SPDAUlib

v01.04

Generated by Doxygen 1.7.3

Mon Oct 17 2011 18:20:08

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Module Documentation

2.1 Version control

Functions

- SPDAU_DLL_API int SPDAU_dll_version (char *dll_version)
  Requests DLL release version.
- SPDAU_DLL_API int SPDAU_hardware_id (char *hardware_id)
  Requests release verion of SPDAU hardware and firmware units.
- SPDAU_DLL_API int SPDAU_get_error_string (char *error_string)
  Requests description of last error caugth during DLL execution.
- SPDAU_DLL_API int SPDAU_status (char *status_str)
  Requests remote SPDAU logger status.
- SPDAU_DLL_API int SPDAU_Software_status (char *status_str)
  Requests remote Software SPDAU status.

2.1.1 Detailed Description

Functions lookup revision numbers of software, hardware elements of SPDAU and status messages.

2.1.2 Function Documentation

2.1.2.1 SPDAU_DLL_API int SPDAU_dll_version (char * *dll_version*)

int SPDAU_dll_version(char * dll_version);

Requests DLL release version

Parameters

out	*dll_version*	String contaning release version of this DLL API

Returns

1 on success otherwise -1

2.1.2.2 SPDAU_DLL_API int SPDAU_hardware_id (char * *hardware_id*)

int SPDAU_hardware_id(char * hardware_id);

Requests release verion of SPDAU hardware and firmware units

Parameters

out	*hardware_id*	String contaning release version of remote firmware,i.e. embedded application, device driver, VHDL-FPGA revisions and hardware revision number.

Returns

1 on success otherwise -1

2.1.2.3 SPDAU_DLL_API int SPDAU_get_error_string (char * *error_string*)

int SPDAU_get_error_string(char * error_string);

Requests description of last error caugth during DLL execution

Parameters

out	*error_string*	String contaning description of last error

Returns

1 if error_string is a valid string

2.1.2.4 SPDAU_DLL_API int SPDAU_status (char * *status_str*)

int SPDAU_status(char * status_str);

Requests remote SPDAU logger status, If successfull this function must return one string defined in the function parameter description. On Communication error this function returns a Software Status String.

Parameters

out	*status_str*	String contaning description of present SPDAU logger status (possible strings) • "Idle" (Logger disabled) • "Armed" (Waiting for trigger) • "Triggered" (Recording, filling buffers) • "Copy" (Transfering data from Logger memory to Average memory) • "Done" (Programmed sequence finished)

Returns

1 if status_str is a valid string

2.1.2.5 SPDAU_DLL_API int SPDAU_Software_status (char * *status_str*)

int SPDAU_Software_status(char * status_str);

Requests remote Software SPDAU status

Parameters

out	*status_str*	String contaning description of present SPDAU status

Returns

1 if status_str is a valid string

2.2 Connectivity

Functions

- SPDAU_DLL_API int SPDAU_open (char *ipaddress)

  Opens TCP/IP connection.

- SPDAU_DLL_API int SPDAU_close (void)

  Closes TCP/IP connection.

2.2.1 Detailed Description

Functions establish communication to remote SPDAU hardware over ethernet.

2.2.2 Function Documentation

2.2.2.1 SPDAU_DLL_API int SPDAU_open (char * *ipaddress*)

int SPDAU_open(char * ipaddress);

Opens a TCP/IP connection over ethernet to remote SPDAU with a given IP address. Initialises DLL threads listening to a socket on host-side for receiving incoming data. This function must be called before trying to communicate to SPDAU.

Parameters

in	*ipaddress*	String contaning remote IP address

Returns

1 on success otherwise -1

2.2.2.2 SPDAU_DLL_API int SPDAU_close (void)

int SPDAU_close(void);

Closes TCP/IP connection with remote SPDAU. Stops DLL threads needed for communication. This function must be called as last command before closing the host application.

Returns

1 on success otherwise -1

2.3 Data Logger configuration

Functions

- SPDAU_DLL_API int SPDAU_set_samp_freq (unsigned long freq_in_mhz)
  Sets sampling frequency.
- SPDAU_DLL_API int SPDAU_get_samp_freq (unsigned long *freq_in_mhz)
  Requests sampling frequency.
- SPDAU_DLL_API int SPDAU_set_BWLimit (unsigned long limit_mhz)
  Sets analog input bandwith limit.
- SPDAU_DLL_API int SPDAU_get_BWLimit (unsigned long *limit_mhz)
  Requests analog input bandwith limit setup.
- SPDAU_DLL_API int SPDAU_set_trigger_source (unsigned long source_val)
  Trigger Source selection.

- SPDAU_DLL_API int SPDAU_get_trigger_source (unsigned long *source_val)

  Trigger Source.

- SPDAU_DLL_API int SPDAU_set_trigger_mode (unsigned long mode_val)

  Trigger mode selection.

- SPDAU_DLL_API int SPDAU_get_trigger_mode (unsigned long *mode_val)

  Trigger mode.

- SPDAU_DLL_API int SPDAU_set_trigger_edge (unsigned long edge_val)

  Trigger edge selection.

- SPDAU_DLL_API int SPDAU_get_trigger_edge (unsigned long *edge_val)

  Trigger edge request.

- SPDAU_DLL_API int SPDAU_set_num_takes (unsigned long number_of_takes)

  Sets the number of loops for a measurement.

- SPDAU_DLL_API int SPDAU_get_num_takes (unsigned long *number_of_takes)

  Gets the number of loops for a measurement.

- SPDAU_DLL_API int SPDAU_set_pretrig_len (unsigned long pretrig_len_us)

  Sets pretrigger buffer length.

- SPDAU_DLL_API int SPDAU_get_pretrig_len (unsigned long *pretrig_len_us)

  Requests prettrigger buffer length.

- SPDAU_DLL_API int SPDAU_set_posttrig_len (unsigned long posttrig_len_us)

  Sets posttrigger buffer length.

- SPDAU_DLL_API int SPDAU_get_posttrig_len (unsigned long *posttrig_len_us)

  Requests posttrigger buffer length.

2.3.1 Detailed Description

Functions configure SPDAU data logger and oscilloscope functionality.

2.3.2 Function Documentation

2.3.2.1 SPDAU_DLL_API int SPDAU_set_samp_freq (unsigned long *freq_in_mhz*)

int SPDAU_set_samp_freq(unsigned long freq_in_mhz);

Sets sampling frequency of SPDAU logger engine. Maximum frequency value is 400 (MHz).

Parameters

in	*freq_in_mhz*	frequency value in megaherzs

Returns

1 on success otherwise -1

2.3.2.2 SPDAU_DLL_API int SPDAU_get_samp_freq (unsigned long * *freq_in_mhz*)

int SPDAU_get_samp_freq(unsigned long *freq_in_mhz);

Requests sampling frequency of SPDAU logger engine. Returns the frequency value in MHz.

Parameters

out	*freq_in_mhz*	frequency value in megaherzs

Returns

1 on success, otherwise -1

2.3.2.3 SPDAU_DLL_API int SPDAU_set_BWLimit (unsigned long *limit_mhz*)

int SPDAU_set_BWLimit(unsigned long limit_mhz);

Configure the 41-taps low pass FIR filter with a cut frequency limit_mhz in MegaHerzs.

