

Standard WSA Library

Generated by Doxygen 1.7.4

Thu Sep 8 2011 16:03:56

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	How to use the library . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>1</b>
2.1	Data Structures . . . . .	1
<b>3</b>	<b>File Index</b>	<b>1</b>
3.1	File List . . . . .	2
<b>4</b>	<b>Data Structure Documentation</b>	<b>2</b>
4.1	wsa_descriptor Struct Reference . . . . .	2
4.1.1	Field Documentation . . . . .	2
4.2	wsa_device Struct Reference . . . . .	4
4.2.1	Field Documentation . . . . .	4
4.3	wsa_frame_header Struct Reference . . . . .	4
4.3.1	Field Documentation . . . . .	5
4.4	wsa_resp Struct Reference . . . . .	6
4.4.1	Field Documentation . . . . .	6
4.5	wsa_socket Struct Reference . . . . .	6
4.5.1	Field Documentation . . . . .	6
4.6	wsa_time Struct Reference . . . . .	6
4.6.1	Field Documentation . . . . .	7
<b>5</b>	<b>File Documentation</b>	<b>7</b>
5.1	ReadMe.txt File Reference . . . . .	7
5.1.1	Variable Documentation . . . . .	7
5.2	test scpi.txt File Reference . . . . .	7
5.3	wsa_error.h File Reference . . . . .	8
5.3.1	Define Documentation . . . . .	10
5.3.2	Function Documentation . . . . .	12
5.4	wsa_lib.cpp File Reference . . . . .	13
5.4.1	Define Documentation . . . . .	13
5.4.2	Function Documentation . . . . .	13
5.5	wsa_lib.h File Reference . . . . .	18

<b>1 Introduction</b>	<b>1</b>
-----------------------	----------

5.5.1 Define Documentation . . . . .	19
5.5.2 Enumeration Type Documentation . . . . .	19
5.5.3 Function Documentation . . . . .	20
5.6 wsa_lib.txt File Reference . . . . .	24
5.6.1 Detailed Description . . . . .	24

## 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:

<a href="#">wsa_descriptor</a> (This structure stores WSA information )	2
<a href="#">wsa_device</a> (A structure containing the components associate with each WSA device )	4
<a href="#">wsa_frame_header</a> (This structure contains header information related to each frame read by <a href="#">wsa_get_frame()</a> )	4
<a href="#">wsa_resp</a> (This structure contains the response information for each query )	6
<a href="#">wsa_socket</a> (A structure containing the socket parameters used for creating TCP/IP connection for control and data acquisition )	6
<a href="#">wsa_time</a> (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:

<a href="#">wsa_error.h</a>	8
<a href="#">wsa_lib.cpp</a>	13
<a href="#">wsa_lib.h</a>	18

## 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.

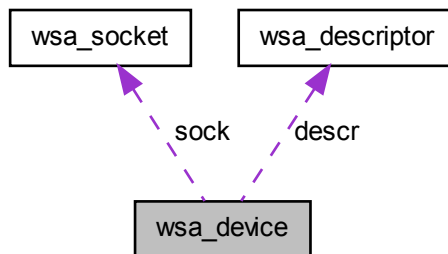
4.1.1.15 **char rfe\_version**

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](#).

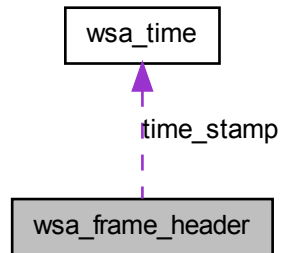
4.2.1.2 **struct `wsa_socket` `sock`**

The socket structure component of the WSA, used for TCP/IP 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 ReadMe.txt File Reference

#### Variables

- and information about the [platforms](#)
- and information about the [configurations](#)
- and information about the and project features selected with the Application Wizard `wsa4000_cli.cpp` This is the main application source file Other standard [files](#)

#### 5.1.1 Variable Documentation

##### 5.1.1.1 and information about the [configurations](#)

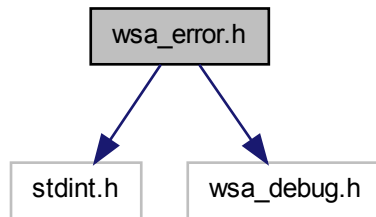
##### 5.1.1.2 and information about the and project features selected with the Application Wizard `wsa4000_cli.cpp` This is the main application source file Other standard [files](#)

