OpenLab - XMC4500

Generated by Doxygen 1.8.11

Contents

Chapter 1

Deprecated List

Global OLB_PROT_CODE_SD

Use OLB_PROT_CODE_ESD instead.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

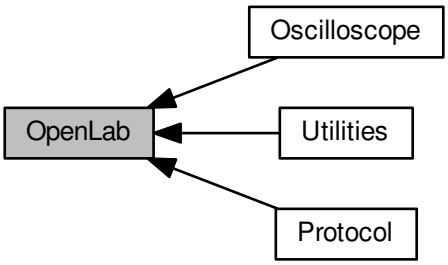
Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 OpenLab

Collaboration diagram for OpenLab:



Modules

- Utilities
- Oscilloscope
- Protocol

Enumerations

- enum Olb_error_t {
  OLB_EOK = 0, OLB_ETOUT = -1, OLB_EWARNING = -2, OLB_ENOT_SUPPORTED = -3,
  OLB_EUNKNOWN_COMMAND = -4, OLB_EFAIL = -5, OLB_EOVERFLOW = -6, OLB_EUNDERFLOW = -7,
  OLB_EAGAIN = -8 }

Functions

- void olb_init (void)
- void olb_task (void)
- void olb_wait (uint32_t wait_time)
- uint16_t olb_receive (uint8_t *data, uint16_t count)
- uint16_t olb_send (uint8_t *data, uint16_t count)

5.1.1 Detailed Description

Implementation of the OpenLab protocol and devices on the XMC4500 Relax Kit development platform.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum Olb_error_t

Errors that may be returned by OpenLab functions

Enumerator

OLB_EOK Successful, no error detected.

OLB_ETOUT Operation timed out.

OLB_EWARNING General warning. Outcome of the operation not defined.

OLB_ENOT_SUPPORTED Requested operation is not supported.

OLB_EUNKNOWN_COMMAND Command not known.

OLB_EFAIL Operation failed to unknown reason.

OLB_EOVERFLOW Operation failed due to memory overflow.

OLB_EUNDERFLOW Operation failed due to lack of memory.

OLB_EAGAIN Operation could not complete yet, try again another time.

5.1.3 Function Documentation

5.1.3.1 void olb_init (void)

Initializes the OpenLab library

This function needs to be called by the user application before any other call to olb_* functions.

5.1.3.2 uint16_t olb_receive (uint8_t * *data*, uint16_t *count*)

Receive data from the default communication interface which is USB virtual COM.

Parameters

data	Pointer to destination data.
count	Number of bytes to receive.

Returns

Returns actual number of bytes received.

5.1.3.3 uint16_t olb_send (uint8_t * *data*, uint16_t *count*)

Sends data to the default communication interface which is USB virtual COM.

Parameters

data	Pointer to source data.
count	Number of bytes to send.

Returns

Actual number of bytes sent.

5.1.3.4 void olb_task (void)

The main task loop function of the OpenLab library. This function must be called periodically to keep the library alive.

5.1.3.5 void olb_wait (uint32_t *wait_time*)

Waits some time before continuing execution.

Parameters

wait_time	Wait time in micro seconds.

Note

Minimum time resolution is 10 micro seconds.

5.2 Utilities

Collaboration diagram for Utilities:

Modules

- Fifo8

5.2.1 Detailed Description

5.3 Fifo8

Collaboration diagram for Fifo8:

Data Structures

- struct Olb_fifo8_t

Functions

- void olb_fifo8_init (Olb_fifo8_t *const fifo, uint8_t *const mem, uint32_t const size)
- uint32_t olb_fifo8_push (Olb_fifo8_t *const fifo, uint8_t const data)
- uint32_t olb_fifo8_pop (Olb_fifo8_t *const fifo, uint8_t *const data)
- uint8_t * olb_fifo8_get_mem (Olb_fifo8_t *const fifo)
- void olb_fifo8_clear (Olb_fifo8_t *const fifo)
- uint32_t olb_fifo8_count (Olb_fifo8_t *const fifo)

5.3.1 Detailed Description

5.3.2 Function Documentation

5.3.2.1 void olb_fifo8_clear (Olb_fifo8_t *const *fifo*)

Clear the FIFO and resets the state variables except the memory pointer.

Parameters

fifo	Pointer to Olb_fifo8_t object.

5.3.2.2 uint32_t olb_fifo8_count (Olb_fifo8_t *const *fifo*)

Returns the current number of bytes stored in the FIFO.

Parameters

fifo	Pointer to Olb_fifo8_t object.

Returns

Number of bytes currently stored in the FIFO.

5.3.2.3 uint8_t* olb_fifo8_get_mem (Olb_fifo8_t *const *fifo*)

Returns a pointer to the internal data memory.

Parameters

fifo	Pointer to Olb_fifo8_t object.

Returns

Pointer to byte buffer.

Note

Check if the pointer may be valid.

5.3.2.4 void olb_fifo8_init (Olb_fifo8_t *const *fifo*, uint8_t *const *mem*, uint32_t const *size*)

Initializes a byte FIFO (first in first out) buffer.

Parameters

fifo	Pointer to Olb_fifo8_t object.
mem	Pointer to memory area that should be used to the internal FIFO buffer.
size	Size of this memory in bytes.

5.3.2.5 uint32_t olb_fifo8_pop (Olb_fifo8_t *const *fifo*, uint8_t *const *data*)

Pops (removes) one byte from the FIFO.

Parameters

fifo	Pointer to Olb_fifo8_t object.
data	Pointer destination byte

Returns

Returns 1 if one byte has been popped, else 0.

5.3.2.6 uint32_t olb_fifo8_push (Olb_fifo8_t *const *fifo*, uint8_t const *data*)

Pushes (adds) one byte of data into the FIFO.

Parameters

fifo	Pointer to Olb_fifo8_t object.
data	Data byte to push

Returns

Returns 1 if successfully pushed, else 0.

5.4 Oscilloscope

Collaboration diagram for Oscilloscope:

Modules

- HoldOff

Data Structures

- struct Olb_osci_trig_cfg_t
- struct Olb_osci_chan_input_cfg_t
- struct Olb_osci_sampl_mode_cfg_t

Macros

- #define **OSCI_CHAN_0_QUEUE_TRIG_SIG** XMC_VADC_REQ_TR_A
- #define **OSCI_CHAN_0_VADC** VADC
- #define **OSCI_CHAN_0_VADC_GROUP** VADC_G0
- #define **OSCI_CHAN_0_VADC_CHANNEL** 0
- #define **OSCI_CHAN_0_VADC_RESULT_REG** 7
- #define **OSCI_CHAN_0_VADC_IRQn** VADC0_G0_0_IRQn
- #define **OSCI_CHAN_0_VADC_IRQ_HANDLER** VADC0_G0_0_IRQHandler
- #define **OSCI_CHAN_0_NVIC_PRIO** 0
- #define **OSCI_CHAN_0_GPIO_PORT** PORT14
- #define **OSCI_CHAN_0_GPIO_PIN** 0
- #define **OSCI_CHAN_0_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_MASTER_GROUP_NUM** 0
- #define **OSCI_CHAN_MASTER_GROUP_IND** 0
- #define **OSCI_CHAN_0_CCU4** CCU40
- #define **OSCI_CHAN_0_CCU4_SLICE** CCU40_CC40
- #define **OSCI_CHAN_0_CCU4_SLICE_NUM** 0
- #define **OSCI_CHAN_0_CCU4_SHADOW_TRANSFER_SLICE** (XMC_CCU4_SHADOW_TRANSFER_S↩ LICE_0 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_0)
- #define **OSCI_CHAN_0_CCU4_SLICE_INPUT** XMC_CCU4_SLICE_INPUT_C
- #define **OSCI_CHAN_0_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_2
- #define **OSCI_CHAN_0_CCU4_SLICE_NVIC_IRQN** CCU40_2_IRQn
- #define **OSCI_CHAN_0_CCU4_SLICE_NVIC_IRQ_HDLR** CCU40_2_IRQHandler
- #define **OSCI_CHAN_0_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_CHAN_0_CCU4_SLICE_START_EVT_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_EVENT0
- #define **OSCI_CHAN_0_CCU4_SLICE_START_EVT_SR_ID** XMC_CCU4_SLICE_SR_ID_3