Parameters

in	*limit_mhz*	cut frequency of the input low pass filter in MHz. A value of 0 or bigger than 199 disables de filter.

Returns

1 on success otherwise -1

2.3.2.4 SPDAU_DLL_API int SPDAU_get_BWLimit (unsigned long * *limit_mhz*)

SPDAU_DLL_API int SPDAU_get_BWLimit(unsigned long * limit_mhz);

Requests analog input bandwith limit setup.

Parameters

out	*limit_mhz*	cut frequency of the low pass filter at the input signal path.

Returns

1 on success, otherwise -1

2.3.2.5 SPDAU_DLL_API int SPDAU_set_trigger_source (unsigned long *source_val*)

int SPDAU_set_trigger_source(unsigned long source_val);

Selects the trigger source of the recording and oscilloscope engines. The trigger source can be selected between the digital outputs, digital inputs or analog input.

Parameters

in	*source_val*	Trigger source selected by integer (see definition) • #define SPDAU_TRIGSOURCE_DIGOUT1 0 • #define SPDAU_TRIGSOURCE_DIGOUT2 1 • #define SPDAU_TRIGSOURCE_DIGOUT3 2 • #define SPDAU_TRIGSOURCE_DIGOUT4 3 • #define SPDAU_TRIGSOURCE_DIGIN1 4 • #define SPDAU_TRIGSOURCE_DIGIN2 5 • #define SPDAU_TRIGSOURCE_DIGIN3 6 • #define SPDAU_TRIGSOURCE_DIGIN4 7 • #define SPDAU_TRIGSOURCE_ANIN 8

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.6 SPDAU_DLL_API int SPDAU_get_trigger_source (unsigned long * *source_val*)

int SPDAU_get_trigger_source(unsigned long *source_val);

Requests the trigger source of recording and oscilloscope engines. The trigger source can be selected between the digital outputs, digital inputs or analog input.

Parameters

out	*source_val*	On success, trigger source selected (see definition on SPDAU_-set_trigger_source)

Returns

1 on success, otherwise -1

2.3.2.7 SPDAU_DLL_API int SPDAU_set_trigger_mode (unsigned long *mode_val*)

int SPDAU_set_trigger_mode(unsigned long mode_val);

Selects the trigger mode of the oscilloscope engine. The trigger can be configured as auto trigger or normal trigger.

Parameters

in	*mode_val*	Trigger mode selected by integer (see definition) • #define SPDAU_TRIGMODE_NORMAL 0 • #define SPDAU_TRIGMODE_AUTO 1

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.8 SPDAU_DLL_API int SPDAU_get_trigger_mode (unsigned long * *mode_val*)

int SPDAU_get_trigger_mode(unsigned long * mode_val);

Requests the trigger mode of the oscilloscope engine.. The trigger can be configured as auto trigger or normal trigger.

Parameters

out	*mode_val*	Trigger mode selected by integer (see definition on SPDAU_set_trigger_mode)

Returns

1 on Success, otherwise -1

2.3.2.9 SPDAU_DLL_API int SPDAU_set_trigger_edge (unsigned long *edge_val*)

int SPDAU_set_trigger_edge(unsigned long edge_val);

Selects the trigger edge of the recording and oscilloscope engines. The trigger can be configured sensitive to rising edge or falling edge of the signal selected by 'int SPDAU_set_trigger_source(int trig_source)'.

Parameters

in	*edge_val*	Trigger edge selected by integer (see definition) • #define SPDAU_TRIGEDGE_FALLING 0 • #define SPDAU_TRIGEDGE_RISING 1

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.10 SPDAU_DLL_API int SPDAU_get_trigger_edge (unsigned long * *edge_val*)

int SPDAU_get_trigger_edge(unsigned long * edge_val);

Requests the trigger edge of the recording and oscilloscope engines. The trigger can be configured sensitive to rising edge or falling edge of the signal selected by 'int SPDAU_set_trigger_source(int trig_source)'.

Parameters

out	*edge_val*	On Success, trigger edge selected by integer (see definition SPDAU_set_trigger_edge)

Returns

1 on success, otherwise -1

2.3.2.11 SPDAU_DLL_API int SPDAU_set_num_takes (unsigned long *number_of_takes*)

int SPDAU_set_num_takes(unsigned long number_of_takes);

Sets the number of loops for a measurement/averaging process.

Parameters

in	*number_of_takes*	Number of loops. Number between 1 and 4096.

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.12 SPDAU_DLL_API int SPDAU_get_num_takes (unsigned long * *number_of_takes*)

int SPDAU_get_num_takes(unsigned long *number_of_takes);

Requests the number of loops configured for a measurement/averaging process.

Parameters

out	*number_of_takes*	Number of loops. Number between 1 and 4096.

Returns

1 on success, otherwise -1

2.3.2.13 SPDAU_DLL_API int SPDAU_set_pretrig_len (unsigned long *pretrig_len_us*)

int SPDAU_set_pretrig_len(unsigned long pretrig_len_us);

Sets the length of the recording pretrigger buffer. It configures signal length stored in memory before a trigger event occurs.

Parameters

in	*pretrig_len_us*	Pretrigger buffer length in microseconds.

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.14 SPDAU_DLL_API int SPDAU_get_pretrig_len (unsigned long * *pretrig_len_us*)

int SPDAU_get_pretrig_len(unsigned long *pretrig_len_us);

Requests the length of the recording pretrigger buffer.

Parameters

out	*pretrig_len_us*	Pretrigger buffer length in microseconds.

Returns

1 on success, otherwise -1

2.3.2.15 SPDAU_DLL_API int SPDAU_set_posttrig_len (unsigned long *posttrig_len_us*)

int SPDAU_set_posttrig_len(unsigned long posttrig_len_us);

Sets the length of the recording postrigger buffer. It configures signal length stored in memory after a trigger event occurs.

Parameters

in	*posttrig_len_us*	Posttrigger buffer length in microseconds.

Returns

1 if the command has been sent succesfully, otherwise -1

2.3.2.16 SPDAU_DLL_API int SPDAU_get_posttrig_len (unsigned long * *posttrig_len_us*)

int SPDAU_get_posttrig_len(unsigned long *posttrig_len_us);

Requests the length of the recording postrigger buffer.

Parameters

out	*posttrig_len_us*	Posttrigger buffer length in microseconds.

Returns

1 on success, otherwise -1

2.4 Signal generator configuration

Functions

- SPDAU_DLL_API int SPDAU_set_digout_enable (unsigned long enable, unsigned long chan)

  Preconfigure digital outputs enable flags.

- SPDAU_DLL_API int SPDAU_get_digout_enable (unsigned long *enable, unsigned long chan)

  Read digital output enable flags.

- SPDAU_DLL_API int SPDAU_set_digout_timer (unsigned long timer_val, unsigned long chan)

  Sets the timer value for cyclic digital output deployment on a given channel.

- SPDAU_DLL_API int SPDAU_get_digout_timer (unsigned long *timer_val, unsigned long chan)

  Requests the timer value for cyclic digital output deployment on a given channel.

- SPDAU_DLL_API int SPDAU_set_digout_mode (unsigned long mode_val, unsigned long chan)

  Sets the mode of deployment for a given digital output channel.