##### 5.1.1.3 and information about the [platforms](#)

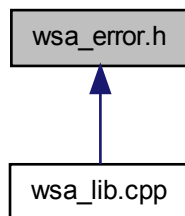
### 5.2 test scpi.txt File Reference

### 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_UNKNOWNPRODSE` (`LNEG_NUM` - 4)
- #define `WSA_ERR_UNKNOWNPRODSN` (`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`)

#### 5.3.1 Define Documentation

- 5.3.1.1 `#define LNEG_NUM` (-10000)
- 5.3.1.2 `#define WSA_ERR_ANNASETFAILED` (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\_INVANTENNAPOST (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\_MALLOCF FAILED (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\_NOWSA (LNEG\_NUM - 1)
- 5.3.1.44 #define WSA\_ERR\_OPENFAILED (LNEG\_NUM - 104)

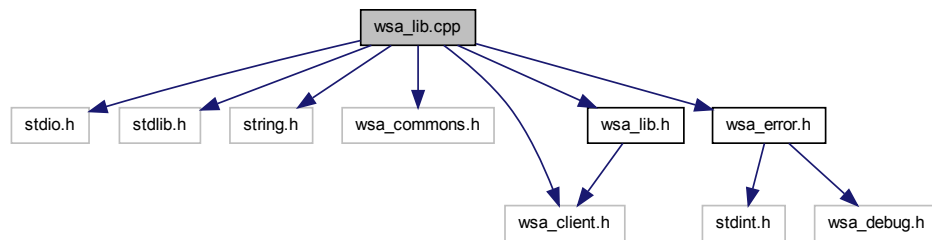
- 5.3.1.45 `#define WSA_ERR_PLLOCKFAILED (LNEG_NUM - 604)`
- 5.3.1.46 `#define WSA_ERR_PRODOBSOLETE (LNEG_NUM - 8)`
- 5.3.1.47 `#define WSA_ERR_READFRAMEFAILED (LNEG_NUM - 402)`
- 5.3.1.48 `#define WSA_ERR_RFGAINSETFAILED (LNEG_NUM - 804)`
- 5.3.1.49 `#define WSA_ERR_SETFAILED (LNEG_NUM - 103)`
- 5.3.1.50 `#define WSA_ERR_SOCKETSETFUPFAILED (LNEG_NUM - 211)`
- 5.3.1.51 `#define WSA_ERR_STARTOOB (LNEG_NUM - 1203)`
- 5.3.1.52 `#define WSA_ERR_STOPOOB (LNEG_NUM - 1204)`
- 5.3.1.53 `#define WSA_ERR_UNKNOWN_ERROR (LNEG_NUM - 2003)`
- 5.3.1.54 `#define WSA_ERR_UNKNOWNFWRVSN (LNEG_NUM - 6)`
- 5.3.1.55 `#define WSA_ERR_UNKNOWNPRODSE (LNEG_NUM - 4)`
- 5.3.1.56 `#define WSA_ERR_UNKNOWNPRODVSN (LNEG_NUM - 5)`
- 5.3.1.57 `#define WSA_ERR_UNKNOWNRFEVSN (LNEG_NUM - 7)`
- 5.3.1.58 `#define WSA_ERR_USBINITFAILED (LNEG_NUM - 205)`
- 5.3.1.59 `#define WSA_ERR_USBNOTAVBL (LNEG_NUM - 203)`
- 5.3.1.60 `#define WSA_ERR_USBOPENFAILED (LNEG_NUM - 204)`
- 5.3.1.61 `#define WSA_ERR_WINSOCKSTARTUPFAILED (LNEG_NUM - 210)`
- 5.3.1.62 `#define WSA_ERR_WSAINUSE (LNEG_NUM - 102)`
- 5.3.1.63 `#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.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)`

#### 5.4.1 Define Documentation

##### 5.4.1.1 `#define MAX_FILE_LINES 300`

##### 5.4.1.2 `#define SEP_CHARS "\n\r"`

#### 5.4.2 Function Documentation

5.4.2.1 `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.

