_t ∗i_buf, int32_t ∗q_buf, uint64_t sample_size)

5.2.1 Define Documentation

5.2.1.1 #define MAX_FILE_LINES 300

5.2.1.2 #define SEP_CHARS "\n\r"

5.2.2 Function Documentation

5.2.2.1 int16_t wsa_connect (struct wsa_device ∗ *dev*, char ∗ *cmd_syntax*, char ∗ *intf_method*)

Connect to a WSA through the specified interface method **intf_method**, and communicate control commands in the format of the given command syntax.

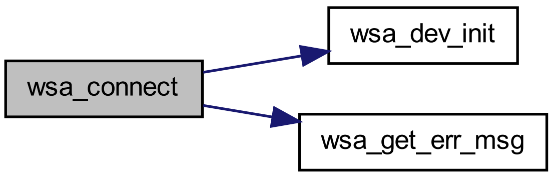
Parameters

dev	- A pointer to the WSA device structure to be connected/establised.
cmd_syntax	- A char pointer to store standard for control commands communication to the WSA. Currently supported standard command syntax type is: SCPI.
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. TODO: define ERROR values with associated messages....

Here is the call graph for this function:



5.2.2.2 int16_t wsa_dev_init (struct **wsa_device** ∗ *dev*)

Initialized the the wsa_device structure

Parameters

dev	- A pointer to the WSA device structure.

Returns

None

5.2.2.3 int16_t wsa_disconnect (struct **wsa_device** ∗ *dev*)

Close the device connection if one is started, stop any existing data capture, and perform any necessary clean ups.

Parameters

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

Returns

0 on success, or a negative number on error.

5.2.2.4 int64_t wsa_get_frame (struct **wsa_device** ∗ *dev*, struct **wsa_frame_header** ∗ *header*, int32_t ∗ *i_buf*, int32_t ∗ *q_buf*, uint64_t *sample_size*)

Reads a frame of data. *Each* frame consists of a header, and I and Q buffers of data of length determine by the **sample_size** parameter.

Parameters

dev	- A pointer to the WSA device structure.

header	- A pointer to **wsa_frame_header** structure to store information for the frame.
i_buf	- A 16-bit signed integer pointer for the unscaled, I data buffer with size specified by the sample_size.
q_buf	- A 16-bit signed integer pointer for the unscaled Q data buffer with size specified by the sample_size.
sample_size	- A 64-bit unsigned integer sample size (i.e. {I, Q} sample pairs) per data frame to be captured. The frame size is limited to a maximum number, **max_sample_size**, listed in the **wsa_descriptor** structure.

Returns

Number of samples read on success, or a negative number on error.

5.2.2.5 int16_t wsa_list_devs (char ** *wsa_list*)

List (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.

5.2.2.6 int16_t wsa_query_error (struct wsa_device * *dev*)

Querry the WSA for any error.

Parameters

dev	- A pointer to the WSA device structure.

Returns

0 on success, or a negative number on error.

5.2.2.7 int16_t wsa_send_command (struct wsa_device * *dev*, char * *command*)

Open a file or print the help commands information associated with the WSA used.

Parameters

dev	- The WSA device structure from which the help information will be provided.

Returns

0 on success, or a negative number on error. Send the control command string to the WSA device specified by **dev**. The commands format must be written according to the specified standard syntax in wsa_connect().

Parameters

dev	- A pointer to the WSA device structure.
command	- A char pointer to the control command string written in the format specified by the syntax standard in wsa_connect()

Returns

Number of bytes sent on success, or a negative number on error.

5.2.2.8 int16_t wsa_send_command_file (struct **wsa_device** * *dev*, char * *file_name*)

Read command line(s) stored in the given **file_name** and send 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.

