Standard WSA Library

Generated by Doxygen 1.7.4

Tue Aug 16 2011 15:20:27

Contents

1 Introduction

The WSA4000 LIB is a library with high level interfaces to a WSA4000 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 protocols, such as SCPI, to get WSA status, set gain, set centre frequency, etc., and perform data acquisition.

The WSA4000 LIB is designed using mixed C/C++ languages. To use the library, you need to include the header file, wsa4k_lib.h, in files that will use any of its functions to access a WSA4000, and a link to the wsa4k_lib.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.

`#include <wsa_lib.h>`

Data Fields

- char prod_name [50]
- char prod_serial [20]
- char prod_version [20]
- char rfe_name [20]
- char rfe_version [20]
- char fw_version [20]

4.1.1 Detailed Description

This structure stores WSA information.

4.1.2 Field Documentation

4.1.2.1 char fw_version

The firmware version currently in the WSA.

4.1.2.2 char prod_name

WSA product name.

4.1.2.3 char prod_serial

WSA product serial number.

4.1.2.4 char prod_version

WSA product version number.

4.1.2.5 char rfe_name

WSA product name.

4.1.2.6 char rfe_version

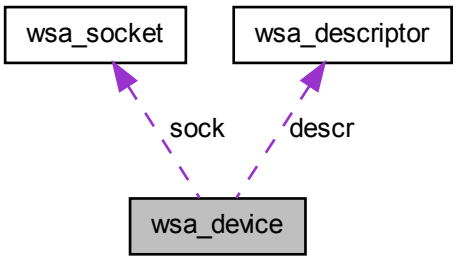
WSA product version number.

4.2 wsa_device Struct Reference

A structure containing the components associate with each WSA device.

`#include <wsa_lib.h>`

Collaboration diagram for wsa_device:



Data Fields

- struct wsa_descriptor descr
- struct wsa_socket sock

4.2.1 Detailed Description

A structure containing the components associate with each WSA device.

4.2.2 Field Documentation

4.2.2.1 struct wsa_descriptor descr

The information component of the WSA, stored in wsa_descriptor.

4.2.2.2 struct wsa_socket sock

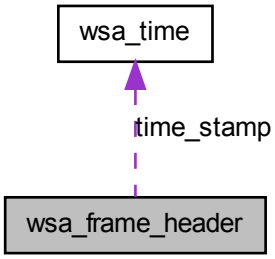
The socket structure component of the WSA, used for TCPIP connection.

4.3 wsa_frame_header Struct Reference

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

`#include <wsa_lib.h>`

Collaboration diagram for wsa_frame_header:



Data Fields

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

4.3.1 Detailed Description

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

4.3.2 Field Documentation

4.3.2.1 uint32_t frame_size

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

4.3.2.2 uint64_t freq

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

4.3.2.3 char gain

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

4.3.2.4 char prod_serial[20]

4.3.2.5 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.

`#include <wsa_lib.h>`

Data Fields

- int32_t status
- char * result

4.4.1 Detailed Description

This structure contains the response information for each query.

4.4.2 Field Documentation

4.4.2.1 char* result

4.4.2.2 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.

`#include <wsa_lib.h>`

Data Fields

- SOCKET cmd

- SOCKET data

4.5.1 Detailed Description

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

4.5.2 Field Documentation

4.5.2.1 SOCKET cmd

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

4.5.2.2 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.

`#include <wsa_lib.h>`

Data Fields

- int32_t sec
- uint32_t nsec

4.6.1 Detailed Description

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

4.6.2 Field Documentation

4.6.2.1 int32_t nsec

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

4.6.2.2 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

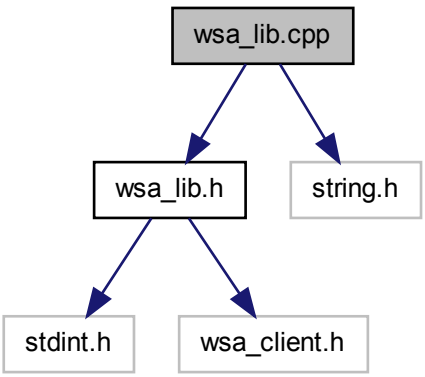
5.1.1.2 and information about the and project features selected with the Application Wizard wsa4000_cli cpp This is the main application source file Other standard files

5.1.1.3 and information about the platforms

5.2 wsa_lib.cpp File Reference

```
#include "wsa_lib.h"
#include <string.h>
```

Include dependency graph for wsa_lib.cpp:



Functions

- int32_t wsa_connect (struct wsa_device ∗dev, char ∗protocol, char ∗intf_method)
- int32_t wsa_close (struct wsa_device ∗dev)
- int32_t wsa_help (struct wsa_device dev)
- int32_t wsa_send_command (struct wsa_device ∗dev, char ∗command)
- struct wsa_resp wsa_send_query (struct wsa_device ∗dev, char ∗command)
- int32_t wsa_query_error (struct wsa_device ∗dev)
- int32_t wsa_read_data (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int32_t ∗i_buf, int32_t ∗q_buf, uint32_t frame_size)

5.2.1 Function Documentation

5.2.1.1 int32_t wsa_close (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	- The WSA device structure to be closed.

Returns

0 on success, or a negative number on error.