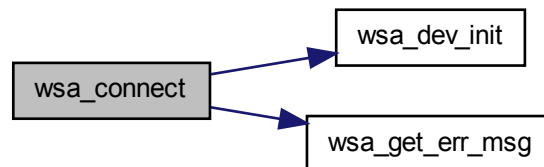
#### Parameters

<i>dev</i>	- A pointer to the WSA device structure to be connected/established.
<i>cmd_syntax</i>	- A char pointer to store standard for control commands communication to the WSA. Currently supported standard command syntax type is: SCPI.
<i>intf_method</i>	- A char pointer to store the interface method to the WSA. Possible methods: <ul style="list-style-type: none"> <li>• With LAN, use: "TCPIP::<ip address="" of="" the="" wsa="">::HISLIP"</ip></li> <li>• With USB, use: "USB" (check if supported with the WSA version used)</li> </ul>

#### 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:



5.4.2.2 `int16_t wsa_dev_init ( struct wsa_device * dev )`

Initialized the the [wsa\\_device](#) structure

#### Parameters

<i>dev</i>	- A pointer to the WSA device structure.
------------	--

#### Returns

None



#### 5.4.2.3 `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

<i>dev</i>	- A pointer to the WSA device structure to be closed.
------------	---

##### Returns

0 on success, or a negative number on error.

#### 5.4.2.4 `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

<i>dev</i>	- A pointer to the WSA device structure.
<i>header</i>	- A pointer to <a href="#">wsa_frame_header</a> structure to store information for the frame.
<i>i_buf</i>	- A 16-bit signed integer pointer for the unscaled, I data buffer with size specified by the <i>sample_size</i> .
<i>q_buf</i>	- A 16-bit signed integer pointer for the unscaled Q data buffer with size specified by the <i>sample_size</i> .
<i>sample_size</i>	- 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, <b>max_sample_size</b> , listed in the <a href="#">wsa_descriptor</a> structure.

##### Returns

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

#### 5.4.2.5 `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

<i>wsa_list</i>	- 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.

#### 5.4.2.6 `int16_t wsa_query_error ( struct wsa_device * dev )`

Query the WSA for any error.

##### Parameters

<i>dev</i> - A pointer to the WSA device structure.
---

##### Returns

0 on success, or a negative number on error.

#### 5.4.2.7 `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

<i>dev</i> - 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

<i>dev</i> - A pointer to the WSA device structure.
---

<i>command</i> - A char pointer to the control command string written in the format specified by the syntax standard in <a href="#">wsa_connect()</a>
---

##### Returns

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

#### 5.4.2.8 `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

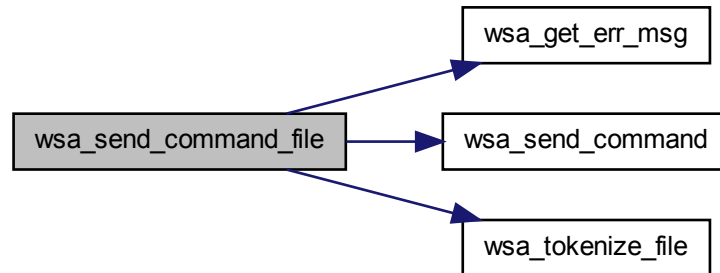
<i>dev</i> - A pointer to the WSA device structure.
---

<i>file_name</i> - 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:



**5.4.2.9** `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

<i>dev</i>	- A pointer to the WSA device structure.
<i>command</i>	- A char pointer to the query command string written in the format specified by the command syntax in <a href="#">wsa_connect()</a> .

#### Returns

The result stored in a [wsa\\_resp](#) struct format.

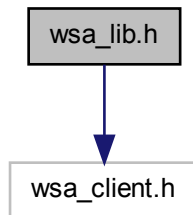
Here is the call graph for this function:



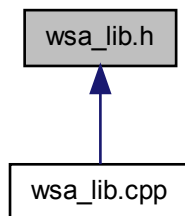
**5.4.2.10** `int16_t wsa_tokenize_file ( FILE * fptr, char * cmd_str[] )`