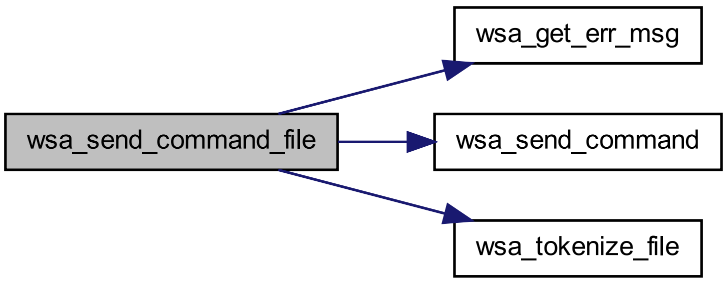
Parameters

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.

Here is the call graph for this function:



5.2.2.9 struct wsa_resp wsa_send_query (struct wsa_device * *dev*, char * *command*) [read]

Send query command to the WSA device specified by **dev**. The commands format must be written according to the specified command syntax in wsa_connect().

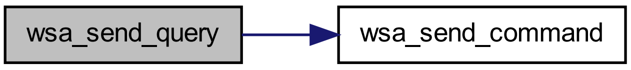
Parameters

dev	- A pointer to the WSA device structure.
command	- A char pointer to the query command string written in the format specified by the command syntax in wsa_connect().

Returns

The result stored in a wsa_resp struct format.

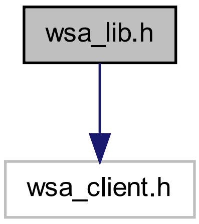
Here is the call graph for this function:



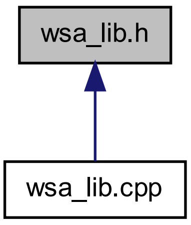
5.2.2.10 int16_t wsa_tokenize_file (FILE * *fptr*, char * *cmd_str[]*)

5.3 wsa_lib.h File Reference

Include dependency graph for wsa_lib.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.

Defines

- #define FALSE 0
- #define TRUE 1
- #define NUM_RF_GAINS 5
- #define SCPI "SCPI"

Enumerations

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

Functions

- int16_t wsa_connect (struct wsa_device ∗dev, char ∗cmd_syntax, char ∗intf_-method)
- int16_t wsa_disconnect (struct wsa_device ∗dev)
- int16_t wsa_list_devs (char ∗∗wsa_list)
- int16_t wsa_send_command (struct wsa_device ∗dev, char ∗command)
- int16_t wsa_send_command_file (struct wsa_device ∗dev, char ∗file_name)
- struct wsa_resp wsa_send_query (struct wsa_device ∗dev, char ∗command)
- int16_t wsa_query_error (struct wsa_device ∗dev)
- int64_t wsa_get_frame (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int32_t ∗i_buf, int32_t ∗q_buf, uint64_t sample_size)

5.3.1 Define Documentation

5.3.1.1 #define FALSE 0

5.3.1.2 #define NUM_RF_GAINS 5

5.3.1.3 #define SCPI "SCPI"

5.3.1.4 #define TRUE 1

5.3.2 Enumeration Type Documentation

5.3.2.1 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.

5.3.3 Function Documentation

5.3.3.1 int16_t wsa_connect (struct wsa_device ∗ *dev*, char ∗ *cmd_syntax*, char ∗ *intf_method*)

Connect to a WSA through the specified interface method **intf_method**, and communicate control commands in the format of the given command syntax.

Parameters

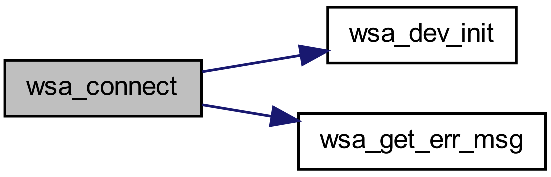
dev	- A pointer to the WSA device structure to be connected/establised.

cmd_syntax	- A char pointer to store standard for control commands communication to the WSA. Currently supported standard command syntax type is: SCPI.
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. TODO: define ERROR values with associated messages....