5.2.1.2 int32_t wsa_connect (struct wsa_device ∗ *dev,* char ∗ *protocol,* char ∗ *intf_method*)

Connect to a WSA through the specified interface method **intf_method**, and communicate control commands in the given **protocol** format

Parameters

dev	- the WSA device structure to be connected/establised.
protocol	- The standard for control commands communication to the WSA. Currently supported protocols are: SCPI, CLI(?).
intf_method	- 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....

5.2.1.3 int32_t wsa_help (struct wsa_device *dev*)

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.

5.2.1.4 int32_t wsa_query_error (struct wsa_device * *dev*)

Querry the WSA for any error

Parameters

dev	- The WSA device structure from which the error query is sent to

Returns

0 on success, or a negative number on error.

5.2.1.5 int32_t wsa_read_data (struct wsa_device * *dev*, struct wsa_frame_header * *header*, int32_t * *i_buf*, int32_t * *q_buf*, uint32_t *frame_size*)

Parameters

dev	- The WSA device structure from which the command is sent to.
header	- A wsa_frame_header structure of information for the frame.
i_buf	- The unscaled, signed integer I data buffer with size specified by the frame_size.
q_buf	- The unscaled, signed integer Q data buffer with size specified by the frame_size.
frame_size	- The number of samples (i.e. {I, Q} sample pairs) to be captured per frame.

Returns

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

5.2.1.6 int32_t wsa_send_command (struct wsa_device * *dev*, char * *command*)

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

Parameters

dev	- The WSA device structure from which the command is sent to
command	- The control command string written in the format specified by the protocol in wsa_connect()

Returns

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

5.2.1.7 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 protocol in wsa_connect().

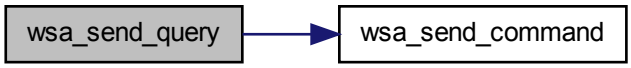
Parameters

dev	- The WSA device structure from which the query is sent to
command	- The query command string written in the format specified by the protocol in wsa_connect()

Returns

The result stored in a wsa_resp struct format.

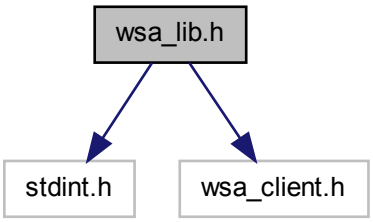
Here is the call graph for this function:



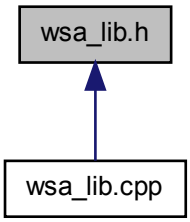
5.3 wsa_lib.h File Reference

```
#include "stdint.h"
#include "wsa_client.h"
```

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_read_data().
- 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 SCPI "SCPI"

Functions

- int32_t wsa_connect (struct wsa_device ∗dev, char ∗protocol, char ∗intf_method)
- int32_t wsa_close (struct wsa_device ∗dev)
- int32_t wsa_help (struct wsa_device dev)
- int32_t wsa_send_command (struct wsa_device ∗dev, char ∗command)

- struct wsa_resp wsa_send_query (struct wsa_device ∗dev, char ∗command)
- int32_t wsa_query_error (struct wsa_device ∗dev)
- int32_t wsa_read_data (struct wsa_device ∗dev, struct wsa_frame_header ∗header, int32_t ∗i_buf, int32_t ∗q_buf, uint32_t frame_size)

5.3.1 Define Documentation

5.3.1.1 #define SCPI "SCPI"

5.3.2 Function Documentation

5.3.2.1 int32_t wsa_close (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	- The WSA device structure to be closed.

Returns

0 on success, or a negative number on error.

5.3.2.2 int32_t wsa_connect (struct wsa_device ∗ *dev,* char ∗ *protocol,* char ∗ *intf_method*)

Connect to a WSA through the specified interface method **intf_method**, and communicate control commands in the given **protocol** format

Parameters

dev	- the WSA device structure to be connected/establised.
protocol	- The standard for control commands communication to the WSA. Currently supported protocols are: SCPI, CLI(?).
intf_method	- 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....

5.3.2.3 int32_t wsa_help (struct wsa_device *dev*)

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.

5.3.2.4 int32_t wsa_query_error (struct wsa_device * *dev*)

Querry the WSA for any error

Parameters

dev	- The WSA device structure from which the error query is sent to

Returns

0 on success, or a negative number on error.

5.3.2.5 int32_t wsa_read_data (struct wsa_device * *dev*, struct wsa_frame_header * *header*, int32_t * *i_buf*, int32_t * *q_buf*, uint32_t *frame_size*)

Parameters

dev	- The WSA device structure from which the command is sent to.
header	- A wsa_frame_header structure of information for the frame.
i_buf	- The unscaled, signed integer I data buffer with size specified by the frame_size.
q_buf	- The unscaled, signed integer Q data buffer with size specified by the frame_size.
frame_size	- The number of samples (i.e. {I, Q} sample pairs) to be captured per frame.

Returns

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

5.3.2.6 int32_t wsa_send_command (struct wsa_device * *dev*, char * *command*)

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

Parameters

dev	- The WSA device structure from which the command is sent to
command	- The control command string written in the format specified by the protocol in wsa_connect()

Returns

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

5.3.2.7 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 protocol in wsa_connect().

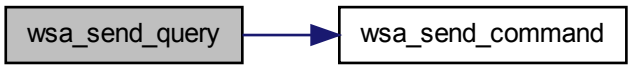
Parameters

dev	- The WSA device structure from which the query is sent to
command	- The query command string written in the format specified by the protocol 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

Contain some code documents for wsa_lib.h.

Index