- SPDAU_DLL_API int SPDAU_get_digout_mode (unsigned long *mode_val, unsigned long chan)

  Requests the mode of deployment for a given digital output channel.

- SPDAU_DLL_API int SPDAU_set_digout_type (unsigned long type_val, unsigned long chan)

  Sets the signal type assigned to a given digital output channel.

- SPDAU_DLL_API int SPDAU_get_digout_type (unsigned long *type_val, unsigned long chan)

  Requests the signal type assigned to a given digital output channel.

- SPDAU_DLL_API int SPDAU_set_digout_pulsewidth (unsigned long width_-val, unsigned long chan)

  Sets the pulse width on a given digital output channel.

- SPDAU_DLL_API int SPDAU_get_digout_pulsewidth (unsigned long *width_-val, unsigned long chan)

  Requests the pulse width on a given digital output channel.

- SPDAU_DLL_API int SPDAU_set_digout_lut (unsigned long *lut_val, unsigned long chan)

  Sets the bitMap look up table for a given digital output channel.

- SPDAU_DLL_API int SPDAU_get_digout_lut (unsigned long *lut_val, unsigned long chan)

  Requests the bitMap look up table for a given digital output channel.

- SPDAU_DLL_API int SPDAU_set_digout_button (unsigned long button_val, unsigned long chan)

  Assigns a hardware push button to a given digital output signal.

- SPDAU_DLL_API int SPDAU_get_digout_button (unsigned long *button_val, unsigned long chan)

  Requests the assignment of the hardware push button to a given digital output signal.

- SPDAU_DLL_API int SPDAU_set_anout_enable (unsigned long enable, unsigned long chan)

  Preconfigure analog outputs enable flags.

- SPDAU_DLL_API int SPDAU_get_anout_enable (unsigned long *enable, unsigned long chan)

  Read analog output enable flags.

- SPDAU_DLL_API int SPDAU_set_anout_timer (unsigned long timer_val, unsigned long chan)

  Sets the timer value for cyclic analog output deployment on a given channel.

- SPDAU_DLL_API int SPDAU_get_anout_timer (unsigned long *timer_val, unsigned long chan)

  Requests the timer value for cyclic analog output deployment on a given channel.

- SPDAU_DLL_API int SPDAU_set_anout_mode (unsigned long mode_val, unsigned long chan)

  Sets the mode of deployment for a given analog output channel.

- SPDAU_DLL_API int SPDAU_get_anout_mode (unsigned long *mode_val, unsigned long chan)

  Requests the mode of deployment for a given analog output channel.

- SPDAU_DLL_API int SPDAU_set_anout_type (unsigned long type_val, unsigned long chan)

  Sets the signal type assigned to a given analog output channel.

- SPDAU_DLL_API int SPDAU_get_anout_type (unsigned long *type_val, unsigned long chan)

  Requests the signal type assigned to a given analog output channel.

- SPDAU_DLL_API int SPDAU_set_anout_lut (unsigned long *lut_val, unsigned long chan)

  Sets the look up table for a given analog output channel.

- SPDAU_DLL_API int SPDAU_get_anout_lut (unsigned long *lut_val, unsigned long chan)

  Requests the look up table for a given analog output channel.

- SPDAU_DLL_API int SPDAU_set_anout_offset (unsigned long offset_val, unsigned long chan)

  Sets the offset value for a selected analog output channel.

- SPDAU_DLL_API int SPDAU_get_anout_offset (unsigned long *offset_val, unsigned long chan)

  Requests the offset value for a selected analog output channel.

- SPDAU_DLL_API int SPDAU_set_anout_button (unsigned long button_val, unsigned long chan)

  Assigns a hardware push button to a given analog output signal.

- SPDAU_DLL_API int SPDAU_get_anout_button (unsigned long *button_val, unsigned long chan)

  Requests the assignment of the hardware push button to a given analog output signal.

- SPDAU_DLL_API int SPDAU_set_anout_attenuation (double attenuation_val, unsigned long chan)

  Sets an attenuation factor to a given analog output channel.

- SPDAU_DLL_API int SPDAU_get_anout_attenuation (double *attenuation_val, unsigned long chan)

  Requests an attenuation factor to a given analog output channel.

- SPDAU_DLL_API int SPDAU_set_offsetaux (unsigned long offset_val, unsigned long chan)

  Loads a value to an auxiliary offset output channel.

- SPDAU_DLL_API int SPDAU_get_offsetaux (unsigned long *offset_val, unsigned long chan)

  Requests a value from auxiliary offset channel.

2.4.1 Detailed Description

Functions configure SPDAU signal generator functionality.

2.4.2 Function Documentation

2.4.2.1 SPDAU_DLL_API int SPDAU_set_digout_enable (unsigned long *enable*, unsigned long *chan*)

int SPDAU_set_digout_enable(unsigned long enable,unsigned long chan);

Loads enable flag of a given digital output to a shadow register in remote hardware.

Parameters

in	*enable*	Enable flag of the selected channel. • 0 : enable • 1 : disable
in	*chan*	Selects the digital output signal to be enabled/disabled. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 if the command has been sent succesfully, otherwise -1

2.4.2.2 SPDAU_DLL_API int SPDAU_get_digout_enable (unsigned long * *enable*, unsigned long *chan*)

int SPDAU_get_digout_enable(unsigned long *enable,unsigned long chan);

Requests a digital output enable flag of a given digital output channel.

Parameters

out	*enable*	Enable flag of the selected channel. • 0 : enable • 1 : disable
in	*chan*	Selects the digital output signal. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.3 SPDAU_DLL_API int SPDAU_set_digout_timer (unsigned long *timer_val,* unsigned long *chan*)

int SPDAU_set_digout_timer(unsigned long timer_val,unsigned long chan);

When a digital output channel is configured in Timer mode, this function sets the timer value that will set the rate at which the digital signal will be deployed. The timer granularity in 160 ns.

Parameters

in	*timer_val*	Timer value of a given digital output channel in steps of 160ns.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.4 SPDAU_DLL_API int SPDAU_get_digout_timer (unsigned long * *timer_val,* unsigned long *chan*)

int SPDAU_get_digout_timer(unsigned long *timer_val,unsigned long chan);

Requests the timer value for cyclic digital output deployment on a given channel.

Parameters

out	*timer_val*	Timer value of a given digital output channel in steps of 160ns.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.5 SPDAU_DLL_API int SPDAU_set_digout_mode (unsigned long *mode_val*, unsigned long *chan*)

int SPDAU_set_digout_mode(unsigned long mode_val,unsigned long chan);

Sets the mode of deployment for a given digital output channel

Parameters

in	*mode_val*	Deployment mode. • 0 : Single shot mode, selected digital output channel shall be activated only once. • 1 : Timer mode, selected digital output channel shall be activated at a rate specified by Timer. • 2 : Loop mode, selected digital output channel shall be activated at a fixed rate of 40.96 microseconds.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.6 SPDAU_DLL_API int SPDAU_get_digout_mode (unsigned long * *mode_val*, unsigned long *chan*)

int SPDAU_get_digout_mode(unsigned long *mode_val,unsigned long chan);

Requests the mode of deployment for a given digital output channel

Parameters

out	*mode_val*	Deployment mode. • 0 : Single shot mode, selected digital output channel shall be activated only once. • 1 : Timer mode, selected digital output channel shall be activated at a rate specified by Timer. • 2 : Loop mode, selected digital output channel shall be activated at a fixed rate of 40.96 microseconds.

in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.7 SPDAU_DLL_API int SPDAU_set_digout_type (unsigned long *type_val,* unsigned long *chan*)

int SPDAU_set_digout_type(unsigned long type_val,unsigned long chan);

Sets the signal type assigned to a given digital output channel. Every Digital output channel in SPDAU can be configured as single pulse signal (negative or positve) or as a signal derived from a user defined look up table.