## 5.5 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)

#### 5.5.1 Define Documentation

##### 5.5.1.1 #define FALSE 0

##### 5.5.1.2 #define NUM\_RF\_GAINS 5

##### 5.5.1.3 #define SCPI "SCPI"

##### 5.5.1.4 #define TRUE 1

#### 5.5.2 Enumeration Type Documentation

## 5.5.2.1 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.

## 5.5.3 Function Documentation

## 5.5.3.1 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.

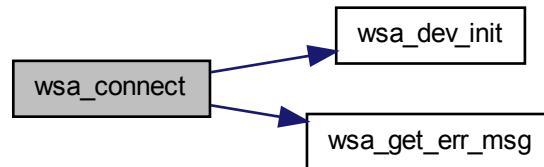
**Parameters**

<i>dev</i>	- A pointer to the WSA device structure to be connected/established.
<i>cmd_syntax</i>	- A char pointer to store standard for control commands communication to the WSA. Currently supported standard command syntax type is: SCPI.
<i>intf_method</i>	- A char pointer to store the interface method to the WSA. Possible methods: <ul style="list-style-type: none"> <li>• With LAN, use: "TCPIP::<ip address="" of="" the="" wsa="">::HISLIP"</ip></li> <li>• With USB, use: "USB" (check if supported with the WSA version used)</li> </ul>

**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:



#### 5.5.3.2 `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

<code>dev</code>	- A pointer to the WSA device structure to be closed.
------------------	---

##### Returns

0 on success, or a negative number on error.

#### 5.5.3.3 `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

<code>dev</code>	- A pointer to the WSA device structure.
<code>header</code>	- A pointer to <a href="#">wsa_frame_header</a> structure to store information for the frame.
<code>i_buf</code>	- A 16-bit signed integer pointer for the unscaled, I data buffer with size specified by the <code>sample_size</code> .
<code>q_buf</code>	- A 16-bit signed integer pointer for the unscaled Q data buffer with size specified by the <code>sample_size</code> .
<code>sample_size</code>	- 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, <b>max_sample_size</b> , listed in the <a href="#">wsa_descriptor</a> structure.

**Returns**

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

**5.5.3.4 `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**

<code>wsa_list</code>	- 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.

**5.5.3.5 `int16_t wsa_query_error ( struct wsa_device * dev )`**

Query the WSA for any error.

**Parameters**

<code>dev</code>	- A pointer to the WSA device structure.
------------------	--

**Returns**

0 on success, or a negative number on error.

**5.5.3.6 `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**

<code>dev</code>	- 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**

<code>dev</code>	- A pointer to the WSA device structure.
<code>command</code>	- A char pointer to the control command string written in the format specified by the syntax standard in <a href="#">wsa_connect()</a>



**Returns**

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

5.5.3.7 `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.

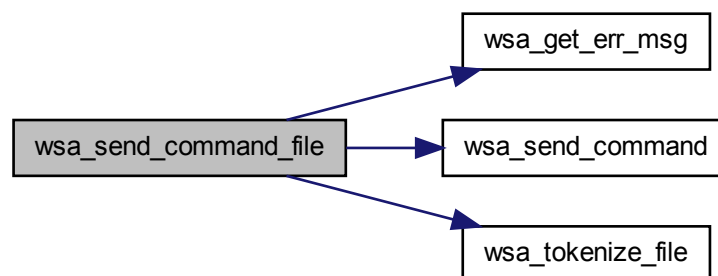
**Parameters**

<i>dev</i>	- A pointer to the WSA device structure.
<i>file_name</i>	- 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:



5.5.3.8 `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**

<i>dev</i>	- A pointer to the WSA device structure.
<i>command</i>	- A char pointer to the query command string written in the format specified by the command syntax in <a href="#">wsa_connect()</a> .

### Returns

The result stored in a [wsa\\_resp](#) struct format.

Here is the call graph for this function:



## 5.6 wsa\_lib.txt File Reference

Contain some code documents for [wsa\\_lib.h](#).

