

OpenLab - XMC4500

Generated by Doxygen 1.8.11

Contents

1	Deprecated List	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	OpenLab	9
5.1.1	Detailed Description	10
5.1.2	Enumeration Type Documentation	10
5.1.2.1	Olb_error_t	10
5.1.3	Function Documentation	10
5.1.3.1	olb_init(void)	10
5.1.3.2	olb_receive(uint8_t *data, uint16_t count)	10
5.1.3.3	olb_send(uint8_t *data, uint16_t count)	11
5.1.3.4	olb_task(void)	11
5.1.3.5	olb_wait(uint32_t wait_time)	11
5.2	Utilities	12
5.2.1	Detailed Description	12
5.3	Fifo8	13

5.3.1	Detailed Description	13
5.3.2	Function Documentation	13
5.3.2.1	olb_fifo8_clear(Olb_fifo8_t *const fifo)	13
5.3.2.2	olb_fifo8_count(Olb_fifo8_t *const fifo)	13
5.3.2.3	olb_fifo8_get_mem(Olb_fifo8_t *const fifo)	14
5.3.2.4	olb_fifo8_init(Olb_fifo8_t *const fifo, uint8_t *const mem, uint32_t const size)	14
5.3.2.5	olb_fifo8_pop(Olb_fifo8_t *const fifo, uint8_t *const data)	14
5.3.2.6	olb_fifo8_push(Olb_fifo8_t *const fifo, uint8_t const data)	14
5.4	Oscilloscope	16
5.4.1	Detailed Description	20
5.4.2	Enumeration Type Documentation	20
5.4.2.1	Olb_osci_coupling_t	20
5.4.2.2	Olb_osci_event_t	20
5.4.2.3	Olb_osci_mode_t	20
5.4.2.4	Olb_osci_trig_active_flank_t	20
5.4.2.5	Olb_osci_trig_mode_t	21
5.4.3	Function Documentation	21
5.4.3.1	olb_osci_chan_abort(void)	21
5.4.3.2	olb_osci_chan_arm(uint32_t smpl_count)	21
5.4.3.3	olb_osci_chan_disable(uint8_t const chan_num)	21
5.4.3.4	olb_osci_chan_enable(uint8_t const chan_num)	21
5.4.3.5	olb_osci_chan_has_samples(uint8_t const chan_num)	22
5.4.3.6	olb_osci_chan_init(void)	22
5.4.3.7	olb_osci_chan_is_busy(uint8_t const chan_num)	22
5.4.3.8	olb_osci_chan_is_enabled(uint8_t const chan_num)	22
5.4.3.9	olb_osci_chan_reset_smpl_data(uint8_t chan)	23
5.4.3.10	olb_osci_chan_set_sample_buffer(uint8_t const chan_num, uint32_t buf_size, uint8_t *const buf)	23
5.4.3.11	olb_osci_chan_set_sample_count(uint8_t const chan_num, uint32_t const count)	23
5.4.3.12	olb_osci_chan_set_trig_src(uint8_t chan)	24
5.4.3.13	olb_osci_chan_write_to_smpl_buf(uint8_t chan, uint8_t const smpl)	24