Parameters

in	*type_val*	Digital out signal type. • 0 : POSPULSE_TYPE. The selected digital output channel is configured as a single positive pulse signal. • 1 : NEGPULSE_TYPE. The selected digital output channel is configured as a single negative pulse signal. • 2 : LUT_TYPE. The selected digital output channel is configured as a signal defined by a Look Up Table.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.8 SPDAU_DLL_API int SPDAU_get_digout_type (unsigned long * *type_val,* unsigned long *chan*)

int SPDAU_get_digout_type(unsigned long *type_val,unsigned long chan);

Requests the signal type assigned to a given digital output channel. Every Digital output channel in SPDAU can be configured as single pulse signal (negative or positive) or as a signal derived from a user defined look up table.

Parameters

out	*type_val*	Digital out signal type. • 0 : POSPULSE_TYPE. The selected digital output channel is configured as a single positive pulse signal. • 1 : NEGPULSE_TYPE. The selected digital output channel is configured as a single negative pulse signal. • 2 : LUT_TYPE. The selected digital output channel is configured as a signal defined by a Look Up Table.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.9 SPDAU_DLL_API int SPDAU_set_digout_pulsewidth (unsigned long *width_val,* unsigned long *chan*)

int SPDAU_set_digout_pulsewidth(unsigned long width_val,unsigned long chan);

Sets the pulse width on a given digital output channel configured in pulse mode. The configuration value must range between 1 and 8190, 1 step represents 5 nano seconds.

Parameters

in	*width_val*	. Pulse Width value between 1 and 8190.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.10 SPDAU_DLL_API int SPDAU_get_digout_pulsewidth (unsigned long * *width_val,* unsigned long *chan*)

int SPDAU_get_digout_pulsewidth(unsigned long *width_val,unsigned long chan);

Requests the pulse width on a given digital output channel configured in pulse mode. 1 step represents 5 nano seconds.

Parameters

out	*width_val*	. Pulse Width value between 1 and 8190. 5 and 40950 nano seconds
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.11 SPDAU_DLL_API int SPDAU_set_digout_lut (unsigned long * *lut_val,* unsigned long *chan*)

int SPDAU_set_digout_lut(unsigned long *lut_val,unsigned long chan);

Sets the bitMap look up table for a given digital output channel. If a digital output channel is configured as Look up Table type (LUT_TYPE), the table defined here will be used to generate the output signal when is deployed. The Look up Table is an array of 256 4-byte-long words. The signal will be generated starting from the least significant bit (bit0) of array position 0 (first bit) until the most significant bit (bit31) of array position 255 (last bit).

Parameters

in	*lut_val*	Pointer to an array of 256 unsigned long integers.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.12 SPDAU_DLL_API int SPDAU_get_digout_lut (unsigned long * *lut_val*, unsigned long *chan*)

int SPDAU_get_digout_lut(unsigned long *lut_val,unsigned long chan);

Requests the bitMap look up table configured for a given digital output channel.

Parameters

out	*lut_val*	Pointer to an array of 256 unsigned long integers.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.13 SPDAU_DLL_API int SPDAU_set_digout_button (unsigned long *button_val*, unsigned long *chan*)

int SPDAU_set_digout_button(unsigned long button_val,unsigned long chan);

This function allows the user to assign the control of the deployment of a digital output signal to the hardware push buttons present in SPDAU hardware front panel.

Parameters

in	*button_val*	Posible values • 0 : SPDAU Harware button has no influence on selected digital output signal. • 1 : SPDAU Harware button triggers the deployment of the selected digital output signal.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.14 SPDAU_DLL_API int SPDAU_get_digout_button (unsigned long * *button_val,* unsigned long *chan*)

int SPDAU_get_digout_button(unsigned long *button_val,unsigned long chan);

This function returns the assignment value of the control variable that links a digital output signal deployment to the hardware push buttons present in SPDAU hardware front panel.

Parameters

out	*button_val*	Posible values • 0 : SPDAU Harware button has no influence on selected digital output signal. • 1 : SPDAU Harware button triggers the deployment of the selected digital output signal.
in	*chan*	Selects the digital output channel. • 0 : DIGOUT1 • 1 : DIGOUT2 • 2 : DIGOUT3 • 3 : DIGOUT4

Returns

1 on success, otherwise -1

2.4.2.15 SPDAU_DLL_API int SPDAU_set_anout_enable (unsigned long *enable,* unsigned long *chan*)

int SPDAU_set_anout_enable(unsigned long enable,unsigned long chan);

Loads enable flag of a given analog output to a shadow register in remote hardware.

Parameters

in	*enable*	Enable flag of the selected channel. • 0 : enable • 1 : disable
in	*chan*	Selects the analog output signal to be enabled/disabled. • 0 : ANOUT0 • 1 : ANOUT1

Returns

1 if the command has been sent succesfully, otherwise -1

2.4.2.16 SPDAU_DLL_API int SPDAU_get_anout_enable (unsigned long * *enable*, unsigned long *chan*)

int SPDAU_get_anout_enable(unsigned long *enable,unsigned long chan);

Requests an analog output enable flag of a given analog output channel.

Parameters

out	*enable*	Enable flag of the selected channel. • 0 : enable • 1 : disable
in	*chan*	Selects the analog output signal. • 0 : ANOUT0 • 1 : ANOUT1

Returns

A positive integer indicating the value of the enable flag on success, otherwise -1

2.4.2.17 SPDAU_DLL_API int SPDAU_set_anout_timer (unsigned long *timer_val*, unsigned long *chan*)

int SPDAU_set_anout_timer(unsigned long timer_val,unsigned long chan);

When an analog output channel is configured in Timer mode, this function sets the timer value that will set the rate at which the analog signal will be deployed. The timer granularity in 12 ns.

Parameters

in	*timer_val*	Timer value of a given analog output channel in steps of 12ns.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.18 SPDAU_DLL_API int SPDAU_get_anout_timer (unsigned long * *timer_val*, unsigned long *chan*)

int SPDAU_get_anout_timer(unsigned long *timer_val,unsigned long chan);

When an analog output channel is configured in Timer mode, this function returns the

timer value that will set the rate at which the analog signal will be deployed. The timer granularity in 12 ns.