Here is the call graph for this function:



5.3.3.2 int16_t wsa_disconnect (struct wsa_device * *dev*)

Close the device connection if one is started, stop any existing data capture, and perform any necessary clean ups.

Parameters

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

Returns

0 on success, or a negative number on error.

5.3.3.3 int64_t wsa_get_frame (struct wsa_device * *dev*, struct wsa_frame_header * *header*, int32_t * *i_buf*, int32_t * *q_buf*, uint64_t *sample_size*)

Reads a frame of data. *Each* frame consists of a header, and I and Q buffers of data of length determine by the **sample_size** parameter.

Parameters

dev	- A pointer to the WSA device structure.
header	- A pointer to **wsa_frame_header** structure to store information for the frame.
i_buf	- A 16-bit signed integer pointer for the unscaled, I data buffer with size specified by the sample_size.
q_buf	- A 16-bit signed integer pointer for the unscaled Q data buffer with size specified by the sample_size.
sample_size	- A 64-bit unsigned integer sample size (i.e. {I, Q} sample pairs) per data frame to be captured. The frame size is limited to a maximum number, **max_sample_size**, listed in the **wsa_descriptor** structure.

Returns

Number of samples read on success, or a negative number on error.

5.3.3.4 int16_t wsa_list_devs (char ∗∗ *wsa_list*)

List (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.

5.3.3.5 int16_t wsa_query_error (struct wsa_device ∗ *dev*)

Querry the WSA for any error.

Parameters

dev	- A pointer to the WSA device structure.

Returns

0 on success, or a negative number on error.

5.3.3.6 int16_t wsa_send_command (struct wsa_device ∗ *dev*, char ∗ *command*)

Open a file or print the help commands information associated with the WSA used.

Parameters

dev	- The WSA device structure from which the help information will be provided.

Returns

0 on success, or a negative number on error. Send the control command string to the WSA device specified by **dev**. The commands format must be written according to the specified standard syntax in wsa_connect().

Parameters

dev	- A pointer to the WSA device structure.
command	- A char pointer to the control command string written in the format specified by the syntax standard in wsa_connect()

Returns

Number of bytes sent on success, or a negative number on error.

5.3.3.7 int16_t wsa_send_command_file (struct wsa_device ∗ *dev*, char ∗ *file_name*)

Read command line(s) stored in the given **file_name** and send 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.

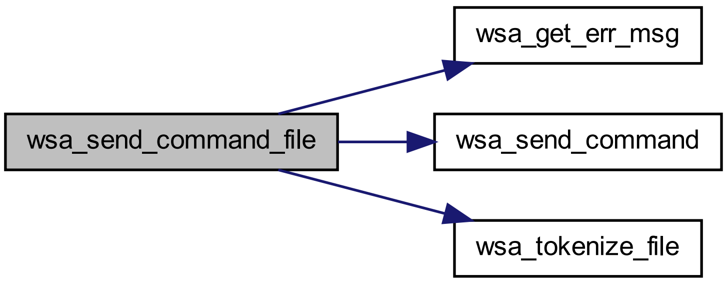
Parameters

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.

Here is the call graph for this function:



5.3.3.8 struct wsa_resp wsa_send_query (struct wsa_device * *dev*, char * *command*) [read]

Send query command to the WSA device specified by **dev**. The commands format must be written according to the specified command syntax in wsa_connect().

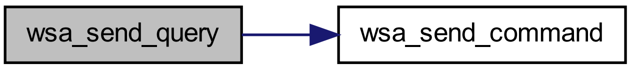
Parameters

dev	- A pointer to the WSA device structure.
command	- A char pointer to the query command string written in the format specified by the command syntax in wsa_connect().

Returns

The result stored in a wsa_resp struct format.

Here is the call graph for this function:



5.4 wsa_lib.txt File Reference

Contain some code documents for wsa_lib.h.

5.4.1 Detailed Description

Index