5.4.3.14	<code>olb_osci_get_samples(uint8_t chan, uint32_t count, uint8_t *const dst)</code>	24
5.4.3.15	<code>olb_osci_init(void)</code>	25
5.4.3.16	<code>olb_osci_init_1kHz(void)</code>	25
5.4.3.17	<code>olb_osci_init_gain(void)</code>	25
5.4.3.18	<code>olb_osci_init_holdoff(void)</code>	25
5.4.3.19	<code>olb_osci_prot_pkt_hdl(Olb_protocol_packet_t *const pkt)</code>	25
5.4.3.20	<code>olb_osci_raise_evt(Olb_osci_event_t evt, int32_t data)</code>	25
5.4.3.21	<code>olb_osci_set_channel_input_cfg(uint8_t chan, Olb_osci_chan_input_cfg_t *const cfg)</code>	26
5.4.3.22	<code>Olb_osci_set_gain(uint8_t const channel, uint8_t gain)</code>	26
5.4.3.23	<code>olb_osci_set_holdoff(uint32_t holdoff_time)</code>	26
5.4.3.24	<code>olb_osci_set_sample_rate(uint32_t sps)</code>	27
5.4.3.25	<code>olb_osci_set_sampling_cfg(Olb_osci_sampl_mode_cfg_t *const cfg)</code>	27
5.4.3.26	<code>olb_osci_set_trigger_cfg(Olb_osci_trig_cfg_t *const cfg)</code>	27
5.4.3.27	<code>olb_osci_start(void)</code>	28
5.4.3.28	<code>olb_osci_start_holdoff(void)</code>	28
5.4.3.29	<code>olb_osci_start_holdoff_cb(void(*cb_f)(void *), void *para)</code>	28
5.4.3.30	<code>olb_osci_stop(void)</code>	28
5.4.3.31	<code>olb_osci_task(void)</code>	29
5.4.3.32	<code>olb_osci_timeout_cb(void(*cb)(void), uint32_t const t_ms, uint32_t const is_↔ single_shot)</code>	29
5.4.3.33	<code>olb_osci_tmr_global_init(void)</code>	29
5.4.3.34	<code>olb_osci_trig_arm(void)</code>	30
5.4.3.35	<code>olb_osci_trig_init()</code>	30
5.4.3.36	<code>olb_osci_trig_set_cfg(Olb_osci_trig_cfg_t *const cfg)</code>	30
5.4.3.37	<code>olb_osci_trig_set_lv(uint8_t const num, uint8_t const lvl)</code>	30
5.4.3.38	<code>olb_osci_trig_set_sample_rate(uint32_t rate)</code>	31
5.4.3.39	<code>olb_osci_trig_set_smpl_mode(Olb_osci_sampl_mode_cfg_t *cfg)</code>	31
5.4.3.40	<code>olb_osci_trig_stop(void)</code>	31
5.4.3.41	<code>olb_osci_wait(uint32_t t_ms)</code>	31
5.4.3.42	<code>olb_osci_wait_holdoff(void)</code>	32

5.5	Protocol	33
5.5.1	Detailed Description	34
5.5.2	Enumeration Type Documentation	34
5.5.2.1	Olb_protocol_code_t	34
5.5.2.2	Olb_protocol_dev_type_t	34
5.5.2.3	Olb_protocol_err_t	34
5.5.3	Function Documentation	35
5.5.3.1	__attribute__((__packed__)) Olb_protocol_head	35
5.5.3.2	olb_prot_decode(uint8_t *const buf, uint32_t count)	35
5.5.3.3	olb_prot_dispatch(Olb_protocol_packet_t *pkt)	35
5.5.3.4	olb_prot_ghwv_handler(Olb_protocol_packet_t *const pkt)	35
5.5.3.5	olb_prot_gswv_handler(Olb_protocol_packet_t *const pkt)	36
5.5.3.6	olb_prot_init(void)	36
5.5.3.7	olb_prot_register_hdl(Olb_protocol_dev_type_t dev, olb_prot_pkt_hdl_f hdl_f)	36
5.5.3.8	olb_prot_send_ack(Olb_protocol_packet_t *const pkt)	36
5.5.3.9	olb_prot_send_nack(Olb_protocol_packet_t *const pkt, Olb_protocol_err_t const err)	36
5.5.3.10	olb_prot_send_pkt(Olb_protocol_packet_t *const pkt)	37
5.6	HoldOff	38
6	Data Structure Documentation	39
6.1	CRC16_CCIT Struct Reference	39
6.2	Olb_fifo8_t Struct Reference	39
6.2.1	Detailed Description	39
6.2.2	Field Documentation	39
6.2.2.1	fill_count	39
6.2.2.2	mem	40
6.2.2.3	size	40
6.2.2.4	w_i	40
6.3	Olb_osci_chan_input_cfg_t Struct Reference	40
6.3.1	Detailed Description	40
6.4	Olb_osci_sampl_mode_cfg_t Struct Reference	40
6.4.1	Detailed Description	41
6.5	Olb_osci_trig_cfg_t Struct Reference	41
6.5.1	Detailed Description	41

7 File Documentation	43
7.1 Libraries/OpenLab/inc/CRC16_CCIT.h File Reference	43
7.1.1 Detailed Description	43
7.2 Libraries/OpenLab/inc/olb.h File Reference	44
7.2.1 Detailed Description	45
7.3 Libraries/OpenLab/inc/olb_protocol.h File Reference	45
7.3.1 Detailed Description	47
7.4 Libraries/OpenLab/src/osci/olb_osci_cfg.h File Reference	47
7.4.1 Detailed Description	49
7.5 Libraries/OpenLab/src/osci/olb_osci_chan.h File Reference	50
7.5.1 Detailed Description	51
7.6 Libraries/OpenLab/src/osci/olb_osci_holdoff.h File Reference	51
7.6.1 Detailed Description	51
7.7 Libraries/OpenLab/src/osci/olb_osci_trig.h File Reference	52
7.7.1 Detailed Description	52
7.8 Libraries/OpenLab/src/osci/olb_osci_utilities.h File Reference	53
7.8.1 Detailed Description	53
Index	55