Parameters

out	*timer_val*	Timer value of a given analog output channel in steps of 12ns.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.19 SPDAU_DLL_API int SPDAU_set_anout_mode (unsigned long *mode_val,* unsigned long *chan*)

int SPDAU_set_anout_mode(unsigned long mode_val,unsigned long chan);

Sets the mode of deployment for a given analog output channel

Parameters

in	*mode_val*	Deployment mode. • 0 : Single shot mode, selected analog output channel shall be activated only once. • 1 : Timer mode, selected analog output channel shall be activated at a rate specified by Timer. • 2 : Loop mode, selected analog output channel shall be activated at a fixed rate of 40.96 microseconds.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.20 SPDAU_DLL_API int SPDAU_get_anout_mode (unsigned long * *mode_val,* unsigned long *chan*)

int SPDAU_get_anout_mode(unsigned long *mode_val,unsigned long chan);

Requests the mode of deployment for a given analog output channel

Parameters

out	*mode_val*	Deployment mode. • 0 : Single shot mode, selected analog output channel shall be activated only once. • 1 : Timer mode, selected analog output channel shall be activated at a rate specified by Timer. • 2 : Loop mode, selected analog output channel shall be activated at a fixed rate of 40.96 microseconds.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.21 SPDAU_DLL_API int SPDAU_set_anout_type (unsigned long *type_val,* unsigned long *chan*)

int SPDAU_set_anout_type(unsigned long type_val,unsigned long chan);

Sets the signal type assigned to a given analog output channel. Every analog output channel in SPDAU can be configured as a predefined signal type (SINE,SAWTOOTH,TRIANGLE,SQUARE) or as a signal derived from a user defined look up table.

Parameters

in	*type_val*	analog output signal type. • 0 : SINUS_TYPE. The selected analog output channel is configured as a sine wave. • 1 : POSTRIANGLE_TYPE. The selected analog output channel is configured as a positive triangle wave (starting with positive slope). • 2 : NEGTRIANGLE_TYPE. The selected analog output channel is configured as a negative triangle wave (starting with negative slope). • 3 : POSSAWTOOTH_TYPE. The selected analog output channel is configured as a positive saw tooth wave (starting with positive slope). • 4 : NEGSAWTOOTH_TYPE. The selected analog output channel is configured as a negative saw tooth wave (starting with negative slope). • 5 : POSSQUARE_TYPE. The selected analog output channel is configured as a positive square wave (starting with positive half cycle). • 6 : NEGSQUARE_TYPE. The selected analog output channel is configured as a negative square wave (starting with negative half cycle). • 7 : LUT_TYPE. The selected analog output channel is configured as a signal defined by a Look Up Table.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.22 SPDAU_DLL_API int SPDAU_get_anout_type (unsigned long * *type_val*, unsigned long *chan*)

int SPDAU_set_anout_type(unsigned long type_val,unsigned long chan);

Requests the signal type assigned to a given analog output channel. Every analog output channel in SPDAU can be configured as a predefined signal type (SINE,SAWTOOTH,TRIANGLE,SQUARE) or as a signal derived from a user defined look up table.

Parameters

out	*type_val*	analog output signal type. • 0 : SINUS_TYPE. The selected analog output channel is configured as a sine wave. • 1 : POSTRIANGLE_TYPE. The selected analog output channel is configured as a positive triangle wave (starting with positive slope). • 2 : NEGTRIANGLE_TYPE. The selected analog output channel is configured as a negative triangle wave (starting with negative slope). • 3 : POSSAWTOOTH_TYPE. The selected analog output channel is configured as a positive saw tooth wave (starting with positive slope). • 4 : NEGSAWTOOTH_TYPE. The selected analog output channel is configured as a negative saw tooth wave (starting with negative slope). • 5 : POSSQUARE_TYPE. The selected analog output channel is configured as a positive square wave (starting with positive half cycle). • 6 : NEGSQUARE_TYPE. The selected analog output channel is configured as a negative square wave (starting with negative half cycle). • 7 : LUT_TYPE. The selected analog output channel is configured as a signal defined by a Look Up Table.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.23 SPDAU_DLL_API int SPDAU_set_anout_lut (unsigned long * *lut_val,* unsigned long *chan*)

int SPDAU_set_anout_lut(unsigned long *lut_val,unsigned long chan);

Sets the look up table for a given analog output channel. If an analog output channel is configured as Look up Table type (LUT_TYPE), the table defined here will be used to generate the output signal when deployed. The Look up Table is an array of 8192 4-byte-long words. The signal will be generated starting from the first word at array position 0 (first word) until the last word of array position 8191. Although the array elements are specified as 4-byte words, only the lower 14 bits are valid since the digital to analog converter of SPDAU present hardware revision has a resolution of 14 bits.

Parameters

in	*lut_val*	Pointer to an array of 8192 unsigned long integers.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.24 SPDAU_DLL_API int SPDAU_get_anout_lut (unsigned long * *lut_val,* unsigned long *chan*)

int SPDAU_get_anout_lut(unsigned long *lut_val,unsigned long chan);

Requests the look up table for a given analog output channel.

Parameters

out	*lut_val*	Pointer to an array of 8192 unsigned long integers.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.25 SPDAU_DLL_API int SPDAU_set_anout_offset (unsigned long *offset_val,* unsigned long *chan*)

int SPDAU_set_anout_offset(unsigned long offset_val,unsigned long chan);

Sets the offset value for a selected analog output channel.

Parameters

in	*offset_val*	binary 14 bit value loaded to the digital to analog converter that controls the offset DC voltage of the selected analog output.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.26 SPDAU_DLL_API int SPDAU_get_anout_offset (unsigned long * *offset_val*, unsigned long *chan*)

int SPDAU_get_anout_offset(unsigned long *offset_val,unsigned long chan);

Requests the offset value for a selected analog output channel.

Parameters

out	*offset_val*	binary 14 bit value loaded to the digital to analog converter that controls the offset DC voltage of the selected analog output.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.27 SPDAU_DLL_API int SPDAU_set_anout_button (unsigned long *button_val*, unsigned long *chan*)

int SPDAU_set_anout_button(unsigned long button_val,unsigned long chan);

This function allows the user to assign the control of the deployment of a analog output signal to the hardware push buttons present in SPDAU hardware front panel.

Parameters

in	*button_val*	Posible values • 0 : SPDAU Harware button has no influence on selected analog output signal. • 1 : SPDAU Harware button triggers the deployment of the selected analog output signal.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.28 SPDAU_DLL_API int SPDAU_get_anout_button (unsigned long * *button_val*, unsigned long *chan*)

int SPDAU_get_anout_button(unsigned long *button_val,unsigned long chan);

This function returns the assignment value of the control variable that links the analog output signal deployment to the hardware push buttons present in SPDAU hardware front panel.

Parameters

out	*button_val*	Posible values • 0 : SPDAU Harware button has no influence on selected analog output signal. • 1 : SPDAU Harware button triggers the deployment of the selected analog output signal.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.29 SPDAU_DLL_API int SPDAU_set_anout_attenuation (double *attenuation_val,* unsigned long *chan*)

int SPDAU_set_anout_attenuation(double attenuation_val,unsigned long chan);

This functions sets an attenuation factor that will be used to multiply the values loaded to the digital to analog converter selected by input parameter chan.

Parameters

in	*attenuation_val*	multiplication factor used to ponderate the values loaded to the Digital to analog converter selected by chan input parameter. This value should range between 0 and 1.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.30 SPDAU_DLL_API int SPDAU_get_anout_attenuation (double * *attenuation_val,* unsigned long *chan*)

int SPDAU_set_anout_attenuation(double *attenuation_val,unsigned long chan);

This functions requests the attenuation factor that will be used to multiply the values

loaded to the digital to analog converter selected by input parameter chan.