- #define **OSCI_CHAN_0_CCU4_SLICE_START_NVIC_IRQN** CCU40_3_IRQn
- #define **OSCI_CHAN_0_CCU4_SLICE_START_NVIC_IRQ_HDLR** CCU40_3_IRQHandler
- #define **OSCI_CHAN_0_TRIG_GPIO_PORT** PORT2
- #define **OSCI_CHAN_0_TRIG_GPIO_PIN** 1
- #define **OSCI_CHAN_0_TRIG_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_1_QUEUE_TRIG_SIG** XMC_VADC_REQ_TR_F
- #define **OSCI_CHAN_1_VADC** VADC
- #define **OSCI_CHAN_1_VADC_GROUP** VADC_G2
- #define **OSCI_CHAN_1_VADC_CHANNEL** 0
- #define **OSCI_CHAN_1_VADC_RESULT_REG** 7
- #define **OSCI_CHAN_1_VADC_IRQn** VADC0_G2_0_IRQn
- #define **OSCI_CHAN_1_VADC_IRQ_HANDLER** VADC0_G2_0_IRQHandler
- #define **OSCI_CHAN_1_NVIC_PRIO** 0
- #define **OSCI_CHAN_1_GPIO_PORT** PORT14
- #define **OSCI_CHAN_1_GPIO_PIN** 4
- #define **OSCI_CHAN_1_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_SLAVE_GROUP_NUM** 2
- #define **OSCI_CHAN_SLAVE_GROUP_IND** 1
- #define **OSCI_CHAN_1_CCU4** CCU43
- #define **OSCI_CHAN_1_CCU4_SLICE** CCU43_CC40
- #define **OSCI_CHAN_1_CCU4_SLICE_NUM** 0
- #define **OSCI_CHAN_1_CCU4_SHADOW_TRANSFER_SLICE** (XMC_CCU4_SHADOW_TRANSFER_S↩ LICE_0 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_0)
- #define **OSCI_CHAN_1_CCU4_SLICE_INPUT** XMC_CCU4_SLICE_INPUT_B
- #define **OSCI_CHAN_1_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_3
- #define **OSCI_CHAN_1_CCU4_SLICE_NVIC_IRQN** CCU43_3_IRQn
- #define **OSCI_CHAN_1_CCU4_SLICE_NVIC_IRQ_HDLR** CCU43_3_IRQHandler
- #define **OSCI_CHAN_1_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_CHAN_1_CCU4_SLICE_START_EVT_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_EVENT0
- #define **OSCI_CHAN_1_CCU4_SLICE_START_EVT_SR_ID** XMC_CCU4_SLICE_SR_ID_1
- #define **OSCI_CHAN_1_CCU4_SLICE_START_NVIC_IRQN** CCU43_1_IRQn
- #define **OSCI_CHAN_1_CCU4_SLICE_START_NVIC_IRQ_HDLR** CCU43_1_IRQHandler
- #define **OSCI_CHAN_1_TRIG_GPIO_PORT** PORT2
- #define **OSCI_CHAN_1_TRIG_GPIO_PIN** 14
- #define **OSCI_CHAN_1_TRIG_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU** CCU80
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE** CCU80_CC81
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_NUM** 1
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_SHADOW_TRANSFER** XMC_CCU8_SHADOW_T↩ RANSFER_SLICE_1
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_COMPARE_CHANNEL** XMC_CCU8_SLICE_COM↩ PARE_CHANNEL_2
- #define **OLB_OSCI_TRIG_PWM_CH0_FREQ** 100000
- #define **OLB_OSCI_TRIG_PWM_CH0_PORT** PORT0
- #define **OLB_OSCI_TRIG_PWM_CH0_PIN** 9
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU** CCU80
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE** CCU80_CC80
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_NUM** 0
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_SHADOW_TRANSFER** XMC_CCU8_SHADOW_T↩ RANSFER_SLICE_0
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_COMPARE_CHANNEL** XMC_CCU8_SLICE_COM↩ PARE_CHANNEL_2
- #define **OLB_OSCI_TRIG_PWM_CH1_FREQ** 100000
- #define **OLB_OSCI_TRIG_PWM_CH1_PORT** PORT0
- #define **OLB_OSCI_TRIG_PWM_CH1_PIN** 10

- #define **OSCI_HOLDOFF_CCU4** CCU40
- #define **OSCI_HOLDOFF_CCU4_SLICE** CCU40_CC41
- #define **OSCI_HOLDOFF_CCU4_SLICE_NUM** 1
- #define **OSCI_HOLDOFF_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_1
- #define **OSCI_HOLDOFF_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_HOLDOFF_CCU4_SHADOW_TRANSFER** (XMC_CCU4_SHADOW_TRANSFER_SLICE_↩ 1 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_1)
- #define **OSCI_HOLDOFF_NVIC_IRQ_PRIO** 1
- #define **OSCI_HOLDOFF_NVIC_IRQN** CCU40_1_IRQn
- #define **OSCI_HOLDOFF_NVIC_IRQ_HDLR** CCU40_1_IRQHandler
- #define **OLB_OSCI_CH0_GAIN0_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN0_PIN** (0U)
- #define **OLB_OSCI_CH0_GAIN1_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN1_PIN** (1U)
- #define **OLB_OSCI_CH0_GAIN2_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN2_PIN** (2U)
- #define **OLB_OSCI_CH1_GAIN0_PORT** XMC_GPIO_PORT0
- #define **OLB_OSCI_CH1_GAIN0_PIN** (0U)
- #define **OLB_OSCI_CH1_GAIN1_PORT** XMC_GPIO_PORT0
- #define **OLB_OSCI_CH1_GAIN1_PIN** (1U)
- #define **OLB_OSCI_CH1_GAIN2_PORT** XMC_GPIO_PORT2
- #define **OLB_OSCI_CH1_GAIN2_PIN** (15U)
- #define **OLB_OSCI_1kHz_FREQ** 1000
- #define **OLB_OSCI_1kHz_CCU** CCU40
- #define **OLB_OSCI_1kHz_CCU_SLICE_NUM** 2
- #define **OLB_OSCI_1kHz_CCU_SLICE** CCU40_CC42
- #define **OLB_OSCI_1kHz_CCU_SHADOW_TRANSF** XMC_CCU4_SHADOW_TRANSFER_SLICE_2
- #define **OLB_OSCI_1kHz_PORT** XMC_GPIO_PORT2
- #define **OLB_OSCI_1kHz_PIN** (10)
- #define **OLB_OSCI_1kHz_NVIC_IRQn** CCU40_0_IRQn
- #define **OLB_OSCI_1kHz_ISR** CCU40_0_IRQHandler
- #define **OLB_OSCI_1kHz_IRQ_PRIO** 2

Enumerations

- enum Olb_osci_mode_t { OLB_OSCI_MODE_REALT, OLB_OSCI_MODE_SETS, OLB_OSCI_MODE_R↩ ETS }
- enum Olb_osci_coupling_t { OLB_OSCI_COUPLING_OFF = 0, OLB_OSCI_COUPLING_DC = 1, OLB_O↩ SCI_COUPLING_AC = 2 }
- enum Olb_osci_trig_mode_t { OLB_OSCI_TRIG_MODE_AUTO = 0, OLB_OSCI_TRIG_MODE_NORMAL = 1 }
- enum Olb_osci_trig_active_flank_t { OLB_OSCI_TRIG_ACT_FLANK_RISING = 0, OLB_OSCI_TRIG_AC↩ T_FLANK_FALLING = 1, OLB_OSCI_TRIG_ACT_FLANK_BOTH = 2, OLB_OSCI_TRIG_ACT_FLANK_N↩ ONE = 3 }
- enum Olb_osci_event_t { OLB_OSCI_EVT_SAMPLING_COMPLETE, OLB_OSCI_EVT_TRIGGER_DET↩ ECT }

