WSA4000 CLI (Command Line Interface) Program Documentation

Generated by Doxygen 1.7.4

Mon Aug 29 2011 14:11:58

Contents

1 Introduction 1

2 Data Structure Index 2

2.1 Data Structures 2

3 Data Structure Documentation 2

3.1 wsa_descriptor Struct Reference 2

3.1.1 Detailed Description 3

3.1.2 Field Documentation 3

3.2 wsa_device Struct Reference 4

3.2.1 Detailed Description 4

3.2.2 Field Documentation 4

3.3 wsa_frame_header Struct Reference 5

3.3.1 Detailed Description 5

3.3.2 Field Documentation 5

3.4 wsa_resp Struct Reference 6

3.4.1 Detailed Description 6

3.4.2 Field Documentation 6

3.5 wsa_socket Struct Reference 7

3.5.1 Detailed Description 7

3.5.2 Field Documentation 7

3.6 wsa_time Struct Reference 7

3.6.1 Detailed Description 7

3.6.2 Field Documentation 8

1 Introduction

This documentation, compiled using Doxygen, shows in details the code structure of the CLI (Command Line Interface) tool. It provides information on all the libraries involved.

The following diagram illustrates the different layers and libraries involved in interfacing with a WSA on the PC side.

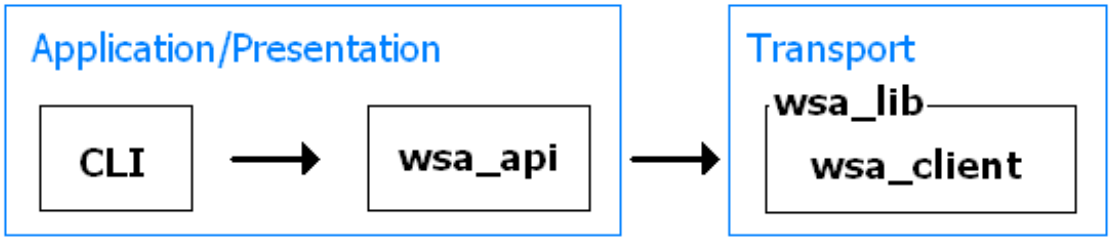


Figure 1: Interface Layers to WSA on PC Side

The CLI interfaces to a WSA through the wsa_api library, which provides functions to set/get particular settings or 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 decodes any responses or packet coming back from the WSA through the wsa_lib.

The wsa_lib, thus, is the main gateway to a WSA box from a PC. The wsa_lib has functions to open, close, send/receive commands, querry the WSA box status, and get data. In this CLI version, wsa_lib calls the wsa_client's functions in the transport layer to establish TCP/IP specific connections. Other connection methods such as USB could be added to the transport layer later on. The wsa_lib, thus, abstracts away the interface method from any application/presentation program calling it.

The CLI, hence, is a direct example of how the wsa_api library could be used. Data will

The WSA4000 CLI is designed using mixed C/C++ languages. The CLI when executed will run in a Windows command prompt console. List of commands available with the CLI is listed in the print_cli_menu() function.

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.

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()) 5

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) 7

wsa_time (This structure contains the time information. It is used for the time stamp in a frame header) 7

3 Data Structure Documentation

3.1 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 [20]
- 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

3.1.1 Detailed Description

This structure stores WSA information.

3.1.2 Field Documentation

3.1.2.1 char fw_version

The firmware version currently in the WSA.

3.1.2.2 uint64_t inst_bw

The WSA instantaneous bandwidth in Hz.

3.1.2.3 char intf_type

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

3.1.2.4 uint64_t max_sample_size

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

3.1.2.5 uint64_t max_tune_freq

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

3.1.2.6 uint64_t min_tune_freq

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

3.1.2.7 char prod_name

WSA product name.

3.1.2.8 char prod_serial

WSA product serial number.

3.1.2.9 char prod_version

WSA product version number.

3.1.2.10 char rfe_name

WSA product name.

3.1.2.11 char rfe_version

WSA product version number.