Parameters

in	*attenuation_val*	multiplication factor used to ponderate the values loaded to the Digital to analog converter.
in	*chan*	Selects the analog output channel. • 0 : ANOUT1 • 1 : ANOUT2

Returns

1 on success, otherwise -1

2.4.2.31 SPDAU_DLL_API int SPDAU_set_offsetaux (unsigned long *offset_val,* unsigned long *chan*)

int SPDAU_set_offsetaux(unsigned long offset_val,unsigned long chan);

This functions sets the value to be loaded to the auxiliary offset digital to analog converter selected by input parameter chan. The offset value must be coded in binary 14 bit value.

Parameters

in	*offset_val*	Binary value to be loaded to the auxiliary offset analog output channel selected by "chan" input parameter.
in	*chan*	Selects the offset auxiliary channel. • 0 : OFFSET_OUT1 • 1 : OFFSET_OUT2

Returns

1 on success, otherwise -1

2.4.2.32 SPDAU_DLL_API int SPDAU_get_offsetaux (unsigned long * *offset_val,* unsigned long *chan*)

int SPDAU_set_offsetaux(unsigned long *offset_val,unsigned long chan);

This functions requests the value to be loaded to the auxiliary offset digital to analog converter selected by input parameter "chan" is OFFSET_OUT1 or OFFSET_OUT2. In this case The offset value is coded in binary 14 bit value. When "chan" input parameter selects OFFSET_IN1, the return value will be an auxiliary offset voltage comming from a 20 bit Analog to digital converter.

Parameters

out	*offset_val*	• when chan==OFFSET_OUT1 14 bit Binary value to be loaded to the auxiliary offset DAC 1. • when chan==OFFSET_OUT2 14 bit Binary value to be loaded to the auxiliary offset DAC 2. • when chan==OFFSET_IN1 20 bit Binary value read from auxiliary offset ADC AUX1.
in	*chan*	Selects the offset auxiliary channel. • 0 : OFFSET_OUT1 (14 bit DAC) • 1 : OFFSET_OUT2 (14 bit DAC) • 2 : OFFSET_IN1 (20 bit ADC)

Returns

1 on success, otherwise -1

2.5 SPDAU calibration

Functions

- SPDAU_DLL_API int SPDAU_set_cal_table (double *cal_table)

  Loads the DAC/ADC calibration table to SPDAU.

- SPDAU_DLL_API int SPDAU_get_cal_table (double *cal_table)

  Requests the DAC/ADC calibration table from SPDAU hardware.

2.5.1 Detailed Description

Functions that sets conversion constants from ADC and DAC values to physical values.

2.5.2 Function Documentation

2.5.2.1 SPDAU_DLL_API int SPDAU_set_cal_table (double * *cal_table*)

int SPDAU_set_cal_table(double *cal_table);

This functions loads to SPDAU a floating point array containing the conversion constants used to convert de ADC and DAC binary values to voltage values. The table is a group of 7 pairs of constants. The table is formed as follows:

- Analog input : BIT_WEIGHT, OFFSET_ERROR

- Analog output1 : BIT_WEIGHT, OFFSET_ERROR
- Analog output2 : BIT_WEIGHT, OFFSET_ERROR
- Offset AuxOut1 : BIT_WEIGHT, OFFSET_ERROR
- Offset AuxOut2 : BIT_WEIGHT, OFFSET_ERROR
- Offset AuxIn1 : BIT_WEIGHT, OFFSET_ERROR
- Analog Out1-2 OFFSET : BIT_WEIGHT1, BIT_WEIGHT2

  Where BIT_WEIGHT and OFFSET_ERROR are in volts. A voltage value from given analog channel shall be calculated using the following formula:

$$VoltageAnChan = (BinaryValue * BIT_WEIGHT) + OFFSET_ERROR;$$

Parameters

in	*cal_table*	Pointer to a calibration table containing 14 floating point numbers. 7 pairs of bit weights and offset correction constants.

Returns

1 on success, otherwise -1

2.5.2.2 SPDAU_DLL_API int SPDAU_get_cal_table (double * *cal_table*)

int SPDAU_get_cal_table(double *cal_table);

This functions requests SPDAU a floating point array containing the conversion constants used to convert de ADC and DAC binary values to voltage values.

Parameters

out	*cal_table*	Pointer to a calibration table containing 14 floating point numbers. 7 pairs of bit weights and offset correction constants.

Returns

1 on success, otherwise -1

2.6 SPDAU FPGA access

Functions

- SPDAU_DLL_API int SPDAU_set_fpga_register (unsigned long reg_address, unsigned long reg_data)

  Write access to a FPGA register.

- SPDAU_DLL_API int SPDAU_get_fpga_register (unsigned long reg_address, unsigned long *reg_data)

Read access to a FPGA register.

2.6.1 Detailed Description

Functions that allows PC host software to access SPDAU FPGA internal registers.

2.6.2 Function Documentation

2.6.2.1 SPDAU_DLL_API int SPDAU_set_fpga_register (unsigned long *reg_address*, unsigned long *reg_data*)

int SPDAU_set_fpga_register(unsigned long reg_address,unsigned long reg_data);

Write the value reg_data to the FPGA register addressed by reg_address.

Parameters

in	*reg_address*	FPGA register address.
in	*reg_data*	FPGA register value to be written.

Returns

1 on success, otherwise -1

2.6.2.2 SPDAU_DLL_API int SPDAU_get_fpga_register (unsigned long *reg_address*, unsigned long * *reg_data*)

int SPDAU_get_fpga_register(unsigned long reg_address,unsigned long *reg_data);

Reads the value from the FPGA register addressed by reg_address.

Parameters

in	*reg_address*	FPGA register address.
out	*reg_data*	FPGA register value.

Returns

1 on success, otherwise -1

2.7 SPDAU control

Functions

- SPDAU_DLL_API int SPDAU_arm (void)

  Sets system in armed mode.

- SPDAU_DLL_API int SPDAU_record (void)
  Sets system in recording mode.
- SPDAU_DLL_API int SPDAU_disarm (void)
  Sets system in iddle state.
- SPDAU_DLL_API int SPDAU_deploy_outputs (void)
  Updates signal generator enable flags.
- SPDAU_DLL_API int SPDAU_get_adc_aux (unsigned long *adc_val)
  Requests the present value of the auxiliary ADC of SPDAU.

2.7.1 Detailed Description

Functions that control system task execution.

2.7.2 Function Documentation

2.7.2.1 SPDAU_DLL_API int SPDAU_arm (void)

int SPDAU_arm(void);

Sets the system in armed mode waiting for a trigger event.

Returns

1 on success otherwise -1

2.7.2.2 SPDAU_DLL_API int SPDAU_record (void)

int SPDAU_record(void);

Sets the remote hardware in recording mode, i.e. deploys internal trigger signal.

Returns

1 on success otherwise -1

2.7.2.3 SPDAU_DLL_API int SPDAU_disarm (void)

int SPDAU_disarm(void);

Sets the remote hardware in iddle state., i.e. cancel recording tasks.

Returns

1 on success otherwise -1

2.7.2.4 SPDAU_DLL_API int SPDAU_deploy_outputs (void)

int SPDAU_deploy_outputs(void)

Updates the enable flags of analog and digital outputs. Intended to assure synchronism between different output signals.