Functions

- void olb_osci_raise_evt (Olb_osci_event_t evt, int32_t data)
- void olb_osci_init (void)
- Olb_error_t olb_osci_prot_pkt_hdl (Olb_protocol_packet_t *const pkt)
- Olb_error_t olb_osci_set_trigger_cfg (Olb_osci_trig_cfg_t *const cfg)
- Olb_error_t olb_osci_set_channel_input_cfg (uint8_t chan, Olb_osci_chan_input_cfg_t *const cfg)
- Olb_error_t olb_osci_set_sample_rate (uint32_t sps)
- Olb_error_t olb_osci_set_sampling_cfg (Olb_osci_sampl_mode_cfg_t *const cfg)
- Olb_error_t olb_osci_get_samples (uint8_t chan, uint32_t count, uint8_t *const dst)
- Olb_error_t olb_osci_start (void)
- Olb_error_t olb_osci_stop (void)
- void olb_osci_init_gain (void)
- Olb_error_t Olb_osci_set_gain (uint8_t const channel, uint8_t gain)
- void olb_osci_init_1kHz (void)
- Olb_error_t olb_osci_task (void)
- void olb_osci_wait (uint32_t t_ms)
- int32_t olb_osci_timeout_cb (void(*cb)(void), uint32_t const t_ms, uint32_t const is_single_shot)
- void **olb_osci_stop_timeout** (int32_t const id)
- void olb_osci_chan_init (void)
- Olb_error_t olb_osci_chan_set_sample_buffer (uint8_t const chan_num, uint32_t buf_size, uint8_t *const buf)
- Olb_error_t olb_osci_chan_set_sample_count (uint8_t const chan_num, uint32_t const count)
- Olb_error_t olb_osci_chan_disable (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_enable (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_has_samples (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_is_busy (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_is_enabled (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_abort (void)
- Olb_error_t olb_osci_chan_arm (uint32_t smpl_count)
- Olb_error_t olb_osci_chan_write_to_smpl_buf (uint8_t chan, uint8_t const smpl)
- Olb_error_t olb_osci_chan_reset_smpl_data (uint8_t chan)
- Olb_error_t olb_osci_chan_set_trig_src (uint8_t chan)
- void olb_osci_init_holdoff (void)
- Olb_error_t olb_osci_set_holdoff (uint32_t holdoff_time)
- Olb_error_t olb_osci_start_holdoff (void)
- Olb_error_t olb_osci_start_holdoff_cb (void(*cb_f)(void *), void *para)
- Olb_error_t olb_osci_wait_holdoff (void)
- Olb_error_t olb_osci_trig_init ()
- Olb_error_t olb_osci_trig_set_cfg (Olb_osci_trig_cfg_t *const cfg)
- Olb_error_t olb_osci_trig_set_lvl (uint8_t const num, uint8_t const lvl)
- Olb_error_t olb_osci_trig_set_sample_rate (uint32_t rate)
- Olb_error_t olb_osci_trig_arm (void)
- Olb_error_t olb_osci_trig_stop (void)
- Olb_error_t olb_osci_trig_set_smpl_mode (Olb_osci_sampl_mode_cfg_t *cfg)
- Olb_error_t **olb_osci_trig_force** (void)
- void **olb_osci_trig_incr_ets_offset** (void)
- void **olb_osci_trig_reset_ets_offset** (void)
- uint32_t **olb_osci_cfg_get_chan_count** (void)
- uint32_t **olb_osci_cfg_get_max_sample_count** (void)
- uint32_t **olb_osci_cfg_get_default_smpl_rate** (void)
- void olb_osci_tmr_global_init (void)

5.4.1 Detailed Description

This module implements all the functionality needed for a common, two channel oscilloscope that supports the OpenLab serial protocol.

5.4.2 Enumeration Type Documentation

5.4.2.1 enum olb_osci_coupling_t

Coupling modes for the analog input channels

Enumerator

OLB_OSCI_COUPLING_OFF Channel is disabled

OLB_OSCI_COUPLING_DC Channel is DC coupled

OLB_OSCI_COUPLING_AC Channel is AC coupled

5.4.2.2 enum olb_osci_event_t

Events that may be fired by OpenLab oscilloscope sub modules.

Enumerator

OLB_OSCI_EVT_SAMPLING_COMPLETE Conversion of all requested samples finished. When calling olb_osci_raise_evt pass the channel number in the data parameter.

OLB_OSCI_EVT_TRIGGER_DETECT OLB_OSCI_EVT_TRIGGER_DETECT. Trigger flank detected.

5.4.2.3 enum olb_osci_mode_t

Supported sampling modes.

Enumerator

OLB_OSCI_MODE_REALT Real time sampling

OLB_OSCI_MODE_SETS Sequential equivalent time sampling

OLB_OSCI_MODE_RETS Random equivalent time sampling

Note

This mode is currently not supported.

5.4.2.4 enum olb_osci_trig_active_flank_t

Active flank which will cause the oscilloscope start sampling.

Enumerator

OLB_OSCI_TRIG_ACT_FLANK_RISING Rising flank

OLB_OSCI_TRIG_ACT_FLANK_FALLING Falling flank

OLB_OSCI_TRIG_ACT_FLANK_BOTH Both flanks

OLB_OSCI_TRIG_ACT_FLANK_NONE Flank is to be ignored. Effectively disabling the trigger.

5.4.2.5 enum Olb_osci_trig_mode_t

Trigger modes

Enumerator

OLB_OSCI_TRIG_MODE_AUTO Automatic trigger detection in case the set trigger does not raise an trigger event within a certain time.

OLB_OSCI_TRIG_MODE_NORMAL Normal trigger. The oscilloscope will wait starting sampling until a trigger event has been detected.

5.4.3 Function Documentation

5.4.3.1 Olb_error_t olb_osci_chan_abort (void)

Immediately aborts sampling of all channel.

Returns

Returns OLB_EOK when channels are hold.

5.4.3.2 Olb_error_t olb_osci_chan_arm (uint32_t *smpl_count*)

Arms the channels to accept trigger events that will result in a conversion request.

Parameters

smpl_count	Number of samples that shall be converted.

5.4.3.3 Olb_error_t olb_osci_chan_disable (uint8_t const *chan_num*)

Disable a specific channel and stops acquiring sample data.

Parameters

chan_num	Channel number starting at zero.

Returns

Returns OLB_EOK on success.

5.4.3.4 Olb_error_t olb_osci_chan_enable (uint8_t const *chan_num*)

Enables a specific channel to acquire sample data.

Parameters

chan_num	Channel number starting at zero.

Returns

Returns OLB_EOK on success.

5.4.3.5 Olb_error_t olb_osci_chan_has_samples (uint8_t const *chan_num*)

Checks whether a channel has sample data available.

This is "all or nothing". If the channel has completed sampling it will return OLB_EOK. If it is inactive or has not acquired all samples it will return OLB_EAGAIN.

Parameters

chan_num	Channel number starting at zero.

Returns

Returns OLB_EOK if all required samples had been sampled for this particular channel. If the channel is valid but is busy it returns OLB_EAGAIN.

5.4.3.6 void olb_osci_chan_init (void)

Initialize the analog input channels.

5.4.3.7 Olb_error_t olb_osci_chan_is_busy (uint8_t const *chan_num*)

Checks if the channel is currently busy.

Parameters

chan_num	Channel number to check starting at zero.

Returns

Returns OLB_EOK if the channel is busy, else OLB_EFAIL.

5.4.3.8 Olb_error_t olb_osci_chan_is_enabled (uint8_t const *chan_num*)

Checks if the channel is currently enabled.

Parameters

chan_num	Channel number to check starting at zero.

Returns

Returns OLB_EOK if the channel is busy, else OLB_EFAIL.

5.4.3.9 Olb_error_t olb_osci_chan_reset_smpl_data (uint8_t *chan*)

Reset sample data counters

Parameters

chan	Channel number for which to reset sample data counters.

Returns

Returns OLB_EOK when done. May return OLB_ENOT_SUPPORTED if the requested channel is invalid.

5.4.3.10 Olb_error_t olb_osci_chan_set_sample_buffer (uint8_t const *chan_num*, uint32_t *buf_size*, uint8_t *const *const *buf*)

Sets the sample buffer to be used to store samples.

Parameters

chan_num	Channel number starting at zero.
buf_size	The size of the buffer in bytes.
buf	Pointer to buffer to use.