### 5.6.1 Detailed Description

## Index

abs\_max\_amp  
    wsa\_descriptor, 2

cmd  
    wsa\_socket, 6

configurations  
    ReadMe.txt, 7

data  
    wsa\_socket, 6

descr  
    wsa\_device, 4

FALSE  
    wsa\_lib.h, 19

files  
    ReadMe.txt, 7

freq  
    wsa\_frame\_header, 5

freq\_resolution  
    wsa\_descriptor, 2

fw\_version  
    wsa\_descriptor, 2

gain  
    wsa\_frame\_header, 5

inst\_bw  
    wsa\_descriptor, 3

intf\_type  
    wsa\_descriptor, 3

LNEG\_NUM  
    wsa\_error.h, 10

MAX\_FILE\_LINES  
    wsa\_lib.cpp, 13

max\_if\_gain  
    wsa\_descriptor, 3

max\_sample\_size  
    wsa\_descriptor, 3

max\_tune\_freq  
    wsa\_descriptor, 3

min\_if\_gain  
    wsa\_descriptor, 3

min\_tune\_freq  
    wsa\_descriptor, 3

nsec  
    wsa\_time, 7

NUM\_RF\_GAINS  
    wsa\_lib.h, 19

platforms  
    ReadMe.txt, 7

prod\_name  
    wsa\_descriptor, 3

prod\_serial  
    wsa\_descriptor, 3  
    wsa\_frame\_header, 5

prod\_version  
    wsa\_descriptor, 3

ReadMe.txt, 7  
    configurations, 7  
    files, 7  
    platforms, 7

result  
    wsa\_resp, 6

rfe\_name  
    wsa\_descriptor, 3

rfe\_version  
    wsa\_descriptor, 3

sample\_size  
    wsa\_frame\_header, 5

SCPI  
    wsa\_lib.h, 19

sec  
    wsa\_time, 7

SEP\_CHARS  
    wsa\_lib.cpp, 13

sock  
    wsa\_device, 4

status  
    wsa\_resp, 6

test scpi.txt, 7

time\_stamp  
    wsa\_frame\_header, 5

TRUE  
    wsa\_lib.h, 19

WSA\_GAIN\_HIGH  
    wsa\_lib.h, 20

WSA\_GAIN\_LOW

- wsa\_lib.h, [20](#)
- WSA\_GAIN\_MEDIUM
  - wsa\_lib.h, [20](#)
- WSA\_GAIN\_VLOW
  - wsa\_lib.h, [20](#)
- wsa\_lib.h
  - WSA\_GAIN\_HIGH, [20](#)
  - WSA\_GAIN\_LOW, [20](#)
  - WSA\_GAIN\_MEDIUM, [20](#)
  - WSA\_GAIN\_VLOW, [20](#)
- wsa\_connect
  - wsa\_lib.cpp, [13](#)
  - wsa\_lib.h, [20](#)
- wsa\_descriptor, [2](#)
  - abs\_max\_amp, [2](#)
  - freq\_resolution, [2](#)
  - fw\_version, [2](#)
  - inst\_bw, [3](#)
  - intf\_type, [3](#)
  - max\_if\_gain, [3](#)
  - max\_sample\_size, [3](#)
  - max\_tune\_freq, [3](#)
  - min\_if\_gain, [3](#)
  - min\_tune\_freq, [3](#)
  - prod\_name, [3](#)
  - prod\_serial, [3](#)
  - prod\_version, [3](#)
  - rfe\_name, [3](#)
  - rfe\_version, [3](#)
- wsa\_dev\_init
  - wsa\_lib.cpp, [14](#)
- wsa\_device, [4](#)
  - descr, [4](#)
  - sock, [4](#)
- wsa\_disconnect
  - wsa\_lib.cpp, [14](#)
  - wsa\_lib.h, [21](#)
- WSA\_ERR\_ANTENNASETFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_CALIBRATESETFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_CMDINVALID
  - wsa\_error.h, [10](#)
- WSA\_ERR\_CMDSENDFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_ETHERNETCONNECTFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_ETHERNETINITFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_ETHERNETNOTAVBL
  - wsa\_error.h, [10](#)
- wsa\_error.h, [10](#)
- WSA\_ERR\_FILECREATEFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FILEOPENFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FILEREADFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FILEWRITEFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FILTERSETFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FREQOUTOFBOUND
  - wsa\_error.h, [10](#)
- WSA\_ERR\_FREQSETFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_IFGAINSETFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_INITFAILED
  - wsa\_error.h, [10](#)
- WSA\_ERR\_INVADCCORRVALUE
  - wsa\_error.h, [10](#)
- WSA\_ERR\_INVAMP
  - wsa\_error.h, [10](#)
- WSA\_ERR\_INVANTENNAPORT
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVCALIBRATEMODE
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVDWELL
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVFILTERMODE
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVFREQRES
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVIFGAIN
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVINTFMETHOD
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVIPADDRESS
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVIPHOSTADDRESS
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVNUMBER
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVNUMFRAMES
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVREGADDR
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVRFGAIN
  - wsa\_error.h, [11](#)
- WSA\_ERR\_INVRUNMODE