Returns

1 on success otherwise -1

2.7.2.5 SPDAU_DLL_API int SPDAU_get_adc_aux (unsigned long * *adc_val*)

int SPDAU_get_adc_aux(unsigned long * adc_val)

Requests the present value delivered by the auxiliary analog to digital converter of SPDAU. The value is a two complemnte

Parameters

out	*adc_val*	20 bits value read directly from the analog to digital converter.

Returns

1 on success otherwise -1

2.8 Data transfer

Functions

- SPDAU_DLL_API int SPDAU_get_data (unsigned long offset, unsigned long *data, unsigned long nsamples)

  Fetches a chunk of data.

- SPDAU_DLL_API int SPDAU_save_data2file (unsigned long offset, unsigned long nsamples, char *filename)

  Fetches a chunk of data and stores it to a file.

2.8.1 Detailed Description

Functions fetch measurement data from remote SPDAU hardware.

2.8.2 Function Documentation

2.8.2.1 SPDAU_DLL_API int SPDAU_get_data (unsigned long *offset*, unsigned long * *data*, unsigned long *nsamples*)

int SPDAU_get_data(unsigned long offset,unsigned long *data,unsigned long nsamples);

Fetches a chunk of data from remote logger memory starting at sample indexed by input parameter "offset". The number of samples to be read is specified by "nsamples" input paramters. The maximum number of samples is 8192 per chunk in one function call. When targeting bigger than 8192 samples data chunks, the host application must concatenate the data chunks accordingly to user requirements.

Parameters

in	*offset*	Sample number of the first sample in the data chunk. The offset is relative to the beggining of the pretrigger memory area.
out	*data*	Measurment data coded in unsigned 32bit words.
in	*nsamples*	Number of samples requested, maximum is 8192.

Returns

1 on success otherwise -1

2.8.2.2 SPDAU_DLL_API int SPDAU_save_data2file (unsigned long *offset*, unsigned long *nsamples*, char * *filename*)

int SPDAU_save_data2file(unsigned long offset,unsigned long nsamples,char *filename)

Fetches a chunk of data from remote logger memory starting at sample indexed by input parameter "offset" and stores it in PC-host Harddisk with a given filename. The number of samples to be read is specified by "nsamples" input paramters. The data is stored in binary format into the file, unsigned 32 bit words.

Parameters

in	*offset*	Sample number of the first sample in the data chunk. The offset is relative to the beggining of the pretrigger memory area.
in	*nsamples*	Number of samples requested.
in	*filename*	Filename where the data shall be stored in host computer

Returns

1 on success otherwise -1

2.9 SPDAU data read out FIR filter

Functions

- SPDAU_DLL_API int SPDAU_FIR_set_decimation_factor (unsigned long factor)

  Sets data read out decimation factor.

- SPDAU_DLL_API int SPDAU_FIR_get_decimation_factor (unsigned long *factor)

  Gets data read out decimation factor.

- SPDAU_DLL_API int SPDAU_FIR_set_param (unsigned long *FIR_param)

  Sets FIR parameters on remote filter design tool.

- SPDAU_DLL_API int SPDAU_FIR_get_param (unsigned long *FIR_param)

  Gets FIR parameters on remote filter design tool.

- SPDAU_DLL_API int SPDAU_FIR_set_coeff (unsigned long *FIR_coeff, int ntaps)

  Sets SPDAU FIR coefficients.

- SPDAU_DLL_API int SPDAU_FIR_get_coeff (unsigned long *FIR_coeff, int *ntaps)

  Gets SPDAU FIR coefficients.

- SPDAU_DLL_API int SPDAU_FIR_open_pipe (unsigned long first_sample)

  Initialises SPDAU FIR filter data pipe.

- SPDAU_DLL_API int SPDAU_FIR_get_pipe_data (unsigned long *data, unsigned long nsamples)

  Reads a chunk of data from the FIR output data stream.

- SPDAU_DLL_API int SPDAU_FIR_get_impulse (unsigned long *data, unsigned long nsamples)

  Executes the FIR filter coefficients and returns its impulse response.

- SPDAU_DLL_API int SPDAU_FIR_set_type (unsigned long FIR_type)

  Sets the FIR filter type to be designed.

- SPDAU_DLL_API int SPDAU_FIR_get_type (unsigned long *FIR_type)

  Gets the FIR filter type used by the embedded FIR filter design tool.

- SPDAU_DLL_API int SPDAU_FIR_set_ampfactor (unsigned long ampfactor)

  Sets the amplification factor at FIR filter output stage.

- SPDAU_DLL_API int SPDAU_FIR_get_ampfactor (unsigned long *ampfactor)

  Gets the ampliflication factor at FIR filter output stage.

2.9.1 Detailed Description

Functions that configure FIR and decimation during data read out.

2.9.2 Function Documentation

2.9.2.1 SPDAU_DLL_API int SPDAU_FIR_set_decimation_factor (unsigned long *factor*)

int SPDAU_FIR_set_decimation_factor(unsigned long factor);

Sets the decimation factor applied to the signal at the output of the FIR filter.

Parameters

in	*factor*	Decimation factor. Defines the rate at which the data is read at the output of the FIR filter.

Returns

1 on success otherwise -1

2.9.2.2 SPDAU_DLL_API int SPDAU_FIR_get_decimation_factor (unsigned long * *factor*)

int SPDAU_FIR_get_decimation_factor(unsigned long * factor);

gets the decimation factor applied to the signal at the output of the FIR filter.

Parameters

out	*factor*	Decimation factor. Returns the rate at which the data is read at the output of the FIR filter.

Returns

1 on success otherwise -1

2.9.2.3 SPDAU_DLL_API int SPDAU_FIR_set_param (unsigned long * *FIR_param*)

int SPDAU_FIR_set_param(unsigned long * FIR_param);

Sets FIR parameters so the filter design tool integrated in SPDAU generates the corresponding coefficients and updates the FIR coefficient memory. This method requires as input parameters the cut frequencies and number of filter taps. Depending on the filter type set by SPDAU_FIR_set_type, input parameter array has different meanings.

- If the filter type is a low pass or high pass:
- Position 0 of the array is assigned to the filter cut frequency expresed in KHz.
- Position 1 of the array is assigned to the number of filter taps with a maximum of 1023 taps.
- If the filter type is a band pass or band stop:
- Position 0 of the array is assigned to the first filter cut frequency expresed in KHz.
- Position 1 of the array is assigned to the second filter cut frequency expresed in KHz.
- Position 2 of the array is assigned to the number of filter taps with a maximum of 1023 taps.

Parameters

in	*FIR_param*	Wrapper data array for filter parameters used by SPDAU filter tool to calculate the filter coefficients.

Returns

1 on success otherwise -1

2.9.2.4 SPDAU_DLL_API int SPDAU_FIR_get_param (unsigned long * *FIR_param*)

int SPDAU_FIR_get_param(unsigned long * FIR_param);

Gets the FIR parameters used by SPDAU filter design tool to calculate filter coefficients. The filter parameters are wrapped in an unsigned long data array .