Returns

Returns OLB_EOK on success.

5.4.3.11 Olb_error_t olb_osci_chan_set_sample_count (uint8_t const *chan_num*, uint32_t const *count*)

Sets the number of samples to be acquired.

Parameters

chan_num	Channel number starting at zero.
count	Number of samples to collect.

Returns

Returns OLB_EOK on success.

5.4.3.12 Olb_error_t olb_osci_chan_set_trig_src (uint8_t *chan*)

Sets the channel number that shall activate the arbiter.

Parameters

chan	Channel number that shall be the trigger source.

Returns

Returns OLB_EOK when done. May return OLB_ENOT_SUPPORTED if the requested channel is invalid.

5.4.3.13 Olb_error_t olb_osci_chan_write_to_smpl_buf (uint8_t *chan,* uint8_t const *smpl*)

Write one sample to sample buffer buffer

Parameters

chan	Channel number
smpl	Sample data

Returns

returns if the sample has been written to the buffer otherwise OLB_EOVERFLOW when the buffer is full. OLB_ENOT_SUPPORTED will be returned if the channel number is invalid.

5.4.3.14 Olb_error_t olb_osci_get_samples (uint8_t *chan,* uint32_t *count,* uint8_t ∗const *dst*)

Tries to read available sample data from the selected channel.

Parameters

chan	Channel number starting at zero
count	Maximum number of samples to read
dst	Pointer to memory to write samples to.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.15 void olb_osci_init (void)

Initialize the oscilloscope of the OpenLab library.

5.4.3.16 void olb_osci_init_1kHz (void)

Initialize the 1kHz reference signal provided for probe adjustment.

5.4.3.17 void olb_osci_init_gain (void)

Initializes the gain control I/O pins.

5.4.3.18 void olb_osci_init_holdoff (void)

Initialize the hold-off timer module

5.4.3.19 Olb_error_t olb_osci_prot_pkt_hdl (Olb_protocol_packet_t *const *pkt*)

Processes an OpenLab protocol packet that has been addressed to the oscilloscope.

This function will decode the protocol head and payload if applicable. After checking the received data the will execute the command if it is valid and send back the corresponding packets. This function is primarily intended to act as a remote control for the Openlab oscilloscope. All of the oscilloscopes module functionality can also be used by calling the additional interface functions.

Parameters

pkt	Pointer to the received Olb_protocol_packet_t.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.20 void olb_osci_raise_evt (Olb_osci_event_t *evt*, int32_t *data*)

Call back function for sub-modules to inform the OpenLab oscilloscope task about certain events.

Parameters

evt	Event that just happened
data	Optional data that may be of interest.

5.4.3.21 Olb_error_t olb_osci_set_channel_input_cfg (uint8_t *chan,* Olb_osci_chan_input_cfg_t ∗const *cfg*)

Set the channel configuration to selected channel

Providing the preset channel configuration structure sets the gain as well as the coupling. By setting the coupling to OLB_OSCI_COUPLING_OFF the selected channel will get disabled.

Parameters

chan	Number of channel starting at zero.
cfg	Pointer to preset Olb_osci_chan_input_cfg_t

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.22 Olb_error_t Olb_osci_set_gain (uint8_t const *channel,* uint8_t *gain*)

Set the gain for the selected channel

Parameters

channel	Channel number starting at zero
gain	Gain setting, valid range 0-7.

Returns

Returns OLB_EOK on success, else some other value of Olb_error_t

5.4.3.23 Olb_error_t olb_osci_set_holdoff (uint32_t *holdoff_time*)

Sets the hold-off time.

To lower the latency for starting the hold-off timer the hold-off time can be set and stored with this function.

Parameters

holdoff_time	Hold-off time in micro seconds.

Returns

Will return OLB_EOK if hold-off time is valid.

5.4.3.24 Olb_error_t olb_osci_set_sample_rate (uint32_t *sps*)

Sets the sample rate of all channels. of the oscilloscope.

Parameters

sps	Samples per second

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.25 Olb_error_t olb_osci_set_sampling_cfg (Olb_osci_sampl_mode_cfg_t ∗const *cfg*)

Sets the requested sampling mode for all channels.

Parameters

cfg	Sampling mode configuration to be set. Pointer to a preset Olb_osci_sampl_mode_cfg_t object.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.26 Olb_error_t olb_osci_set_trigger_cfg (Olb_osci_trig_cfg_t ∗const *cfg*)

Set the requested trigger configuration

Parameters

cfg	Pointer to a preset Olb_osci_trig_cfg_t object.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.27 Olb_error_t olb_osci_start (void)

Starts the oscilloscope thus enables it to acquire sample data from its input channels.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.28 Olb_error_t olb_osci_start_holdoff (void)

Start the hold-off timer.

Check if the hold-off time has elapsed by calling olb_osci_wait_holdoff.

Returns

Returns OLB_EOK if hold-off has been started..

5.4.3.29 Olb_error_t olb_osci_start_holdoff_cb (void(*)(void *) *cb_f*, void * *para*)

Starts the hold-off timer and passes a call back function that shall called when the time has elapsed.

Parameters

cb_f	Call back function to call at end of hold-off time.
para	An optional pointer may be passed that the call-back back function may use.

Returns

Returns OLB_EOK if hold-off timer has been started.

5.4.3.30 Olb_error_t olb_osci_stop (void)

Stops the oscilloscope.

Returns

Will return OLB_EOK if successful.

Note

The return should be checked.

5.4.3.31 Olb_error_t olb_osci_task (void)

Managing task for the oscilloscope module.

Note

This function should be called periodically keep the oscilloscope running.

Returns

Returns OLB_EOK. It will return another code from Olb_error_t if something went terribly wrong.

5.4.3.32 int32_t olb_osci_timeout_cb (void(*)(void) *cb*, uint32_t const *t_ms*, uint32_t const *is_single_shot*)

Start a timer that will call the provided function after the set time.

Parameters

cb	Call back function to call when the timer has elapsed.
t_ms	Time in milliseconds
is_single_shot	Shall the time-out stay active after it first fired?

Note

This functions is actually implemented within the 1kHz module. So make sure to initialize the 1kHz module first.

Returns

Returns the identifier number of the call-back. Use to ID when stopping the corresponding time-out call-back. If a negative ID was returned the call-back has not registered.

5.4.3.33 void olb_osci_tmr_global_init (void)

Do basic initialization of all timer modules used by OpenLab.

5.4.3.34 Olb_error_t olb_osci_trig_arm (void)

Arms the start event for the currently active trigger timer. Once a valid flank is detected the timer will start generating trigger events to the VADC.

Returns

Returns OLB_EOK on success.

5.4.3.35 Olb_error_t olb_osci_trig_init ( )

Initialize the trigger PWM output

Returns

If successful it will return OLB_EOK.

5.4.3.36 Olb_error_t olb_osci_trig_set_cfg (Olb_osci_trig_cfg_t *const *cfg*)

Apply the provided trigger configuration.

This function does the basic initialization work get the timer ready. To enable trigger generation for the VADC call olb_osci_trig_arm to actually arm the trigger timers to start when an external trigger arrives.

Parameters

cfg	Pointer to the trigger configuration Olb_osci_trig_cfg_t that shall be applied

Returns

If successful it will return OLB_EOK.

5.4.3.37 Olb_error_t olb_osci_trig_set_lvl (uint8_t const *num*, uint8_t const *lvl*)

Sets the trigger level at the requested channel number

Parameters

num	Channel number
lvl	Trigger level

Returns

If successful it will return OLB_EOK.

5.4.3.38 Olb_error_t olb_osci_trig_set_sample_rate (uint32_t *rate*)

Sets the sample rate for all channels.

Parameters

rate	Sample rate to set, expected value is S/sec

Returns

Returns OLB_EOK on success.

Note

Check the return value, as this function could fail.

5.4.3.39 Olb_error_t olb_osci_trig_set_smpl_mode (Olb_osci_sampl_mode_cfg_t * *cfg*)

Sets the sampling mode. The oscilloscope module may operate in real time sampling mode (RTS) or sequential equivalent time sampling mode (SETS).

Parameters

cfg	Sampling mode to set

Returns

Returns OLB_EOK on success.