wsa\_error.h, 11  
WSA\_ERR\_INVSAMPLESIZE  
wsa\_error.h, 11  
WSA\_ERR\_INVSTARTRES  
wsa\_error.h, 11  
WSA\_ERR\_INVSTOPFREQ  
wsa\_error.h, 11  
WSA\_ERR\_INVSTOPRES  
wsa\_error.h, 11  
WSA\_ERR\_INVTRIGID  
wsa\_error.h, 11  
WSA\_ERR\_INVTRIGRANGE  
wsa\_error.h, 11  
WSA\_ERR\_MALLOCFAILED  
wsa\_error.h, 11  
WSA\_ERR\_NOCTRLPIPE  
wsa\_error.h, 11  
WSA\_ERR\_NODATABUS  
wsa\_error.h, 11  
WSA\_ERR\_NOWSA  
wsa\_error.h, 11  
WSA\_ERR\_OPENFAILED  
wsa\_error.h, 11  
WSA\_ERR\_PLLOCKFAILED  
wsa\_error.h, 11  
WSA\_ERR\_PRODOBSOLETE  
wsa\_error.h, 12  
WSA\_ERR\_READFRAMEFAILED  
wsa\_error.h, 12  
WSA\_ERR\_RFGAINSETFAILED  
wsa\_error.h, 12  
WSA\_ERR\_SETFAILED  
wsa\_error.h, 12  
WSA\_ERR\_SOCKETSETFUPFAILED  
wsa\_error.h, 12  
WSA\_ERR\_STARTOOB  
wsa\_error.h, 12  
WSA\_ERR\_STOPOOB  
wsa\_error.h, 12  
WSA\_ERR\_UNKNOWN\_ERROR  
wsa\_error.h, 12  
WSA\_ERR\_UNKNOWNFWRVSN  
wsa\_error.h, 12  
WSA\_ERR\_UNKNOWNPRODSE  
wsa\_error.h, 12  
WSA\_ERR\_UNKNOWNPRODVSN  
wsa\_error.h, 12  
WSA\_ERR\_UNKNOWNRFEVSN  
wsa\_error.h, 12  
WSA\_ERR\_USBINITFAILED  
wsa\_error.h, 12  
WSA\_ERR\_USBNOTAVBL  
wsa\_error.h, 12  
WSA\_ERR\_USBOPENFAILED  
wsa\_error.h, 12  
WSA\_ERR\_WINSOCKSTARTUPFAILED  
wsa\_error.h, 12  
WSA\_ERR\_WSAINUSE  
wsa\_error.h, 12  
WSA\_ERR\_WSANOTRDY  
wsa\_error.h, 12  
wsa\_error.h, 8  
LNEG\_NUM, 10  
WSA\_ERR\_ANNENASETFAILED, 10  
WSA\_ERR\_CALIBRATESETFAILED,  
10  
WSA\_ERR\_CMDINVALID, 10  
WSA\_ERR\_CMDSENDFAILED, 10  
WSA\_ERR\_ETHERNETCONNECTFAILED,  
10  
WSA\_ERR\_ETHERNETINITFAILED,  
10  
WSA\_ERR\_ETHERNETNOTAVBL, 10  
WSA\_ERR\_FILECREATEFAILED, 10  
WSA\_ERR\_FILEOPENFAILED, 10  
WSA\_ERR\_FILEREADFAILED, 10  
WSA\_ERR\_FILEWRITEFAILED, 10  
WSA\_ERR\_FILTERSETFAILED, 10  
WSA\_ERR\_FREQOUTOFBOUND, 10  
WSA\_ERR\_FREQSETFAILED, 10  
WSA\_ERR\_IFGAINSETFAILED, 10  
WSA\_ERR\_INITFAILED, 10  
WSA\_ERR\_INVADCCORRVALUE, 10  
WSA\_ERR\_INVAMP, 10  
WSA\_ERR\_INVANTENNAPORT, 11  
WSA\_ERR\_INVCALIBRATEMODE, 11  
WSA\_ERR\_INVDWELL, 11  
WSA\_ERR\_INVFILTERMODE, 11  
WSA\_ERR\_INVFREQRES, 11  
WSA\_ERR\_INVIFGAIN, 11  
WSA\_ERR\_INVINTFMETHOD, 11  
WSA\_ERR\_INVIPADDRESS, 11  
WSA\_ERR\_INVIPHOSTADDRESS, 11  
WSA\_ERR\_INVNUMBER, 11  
WSA\_ERR\_INVNUMFRAMES, 11  
WSA\_ERR\_INVREGADDR, 11  
WSA\_ERR\_INVRFGAIN, 11  
WSA\_ERR\_INVRUNMODE, 11  
WSA\_ERR\_INVSAMPLESIZE, 11  
WSA\_ERR\_INVSTARTRES, 11