- If the filter type is a low pass or high pass:
- Position 0 of the array is assigned to the filter cut frequency expresed in KHz.
- Position 1 of the array is assigned to the number of filter taps with a maximum of 1023 taps.
- If the filter type is a band pass or band stop:
- Position 0 of the array is assigned to the first filter cut frequency expresed in KHz.
- Position 1 of the array is assigned to the second filter cut frequency expresed in KHz.
- Position 2 of the array is assigned to the number of filter taps with a maximum of 1023 taps.

Parameters

out	*FIR_param*	Wrapper data array for filter parameters used by SPDAU filter tool to calculate the filter coefficients.

Returns

1 on success otherwise -1

2.9.2.5 SPDAU_DLL_API int SPDAU_FIR_set_coeff (unsigned long * *FIR_coeff,* int *ntaps*)

int SPDAU_FIR_set_coeff(unsigned long * FIR_coeff, int ntaps);

SPDAU integrates a configurable FIR structure. This function initialises the FIR filter coefficient memory with the data contained in FIR_coeff data array. Up to 1024 coefficients coded in 18 bit signed words can be downloaded into the FIR memory. The FIR structure will execute only the number of taps configure by ntaps input parameter.

Parameters

in	*FIR_coeff*	FIR filter coefficient memory. Coefficients must be coded in 18 bit signed integers.
in	*ntaps*	FIR filter number of taps. must be a number range 1 to 1024.

Returns

1 on success otherwise -1

2.9.2.6 SPDAU_DLL_API int SPDAU_FIR_get_coeff (unsigned long * *FIR_coeff,* int * *ntaps*)

int SPDAU_FIR_get_coeff(unsigned long * FIR_coeff, int ntaps);

Returns the FIR coefficient memory block and the number of filter taps programmed.

Parameters

out	*FIR_coeff*	FIR filter coefficient memory. Coefficients coded in 18 bit signed integers.
out	*ntaps*	FIR filter number of taps

Returns

1 on success otherwise -1

2.9.2.7 SPDAU_DLL_API int SPDAU_FIR_open_pipe (unsigned long *first_sample*)

int SPDAU_FIR_open_pipe(unsigned long first_sample);

Initialises SPDAU FIR filter data pipe. Clears FIR data memory and sets the index of the first position of the captured data to be filtered.

Parameters

in	*first_sample*	Position index of the first sample of the data to be filtered.

Returns

1 on success otherwise -1

2.9.2.8 SPDAU_DLL_API int SPDAU_FIR_get_pipe_data (unsigned long * *data*, unsigned long *nsamples*)

int SPDAU_FIR_get_pipe_data(unsigned long *data,unsigned long nsamples);

Returns a chunk of data from the FIR output data stream. The number of items must be specified. Calling this function in a loop returns consecutive data chunks.

Parameters

out	*data*	FIR filter data output.
in	*nsamples*	Number of samples to be read from the FIR output data stream.

Returns

1 on success otherwise -1

2.9.2.9 SPDAU_DLL_API int SPDAU_FIR_get_impulse (unsigned long * *data*, unsigned long *nsamples*)

int SPDAU_FIR_get_impulse(unsigned long *data,unsigned long nsamples);

Executes the FIR filter coefficients and returns its impulse response. The lenght of the impulse response array must be specified. The maximum lenght of the impulse response vector is 8192 points

Parameters

out	*data*	FIR filter impulse response array.
in	*nsamples*	Number of samples to be read from the FIR output data stream.

Returns

1 on success otherwise -1

2.9.2.10 SPDAU_DLL_API int SPDAU_FIR_set_type (unsigned long *FIR_type*)

int SPDAU_FIR_set_type(unsigned long FIR_type);

Sets the FIR filter type to be designed. This function must be called before calling SPDAU_FIR_set_param whose input parameter meaning will change depending in the selected filter.

Parameters

in	*FIR_type*	Filter type to be designed, must be one of the following values. If none of them is set the a low pass filter configuration will be used. • 0 : Low pass filter • 1 : High pass filter • 2 : Band pass filter • 3 : Band Stop filter

Returns

1 on success otherwise -1

2.9.2.11 SPDAU_DLL_API int SPDAU_FIR_get_type (unsigned long * *FIR_type*)

int SPDAU_FIR_get_type(unsigned long *FIR_type);

Gets the FIR filter type used by the embedded FIR filter design tool.

Parameters

out	*FIR_type*	Filter type used by the embedded FIR filter design tool, must be one of the following values. • 0 : Low pass filter • 1 : High pass filter • 2 : Band pass filter • 3 : Band Stop filter

Returns

1 on success otherwise -1

2.9.2.12 SPDAU_DLL_API int SPDAU_FIR_set_ampfactor (unsigned long *ampfactor*)

int SPDAU_FIR_set_ampfactor(unsigned long ampfactor);

FPGA FIR filter accumulator is 60 bits wide and the output of the FIR filter is 32 bits wide. The FIR amlification factor selects the bit range assigned to the FIR output vector. The amplification can be configured with the following values:

- 0 : FIR output is ACCUM_VALUE(40 downto 9); Equivalent gain factor of 1/16
- 1 : FIR output is ACCUM_VALUE(41 downto 10); Equivalent gain factor of 1/8
- 2 : FIR output is ACCUM_VALUE(42 downto 11); Equivalent gain factor of 1/4
- 3 : FIR output is ACCUM_VALUE(43 downto 12); Equivalent gain factor of 1/2

- 4 : FIR output is ACCUM_VALUE(44 downto 13); Equivalent gain factor of 1
- 5 : FIR output is ACCUM_VALUE(45 downto 14); Equivalent gain factor of 2
- 6 : FIR output is ACCUM_VALUE(46 downto 15); Equivalent gain factor of 4
- 7 : FIR output is ACCUM_VALUE(47 downto 16); Equivalent gain factor of 8
- 8 : FIR output is ACCUM_VALUE(48 downto 17); Equivalent gain factor of 16
- 9 : FIR output is ACCUM_VALUE(49 downto 18); Equivalent gain factor of 32
- 10 : FIR output is ACCUM_VALUE(50 downto 19); Equivalent gain factor of 64
- 11 : FIR output is ACCUM_VALUE(51 downto 20); Equivalent gain factor of 128
- 12 : FIR output is ACCUM_VALUE(52 downto 21); Equivalent gain factor of 256
- 13 : FIR output is ACCUM_VALUE(53 downto 22); Equivalent gain factor of 512
- 14 : FIR output is ACCUM_VALUE(54 downto 23); Equivalent gain factor of 1024
- 15 : FIR output is ACCUM_VALUE(55 downto 24); Equivalent gain factor of 2048

Parameters

in	*ampfactor*	FPGA FIR filter out gain factor, in range 0 to 15

Returns

1 on success otherwise -1

2.9.2.13 SPDAU_DLL_API int SPDAU_FIR_get_ampfactor (unsigned long * *ampfactor*)

int SPDAU_FIR_get_ampfactor(unsigned long *ampfactor);

FPGA FIR filter accumulator is 60 bits wide and the output of the FIR filter is 32 bits wide. The FIR amlification factor selects the bit range assigned to the FIR output vector. The amplification can be configured with values enumerated in previous section.

Parameters

out	*ampfactor*	FPGA FIR filter out gain factor, in range 0 to 15

Returns

1 on success otherwise -1

Index