5.4.3.40 Olb_error_t olb_osci_trig_stop (void)

Stops the trigger timers immediately.

Returns

Returns OLB_EOK on success.

5.4.3.41 void olb_osci_wait (uint32_t *t_ms*)

Busy wait for amount of milliseconds set.

Parameters

t_ms	Wait time in milliseconds.

Note

This functions is actually implemented within the 1kHz module. So make sure to initialize the 1kHz module first.

5.4.3.42 Olb_error_t olb_osci_wait_holdoff (void)

Waits for the hold-off time to elapse.

If the hold-off time has already elapsed it will return immediately.

Returns

Returns OLB_EOK when the hold-off time has elapsed.

5.5 Protocol

Collaboration diagram for Protocol:

Typedefs

- typedef Olb_error_t(∗ **olb_prot_pkt_hdl_f**) (Olb_protocol_packet_t ∗const pkt)

Enumerations

- enum Olb_protocol_code_t {
  OLB_PROT_CODE_STS = 1, OLB_PROT_CODE_SCS = 2, OLB_PROT_CODE_STB = 3, OLB_PROT_↩
  CODE_GHWV = 4,
  OLB_PROT_CODE_GSWV = 5, OLB_PROT_CODE_SLB = 6, OLB_PROT_CODE_SSM = 7, OLB_PRO↩
  T_CODE_SD = 8,
  OLB_PROT_CODE_HWV = 9, OLB_PROT_CODE_SWV = 10, OLB_PROT_CODE_ESD = 11, OLB_PR↩
  OT_CODE_ACK = 127,
  OLB_PROT_CODE_NACK = 126 }
- enum Olb_protocol_dev_type_t { OLB_PROT_DEV_ANY = 0, OLB_PROT_DEV_OSCI = 1, OLB_PROT_↩
  DEV_SIG = 2 }
- enum Olb_protocol_err_t {
  OLB_PROT_ERR_WARN = 1, OLB_PROT_ERR_NOT_SUPPORTED = 2, OLB_PROT_ERR_UNKNOWN
  = 3, OLB_PROT_ERR_FAILED = 4,
  OLB_PROT_OVERFLOW = 5 }

Functions

- struct __attribute__ ((__packed__)) Olb_protocol_head
- void olb_prot_init (void)
- void olb_prot_register_hdl (Olb_protocol_dev_type_t dev, olb_prot_pkt_hdl_f hdl_f)
- void olb_prot_send_ack (Olb_protocol_packet_t ∗const pkt)
- void olb_prot_send_nack (Olb_protocol_packet_t ∗const pkt, Olb_protocol_err_t const err)
- void olb_prot_send_pkt (Olb_protocol_packet_t ∗const pkt)
- void olb_prot_decode (uint8_t ∗const buf, uint32_t count)
- void olb_prot_dispatch (Olb_protocol_packet_t ∗pkt)
- Olb_error_t olb_prot_ghwv_handler (Olb_protocol_packet_t ∗const pkt)
- Olb_error_t olb_prot_gswv_handler (Olb_protocol_packet_t ∗const pkt)

Variables

- **Olb_protocol_head_t**
- **Olb_protocol_packet_t**

5.5.1 Detailed Description

5.5.2 Enumeration Type Documentation

5.5.2.1 enum Olb_protocol_code_t

Defines the code for the currently defined OpenLab protocol packets.

Enumerator

OLB_PROT_CODE_STS Set trigger settings.

OLB_PROT_CODE_SCS Set channel settings.

OLB_PROT_CODE_STB Set time base.

OLB_PROT_CODE_GHWV Get hardware version.

OLB_PROT_CODE_GSWV Get software version.

OLB_PROT_CODE_SLB Set loop back.

OLB_PROT_CODE_SSM Set sampling mode.

OLB_PROT_CODE_SD Sample data packet.

Deprecated Use OLB_PROT_CODE_ESD instead.

OLB_PROT_CODE_HWV Hardware version packet.

OLB_PROT_CODE_SWV Software version packet.

OLB_PROT_CODE_ESD Extended sample data packet.

OLB_PROT_CODE_ACK Acknowledge packet.

OLB_PROT_CODE_NACK Not acknowledge packet.

5.5.2.2 enum Olb_protocol_dev_type_t

Defines to which device the packet belongs to.

Enumerator

OLB_PROT_DEV_ANY Any device. That is the OpenLab application has to process the packet itself.

OLB_PROT_DEV_OSCI Packets belongs to the oscilloscope

OLB_PROT_DEV_SIG Packet belongs to the signal generator

Note

This device is not implemented yet.

5.5.2.3 enum Olb_protocol_err_t

Error codes that may be returned in a NAK packet.

Enumerator

OLB_PROT_ERR_WARN General warning.

OLB_PROT_ERR_NOT_SUPPORTED Command or parameter not supported.

OLB_PROT_ERR_UNKNOWN Unknown error.

OLB_PROT_ERR_FAILED Command failed.

OLB_PROT_OVERFLOW Memory overflow detected.

5.5.3 Function Documentation

5.5.3.1 struct __attribute__ ((__packed__))

Definition of the generic packet header that is used by all OpenLab protocol packets.

Definition of a generic OpenLab protocol packet.

This structure may be used as an memory overlay to directly access the individual fields of a packet.

5.5.3.2 void olb_prot_decode (uint8_t *const *buf*, uint32_t *count*)

This function tries to decode an OpenLab protocol packet from the provided data.

The protocol module keeps track of all data that has been received. From this previously received data and the new data it tries to find a valid packet. Once a valid packet has been decoded the command will be dispatched to the appropriate handler which may either be the common command handler if not specific device has been addresses or the device specific handler.

Parameters

buf	Pointer to memory area that holds new packet data.
count	Number of bytes to process.

5.5.3.3 void olb_prot_dispatch (Olb_protocol_packet_t * *pkt*)

Try to dispatch a possible valid OpenLab protocol packet to its intended receiver.

Parameters

pkt	Pointer to received Olb_protocol_packet_t

5.5.3.4 Olb_error_t olb_prot_ghwv_handler (Olb_protocol_packet_t *const *pkt*)

Private function used to reply to GHWV requests.

Parameters

pkt	Pointer to Olb_protocol_packet_t object that will be used to hold the HWV packet.

Returns

Value of Olb_error_t.

5.5.3.5 Olb_error_t olb_prot_gswv_handler (Olb_protocol_packet_t ∗const *pkt*)

Private function used to reply to GSWV requests.

Parameters

pkt	Pointer to Olb_protocol_packet_t object that will be used to hold the SWV packet.

Returns

Value of Olb_error_t.

5.5.3.6 void olb_prot_init (void)

Initializes the protocol handler.

Note

Must be called before any other function of this module.

5.5.3.7 void olb_prot_register_hdl (Olb_protocol_dev_type_t *dev*, olb_prot_pkt_hdl_f *hdl_f*)

Register a protocol handler function for a specific device.

Parameters

dev	Device code to register
hdl_f	Handler function

5.5.3.8 void olb_prot_send_ack (Olb_protocol_packet_t ∗const *pkt*)

Send an OpenLab acknowledge message

Parameters

msg	Pointer to Olb_protocol_packet_t object

This is a convenience function so that the user needs not to configure the message before sending it. create the acknowledge message

5.5.3.9 void olb_prot_send_nack (Olb_protocol_packet_t ∗const *pkt*, Olb_protocol_err_t const *err*)

Send an OpenLab not acknowledge message with an error code.

This is a convenience function so that the user needs not to configure the message before sending it.

Parameters

msg	Pointer to Olb_protocol_packet_t object
err	Error code to send

5.5.3.10 void olb_prot_send_pkt (Olb_protocol_packet_t ∗const *pkt*)

Send an OpenLab message

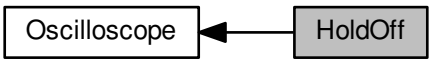
Sends the message as provided. This function will not do any changes to it except calculating the checksums. Therefore the user has to set all fields except the checksums correctly before calling this function.