- WSA\_ERR\_INVSTOPFREQ, 11
- WSA\_ERR\_INVSTOPRES, 11
- WSA\_ERR\_INVTRIGID, 11
- WSA\_ERR\_INVTRIGRANGE, 11
- WSA\_ERR\_MALLOCF FAILED, 11
- WSA\_ERR\_NOCTRLPIPE, 11
- WSA\_ERR\_NODATABUS, 11
- WSA\_ERR\_NOWSA, 11
- WSA\_ERR\_OPENFAILED, 11
- WSA\_ERR\_PLLOCKFAILED, 11
- WSA\_ERR\_PRODOBSOLETE, 12
- WSA\_ERR\_READFRAMEFAILED, 12
- WSA\_ERR\_RFGAINSETFAILED, 12
- WSA\_ERR\_SETFAILED, 12
- WSA\_ERR\_SOCKETSETFUPFAILED, 12
- WSA\_ERR\_STARTOOB, 12
- WSA\_ERR\_STOPOOB, 12
- WSA\_ERR\_UNKNOWN\_ERROR, 12
- WSA\_ERR\_UNKNOWNFWRVSN, 12
- WSA\_ERR\_UNKNOWNPRODSE, 12
- WSA\_ERR\_UNKNOWNPRODVS, 12
- WSA\_ERR\_UNKNOWNRFEVSN, 12
- WSA\_ERR\_USBINITFAILED, 12
- WSA\_ERR\_USBNOTAVBL, 12
- WSA\_ERR\_USBOPENFAILED, 12
- WSA\_ERR\_WINSOCKSTARTUPFAILED, 12
- WSA\_ERR\_WSAINUSE, 12
- WSA\_ERR\_WSANOTRDY, 12
- wsa\_get\_err\_msg, 12
- wsa\_frame\_header, 4
  - freq, 5
  - gain, 5
  - prod\_serial, 5
  - sample\_size, 5
  - time\_stamp, 5
- wsa\_gain
  - wsa\_lib.h, 19
- wsa\_get\_err\_msg
  - wsa\_error.h, 12
- wsa\_get\_frame
  - wsa\_lib.cpp, 15
  - wsa\_lib.h, 21
- wsa\_lib.cpp, 13
  - MAX\_FILE\_LINES, 13
  - SEP\_CHARS, 13
  - wsa\_connect, 13
  - wsa\_dev\_init, 14
  - wsa\_disconnect, 14
  - wsa\_get\_frame, 15
  - wsa\_list\_devs, 15
  - wsa\_query\_error, 15
  - wsa\_send\_command, 16
  - wsa\_send\_command\_file, 16
  - wsa\_send\_query, 17
  - wsa\_tokenize\_file, 17
- wsa\_lib.h, 18
  - FALSE, 19
  - NUM\_RF\_GAINS, 19
  - SCPI, 19
  - TRUE, 19
  - wsa\_connect, 20
  - wsa\_disconnect, 21
  - wsa\_gain, 19
  - wsa\_get\_frame, 21
  - wsa\_list\_devs, 22
  - wsa\_query\_error, 22
  - wsa\_send\_command, 22
  - wsa\_send\_command\_file, 23
  - wsa\_send\_query, 23
- wsa\_lib.txt, 24
- wsa\_list\_devs
  - wsa\_lib.cpp, 15
  - wsa\_lib.h, 22
- wsa\_query\_error
  - wsa\_lib.cpp, 15
  - wsa\_lib.h, 22
- wsa\_resp, 6
  - result, 6
  - status, 6
- wsa\_send\_command
  - wsa\_lib.cpp, 16
  - wsa\_lib.h, 22
- wsa\_send\_command\_file
  - wsa\_lib.cpp, 16
  - wsa\_lib.h, 23
- wsa\_send\_query
  - wsa\_lib.cpp, 17
  - wsa\_lib.h, 23
- wsa\_socket, 6
  - cmd, 6
  - data, 6
- wsa\_time, 6
  - nsec, 7
  - sec, 7
- wsa\_tokenize\_file
  - wsa\_lib.cpp, 17