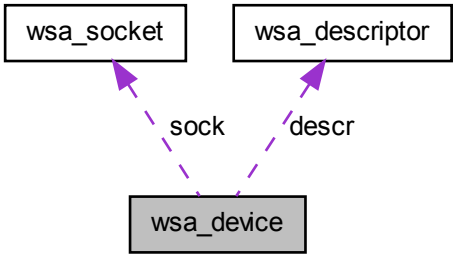
The documentation for this struct was generated from the following files:

- wsa_api.h
- wsa_lib.h
- wsa_lib.txt

3.2 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

3.2.1 Detailed Description

A structure containing the components associate with each WSA device.

3.2.2 Field Documentation

3.2.2.1 struct wsa_descriptor descr

The information component of the WSA, stored in wsa_descriptor.

3.2.2.2 struct wsa_socket sock

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

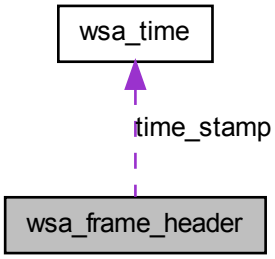
The documentation for this struct was generated from the following files:

- wsa_api.h
- wsa_lib.h
- wsa_lib.txt

3.3 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 frame_size
- struct wsa_time time_stamp

3.3.1 Detailed Description

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

3.3.2 Field Documentation

3.3.2.1 uint32_t frame_size

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

3.3.2.2 uint64_t freq

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

3.3.2.3 char gain

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

3.3.2.4 char prod_serial

WSA product version number.

3.3.2.5 struct wsa_time time_stamp

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

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

- wsa_api.h
- wsa_lib.h
- wsa_lib.txt

3.4 wsa_resp Struct Reference

This structure contains the response information for each query.

Data Fields

- int64_t status
- char * result

3.4.1 Detailed Description

This structure contains the response information for each query.

3.4.2 Field Documentation

3.4.2.1 char result

The resulted string responded to a query.

3.4.2.2 int32_t status

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

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

- wsa_lib.h
- wsa_lib.txt

3.5 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

3.5.1 Detailed Description

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

3.5.2 Field Documentation

3.5.2.1 SOCKET cmd

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

3.5.2.2 SOCKET data

The data socket used for streaming of data

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

- wsa_api.h
- wsa_lib.h
- wsa_lib.txt

3.6 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

3.6.1 Detailed Description

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

3.6.2 Field Documentation

3.6.2.1 int32_t nsec

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

3.6.2.2 int32_t sec

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

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

- wsa_api.h
- wsa_lib.h
- wsa_lib.txt

Index

cmd
    wsa_socket, 7

data
    wsa_socket, 7
descr
    wsa_device, 4

frame_size
    wsa_frame_header, 5
freq
    wsa_frame_header, 5
fw_version
    wsa_descriptor, 3

gain
    wsa_frame_header, 6

inst_bw
    wsa_descriptor, 3
intf_type
    wsa_descriptor, 3

max_sample_size
    wsa_descriptor, 3
max_tune_freq
    wsa_descriptor, 3
min_tune_freq
    wsa_descriptor, 3

nsec
    wsa_time, 8

prod_name
    wsa_descriptor, 3
prod_serial
    wsa_descriptor, 3
    wsa_frame_header, 6
prod_version
    wsa_descriptor, 3

result
    wsa_resp, 6
rfe_name
    wsa_descriptor, 3
rfe_version
    wsa_descriptor, 3

sec
    wsa_time, 8
sock
    wsa_device, 4
status
    wsa_resp, 6

time_stamp
    wsa_frame_header, 6

wsa_descriptor, 2
    fw_version, 3
    inst_bw, 3
    intf_type, 3
    max_sample_size, 3
    max_tune_freq, 3
    min_tune_freq, 3
    prod_name, 3
    prod_serial, 3
    prod_version, 3
    rfe_name, 3
    rfe_version, 3
wsa_device, 4
    descr, 4
    sock, 4
wsa_frame_header, 5
    frame_size, 5
    freq, 5
    gain, 6
    prod_serial, 6
    time_stamp, 6
wsa_resp, 6
    result, 6
    status, 6
wsa_socket, 7
    cmd, 7
    data, 7
wsa_time, 7
    nsec, 8
    sec, 8