Parameters

msg	Pointer to Olb_protocol_packet_t object

5.6 HoldOff

Collaboration diagram for HoldOff:



Chapter 6

Data Structure Documentation

6.1 CRC16_CCIT Struct Reference

Data Fields

- uint16_t **value**

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

- Libraries/OpenLab/inc/CRC16_CCIT.h

6.2 Olb_fifo8_t Struct Reference

```
#include <olb_fifo8.h>
```

Data Fields

- uint8_t * mem
- uint32_t w_i
- uint32_t fill_count
- uint32_t size

6.2.1 Detailed Description

FIFO (first in first out) buffer struct.

6.2.2 Field Documentation

6.2.2.1 uint32_t Olb_fifo8_t::fill_count

Number of bytes currently in the buffer

6.2.2.2 uint8_t* Olb_fifo8_t::mem

Internal buffer memory

6.2.2.3 uint32_t Olb_fifo8_t::size

Total size of buffer memory

6.2.2.4 uint32_t Olb_fifo8_t::w_i

Write index

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

- Libraries/OpenLab/inc/olb_fifo8.h

6.3 Olb_osci_chan_input_cfg_t Struct Reference

`#include <olb_osci.h>`

Data Fields

- Olb_osci_coupling_t **coupling**
- uint8_t **gain**

6.3.1 Detailed Description

Channel settings.

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

- Libraries/OpenLab/inc/olb_osci.h

6.4 Olb_osci_sampl_mode_cfg_t Struct Reference

`#include <olb_osci.h>`

Data Fields

- Olb_osci_mode_t **mode**
- uint32_t **smpls_per_acqu_round**
- uint32_t **acqu_rounds**

6.4.1 Detailed Description

Collected configuration settings regarding sampling of input channels.

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

- Libraries/OpenLab/inc/olb_osci.h

6.5 Olb_osci_trig_cfg_t Struct Reference

`#include <olb_osci.h>`

Data Fields

- Olb_osci_trig_mode_t **mode**
- Olb_osci_trig_active_flank_t **flank**
- uint16_t **lvl**
- uint32_t **hold_off**
- uint8_t **act_chan**

6.5.1 Detailed Description

Summarized configuration settings for the trigger.

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

- Libraries/OpenLab/inc/olb_osci.h

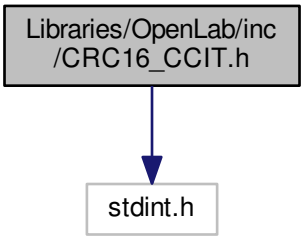
Chapter 7

File Documentation

7.1 Libraries/OpenLab/inc/CRC16_CCIT.h File Reference

```
#include <stdint.h>
```

Include dependency graph for CRC16_CCIT.h:



Data Structures

- struct CRC16_CCIT

Functions

- void **CRC16_CCIT_update** (struct CRC16_CCIT *const crc_state, uint8_t ch)
- void **CRC16_CCIT_reset** (struct CRC16_CCIT *const crc_state)

7.1.1 Detailed Description

Date

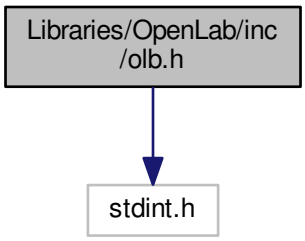
1 Jan 2016

Author

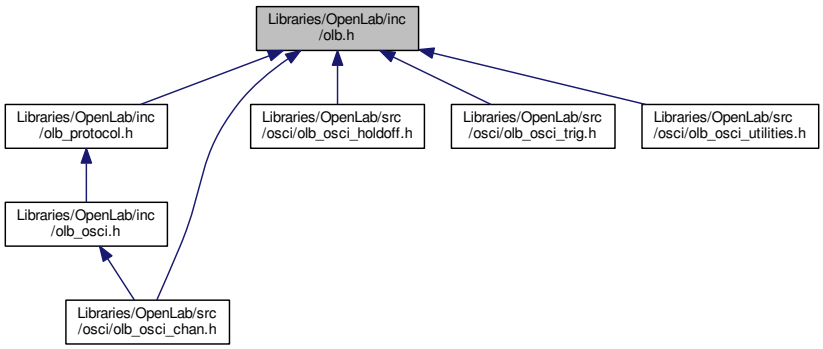
Harald Schloffer

7.2 Libraries/OpenLab/inc/olb.h File Reference

`#include <stdint.h>`
Include dependency graph for olb.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **OLB_SW_MAJ_VERSION** 0
- #define **OLB_SW_MIN_VERSION** 0
- #define **OLB_SW_SUB_VERSION** 10
- #define **OLB_HW_MAJ_VERSION** 2
- #define **OLB_HW_MIN_VERSION** 0
- #define **OLB_HW_SUB_VERSION** 2

Enumerations

- enum Olb_error_t {
  OLB_EOK = 0, OLB_ETOUT = -1, OLB_EWARNING = -2, OLB_ENOT_SUPPORTED = -3,
  OLB_EUNKNOWN_COMMAND = -4, OLB_EFAIL = -5, OLB_EOVERFLOW = -6, OLB_EUNDERFLOW = -7,
  OLB_EAGAIN = -8 }

Functions

- void olb_init (void)
- void olb_task (void)
- void olb_wait (uint32_t wait_time)
- uint16_t olb_receive (uint8_t *data, uint16_t count)
- uint16_t olb_send (uint8_t *data, uint16_t count)

7.2.1 Detailed Description

Date

Jul 11, 2016

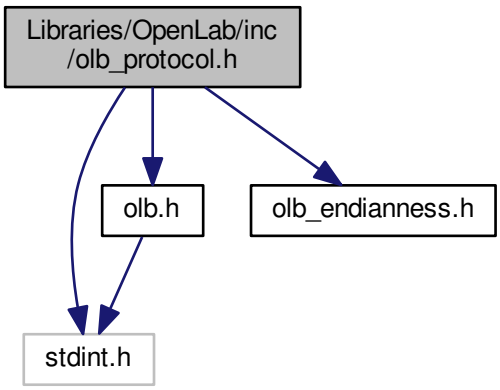
Author

Harald Schloffer

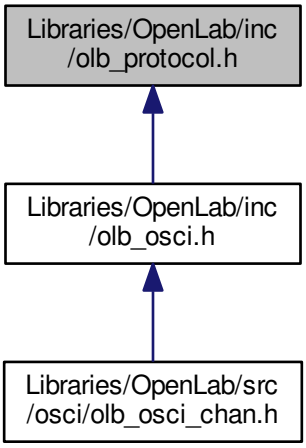
7.3 Libraries/OpenLab/inc/olb_protocol.h File Reference

```
#include <stdint.h>
#include "olb.h"
#include "olb_endianness.h"
```

Include dependency graph for olb_protocol.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **OLB_PROT_REPLY_PLD_SIZE_MAX** 2048
- #define **OLB_PROT_CMD_PLD_SIZE_MAX** 55

Typedefs

- typedef Olb_error_t(∗ **olb_prot_pkt_hdl_f**) (Olb_protocol_packet_t ∗const pkt)

Enumerations

- enum Olb_protocol_code_t {
  OLB_PROT_CODE_STS = 1, OLB_PROT_CODE_SCS = 2, OLB_PROT_CODE_STB = 3, OLB_PROT_↩
  CODE_GHWV = 4,
  OLB_PROT_CODE_GSWV = 5, OLB_PROT_CODE_SLB = 6, OLB_PROT_CODE_SSM = 7, OLB_PRO↩
  T_CODE_SD = 8,
  OLB_PROT_CODE_HWV = 9, OLB_PROT_CODE_SWV = 10, OLB_PROT_CODE_ESD = 11, OLB_PR↩
  OT_CODE_ACK = 127,
  OLB_PROT_CODE_NACK = 126 }
- enum Olb_protocol_dev_type_t { OLB_PROT_DEV_ANY = 0, OLB_PROT_DEV_OSCI = 1, OLB_PROT_↩
  DEV_SIG = 2 }
- enum Olb_protocol_err_t {
  OLB_PROT_ERR_WARN = 1, OLB_PROT_ERR_NOT_SUPPORTED = 2, OLB_PROT_ERR_UNKNOWN
  = 3, OLB_PROT_ERR_FAILED = 4,
  OLB_PROT_OVERFLOW = 5 }

Functions

- struct __attribute__ ((__packed__)) Olb_protocol_head
- void olb_prot_init (void)
- void olb_prot_register_hdl (Olb_protocol_dev_type_t dev, olb_prot_pkt_hdl_f hdl_f)
- void olb_prot_send_ack (Olb_protocol_packet_t *const pkt)
- void olb_prot_send_nack (Olb_protocol_packet_t *const pkt, Olb_protocol_err_t const err)
- void olb_prot_send_pkt (Olb_protocol_packet_t *const pkt)
- void olb_prot_decode (uint8_t *const buf, uint32_t count)
- void olb_prot_dispatch (Olb_protocol_packet_t *pkt)
- Olb_error_t olb_prot_ghwv_handler (Olb_protocol_packet_t *const pkt)
- Olb_error_t olb_prot_gswv_handler (Olb_protocol_packet_t *const pkt)

Variables

- **Olb_protocol_head_t**
- **Olb_protocol_packet_t**

7.3.1 Detailed Description

Date

Jul 13, 2016

Author

Harald Schloffer

7.4 Libraries/OpenLab/src/osci/olb_osci_cfg.h File Reference

Macros

- #define **OSCI_CHAN_0_QUEUE_TRIG_SIG** XMC_VADC_REQ_TR_A
- #define **OSCI_CHAN_0_VADC** VADC
- #define **OSCI_CHAN_0_VADC_GROUP** VADC_G0
- #define **OSCI_CHAN_0_VADC_CHANNEL** 0
- #define **OSCI_CHAN_0_VADC_RESULT_REG** 7
- #define **OSCI_CHAN_0_VADC_IRQn** VADC0_G0_0_IRQn
- #define **OSCI_CHAN_0_VADC_IRQ_HANDLER** VADC0_G0_0_IRQHandler
- #define **OSCI_CHAN_0_NVIC_PRIO** 0
- #define **OSCI_CHAN_0_GPIO_PORT** PORT14
- #define **OSCI_CHAN_0_GPIO_PIN** 0
- #define **OSCI_CHAN_0_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_MASTER_GROUP_NUM** 0
- #define **OSCI_CHAN_MASTER_GROUP_IND** 0
- #define **OSCI_CHAN_0_CCU4** CCU40
- #define **OSCI_CHAN_0_CCU4_SLICE** CCU40_CC40
- #define **OSCI_CHAN_0_CCU4_SLICE_NUM** 0
- #define **OSCI_CHAN_0_CCU4_SHADOW_TRANSFER_SLICE** (XMC_CCU4_SHADOW_TRANSFER_S↩ LICE_0 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_0)

- #define **OSCI_CHAN_0_CCU4_SLICE_INPUT** XMC_CCU4_SLICE_INPUT_C
- #define **OSCI_CHAN_0_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_2
- #define **OSCI_CHAN_0_CCU4_SLICE_NVIC_IRQN** CCU40_2_IRQn
- #define **OSCI_CHAN_0_CCU4_SLICE_NVIC_IRQ_HDLR** CCU40_2_IRQHandler
- #define **OSCI_CHAN_0_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_CHAN_0_CCU4_SLICE_START_EVT_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_EVENT0
- #define **OSCI_CHAN_0_CCU4_SLICE_START_EVT_SR_ID** XMC_CCU4_SLICE_SR_ID_3
- #define **OSCI_CHAN_0_CCU4_SLICE_START_NVIC_IRQN** CCU40_3_IRQn
- #define **OSCI_CHAN_0_CCU4_SLICE_START_NVIC_IRQ_HDLR** CCU40_3_IRQHandler
- #define **OSCI_CHAN_0_TRIG_GPIO_PORT** PORT2
- #define **OSCI_CHAN_0_TRIG_GPIO_PIN** 1
- #define **OSCI_CHAN_0_TRIG_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_1_QUEUE_TRIG_SIG** XMC_VADC_REQ_TR_F
- #define **OSCI_CHAN_1_VADC** VADC
- #define **OSCI_CHAN_1_VADC_GROUP** VADC_G2
- #define **OSCI_CHAN_1_VADC_CHANNEL** 0
- #define **OSCI_CHAN_1_VADC_RESULT_REG** 7
- #define **OSCI_CHAN_1_VADC_IRQn** VADC0_G2_0_IRQn
- #define **OSCI_CHAN_1_VADC_IRQ_HANDLER** VADC0_G2_0_IRQHandler
- #define **OSCI_CHAN_1_NVIC_PRIO** 0
- #define **OSCI_CHAN_1_GPIO_PORT** PORT14
- #define **OSCI_CHAN_1_GPIO_PIN** 4
- #define **OSCI_CHAN_1_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OSCI_CHAN_SLAVE_GROUP_NUM** 2
- #define **OSCI_CHAN_SLAVE_GROUP_IND** 1
- #define **OSCI_CHAN_1_CCU4** CCU43
- #define **OSCI_CHAN_1_CCU4_SLICE** CCU43_CC40
- #define **OSCI_CHAN_1_CCU4_SLICE_NUM** 0
- #define **OSCI_CHAN_1_CCU4_SHADOW_TRANSFER_SLICE** (XMC_CCU4_SHADOW_TRANSFER_S↩ LICE_0 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_0)
- #define **OSCI_CHAN_1_CCU4_SLICE_INPUT** XMC_CCU4_SLICE_INPUT_B
- #define **OSCI_CHAN_1_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_3
- #define **OSCI_CHAN_1_CCU4_SLICE_NVIC_IRQN** CCU43_3_IRQn
- #define **OSCI_CHAN_1_CCU4_SLICE_NVIC_IRQ_HDLR** CCU43_3_IRQHandler
- #define **OSCI_CHAN_1_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_CHAN_1_CCU4_SLICE_START_EVT_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_EVENT0
- #define **OSCI_CHAN_1_CCU4_SLICE_START_EVT_SR_ID** XMC_CCU4_SLICE_SR_ID_1
- #define **OSCI_CHAN_1_CCU4_SLICE_START_NVIC_IRQN** CCU43_1_IRQn
- #define **OSCI_CHAN_1_CCU4_SLICE_START_NVIC_IRQ_HDLR** CCU43_1_IRQHandler
- #define **OSCI_CHAN_1_TRIG_GPIO_PORT** PORT2
- #define **OSCI_CHAN_1_TRIG_GPIO_PIN** 14
- #define **OSCI_CHAN_1_TRIG_GPIO_MODE** XMC_GPIO_MODE_INPUT_TRISTATE
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU** CCU80
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE** CCU80_CC81
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_NUM** 1
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_SHADOW_TRANSFER** XMC_CCU8_SHADOW_T↩ RANSFER_SLICE_1
- #define **OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_COMPARE_CHANNEL** XMC_CCU8_SLICE_COM↩ PARE_CHANNEL_2
- #define **OLB_OSCI_TRIG_PWM_CH0_FREQ** 100000
- #define **OLB_OSCI_TRIG_PWM_CH0_PORT** PORT0
- #define **OLB_OSCI_TRIG_PWM_CH0_PIN** 9
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU** CCU80
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE** CCU80_CC80
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_NUM** 0

- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_SHADOW_TRANSFER** XMC_CCU8_SHADOW_T↩ RANSFER_SLICE_0
- #define **OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_COMPARE_CHANNEL** XMC_CCU8_SLICE_COM↩ PARE_CHANNEL_2
- #define **OLB_OSCI_TRIG_PWM_CH1_FREQ** 100000
- #define **OLB_OSCI_TRIG_PWM_CH1_PORT** PORT0
- #define **OLB_OSCI_TRIG_PWM_CH1_PIN** 10
- #define **OSCI_HOLDOFF_CCU4** CCU40
- #define **OSCI_HOLDOFF_CCU4_SLICE** CCU40_CC41
- #define **OSCI_HOLDOFF_CCU4_SLICE_NUM** 1
- #define **OSCI_HOLDOFF_CCU4_SLICE_SR_ID** XMC_CCU4_SLICE_SR_ID_1
- #define **OSCI_HOLDOFF_CCU4_SLICE_IRQ_ID** XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH
- #define **OSCI_HOLDOFF_CCU4_SHADOW_TRANSFER** (XMC_CCU4_SHADOW_TRANSFER_SLICE_↩ 1 | XMC_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_1)
- #define **OSCI_HOLDOFF_NVIC_IRQ_PRIO** 1
- #define **OSCI_HOLDOFF_NVIC_IRQN** CCU40_1_IRQn
- #define **OSCI_HOLDOFF_NVIC_IRQ_HDLR** CCU40_1_IRQHandler
- #define **OLB_OSCI_CH0_GAIN0_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN0_PIN** (0U)
- #define **OLB_OSCI_CH0_GAIN1_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN1_PIN** (1U)
- #define **OLB_OSCI_CH0_GAIN2_PORT** XMC_GPIO_PORT3
- #define **OLB_OSCI_CH0_GAIN2_PIN** (2U)
- #define **OLB_OSCI_CH1_GAIN0_PORT** XMC_GPIO_PORT0
- #define **OLB_OSCI_CH1_GAIN0_PIN** (0U)
- #define **OLB_OSCI_CH1_GAIN1_PORT** XMC_GPIO_PORT0
- #define **OLB_OSCI_CH1_GAIN1_PIN** (1U)
- #define **OLB_OSCI_CH1_GAIN2_PORT** XMC_GPIO_PORT2
- #define **OLB_OSCI_CH1_GAIN2_PIN** (15U)
- #define **OLB_OSCI_1kHz_FREQ** 1000
- #define **OLB_OSCI_1kHz_CCU** CCU40
- #define **OLB_OSCI_1kHz_CCU_SLICE_NUM** 2
- #define **OLB_OSCI_1kHz_CCU_SLICE** CCU40_CC42
- #define **OLB_OSCI_1kHz_CCU_SHADOW_TRANSF** XMC_CCU4_SHADOW_TRANSFER_SLICE_2
- #define **OLB_OSCI_1kHz_PORT** XMC_GPIO_PORT2
- #define **OLB_OSCI_1kHz_PIN** (10)
- #define **OLB_OSCI_1kHz_NVIC_IRQn** CCU40_0_IRQn
- #define **OLB_OSCI_1kHz_ISR** CCU40_0_IRQHandler
- #define **OLB_OSCI_1kHz_IRQ_PRIO** 2

7.4.1 Detailed Description

Date

Aug 15, 2016

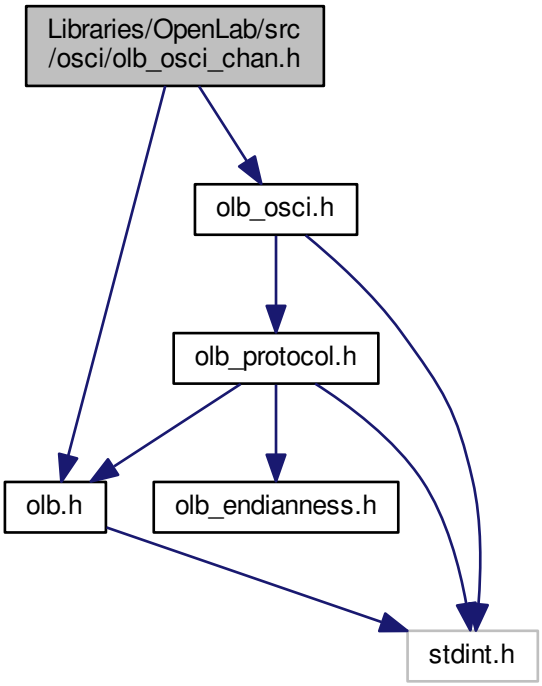
Author

Harald Schloffer

7.5 Libraries/OpenLab/src/osci/olb_osci_chan.h File Reference

```
#include "olb.h"
#include "olb_osci.h"
```

Include dependency graph for olb_osci_chan.h:



Functions

- void olb_osci_chan_init (void)
- Olb_error_t olb_osci_chan_set_sample_buffer (uint8_t const chan_num, uint32_t buf_size, uint8_t *const buf)
- Olb_error_t olb_osci_chan_set_sample_count (uint8_t const chan_num, uint32_t const count)
- Olb_error_t olb_osci_chan_disable (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_enable (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_has_samples (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_is_busy (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_is_enabled (uint8_t const chan_num)
- Olb_error_t olb_osci_chan_abort (void)
- Olb_error_t olb_osci_chan_arm (uint32_t smpl_count)
- Olb_error_t olb_osci_chan_write_to_smpl_buf (uint8_t chan, uint8_t const smpl)
- Olb_error_t olb_osci_chan_reset_smpl_data (uint8_t chan)
- Olb_error_t olb_osci_chan_set_trig_src (uint8_t chan)

7.5.1 Detailed Description

Date

Jul 27, 2016

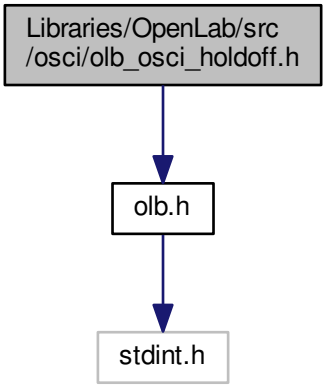
Author

Harald Schloffer

7.6 Libraries/OpenLab/src/osci/olb_osci_holdoff.h File Reference

```
#include "olb.h"
```

Include dependency graph for olb_osci_holdoff.h:



Functions

- void olb_osci_init_holdoff (void)
- Olb_error_t olb_osci_set_holdoff (uint32_t holdoff_time)
- Olb_error_t olb_osci_start_holdoff (void)
- Olb_error_t olb_osci_start_holdoff_cb (void(∗cb_f)(void ∗), void ∗para)
- Olb_error_t olb_osci_wait_holdoff (void)

7.6.1 Detailed Description

Date

Aug 15, 2016

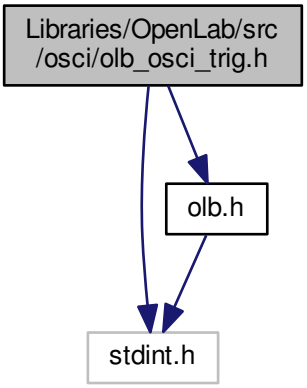
Author

Harald Schloffer

7.7 Libraries/OpenLab/src/osci/olb_osci_trig.h File Reference

```
#include <stdint.h>
#include "olb.h"
```

Include dependency graph for olb_osci_trig.h:



Functions

- Olb_error_t olb_osci_trig_init ()
- Olb_error_t olb_osci_trig_set_cfg (Olb_osci_trig_cfg_t *const cfg)
- Olb_error_t olb_osci_trig_set_lvl (uint8_t const num, uint8_t const lvl)
- Olb_error_t olb_osci_trig_set_sample_rate (uint32_t rate)
- Olb_error_t olb_osci_trig_arm (void)
- Olb_error_t olb_osci_trig_stop (void)
- Olb_error_t olb_osci_trig_set_smpl_mode (Olb_osci_sampl_mode_cfg_t *cfg)
- Olb_error_t **olb_osci_trig_force** (void)
- void **olb_osci_trig_incr_ets_offset** (void)
- void **olb_osci_trig_reset_ets_offset** (void)

7.7.1 Detailed Description

Date

Jul 27, 2016

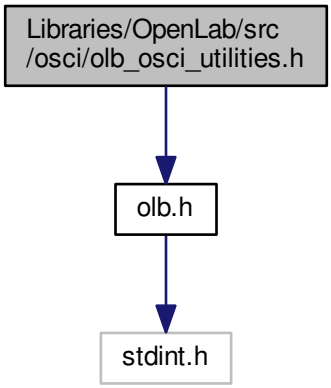
Author

Harald Schloffer

7.8 Libraries/OpenLab/src/osci/olb_osci_utilities.h File Reference

```
#include "olb.h"
```

Include dependency graph for olb_osci_utilities.h:



Functions

- uint32_t **olb_osci_cfg_get_chan_count** (void)
- uint32_t **olb_osci_cfg_get_max_sample_count** (void)
- uint32_t **olb_osci_cfg_get_default_smpl_rate** (void)
- void olb_osci_tmr_global_init (void)

7.8.1 Detailed Description

Date

Aug 13, 2016

Author

Harald Schloffer

Index