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:

OpenLab	9
Utilities	12
Fifo8	13
Oscilloscope	16
HoldOff	38
Protocol	33

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

CRC16_CCIT	39
Olb_fifo8_t	39
Olb_osci_chan_input_cfg_t	40
Olb_osci_sampl_mode_cfg_t	40
Olb_osci_trig_cfg_t	41

Chapter 4

File Index

4.1 File List

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

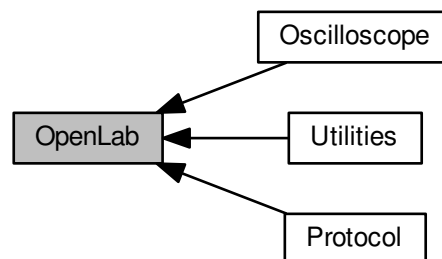
Libraries/OpenLab/inc/ CRC16_CCIT.h	43
Libraries/OpenLab/inc/ olb.h	44
Libraries/OpenLab/inc/ olb_endianness.h	??
Libraries/OpenLab/inc/ olb_fifo8.h	??
Libraries/OpenLab/inc/ olb_osci.h	??
Libraries/OpenLab/inc/ olb_protocol.h	45
Libraries/OpenLab/src/osci/ olb_osci_cfg.h	47
Libraries/OpenLab/src/osci/ olb_osci_chan.h	50
Libraries/OpenLab/src/osci/ olb_osci_holdoff.h	51
Libraries/OpenLab/src/osci/ olb_osci_trig.h	52
Libraries/OpenLab/src/osci/ olb_osci_utilities.h	53

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

<i>data</i>	Pointer to destination data.
<i>count</i>	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

<i>data</i>	Pointer to source data.
<i>count</i>	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

<i>wait_time</i>	Wait time in micro seconds.
------------------	-----------------------------

Note

Minimum time resolution is 10 micro seconds.

5.2 Utilities

Collaboration diagram for Utilities:



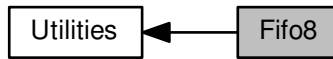
Modules

- [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

<i>fifo</i>	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

<i>fifo</i>	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

<i>fifo</i>	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

<i>fifo</i>	Pointer to Olb_fifo8_t object.
<i>mem</i>	Pointer to memory area that should be used to the internal FIFO buffer.
<i>size</i>	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

<i>fifo</i>	Pointer to Olb_fifo8_t object.
<i>data</i>	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

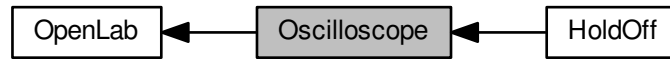
<i>fifo</i>	Pointer to Olb_fifo8_t object.
<i>data</i>	Data byte to push

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_SLICE_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_SLICE_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_TRANSFER_SLICE_1
- #define OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_COMPARE_CHANNEL XMC_CCU8_SLICE_COMPARE_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_TRANSFER_SLICE_0
- #define OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_COMPARE_CHANNEL XMC_CCU8_SLICE_COMPARE_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_osc_mode_t { OLB_OSCI_MODE_REALT, OLB_OSCI_MODE_SETS, OLB_OSCI_MODE_R↵
ETS }`
- `enum Olb_osc_coupling_t { OLB_OSCI_COUPLING_OFF = 0, OLB_OSCI_COUPLING_DC = 1, OLB_O↵
SCI_COUPLING_AC = 2 }`
- `enum Olb_osc_trig_mode_t { OLB_OSCI_TRIG_MODE_AUTO = 0, OLB_OSCI_TRIG_MODE_NORMAL =
1 }`
- `enum Olb_osc_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_osc_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 smp_l_count)`

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

Parameters

<i>smp_l_count</i>	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

<i>chan_num</i>	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

<i>chan_num</i>	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

<i>chan_num</i>	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

<i>chan_num</i>	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

<i>chan_num</i>	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

<i>chan</i>	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 buf)`

Sets the sample buffer to be used to store samples.

Parameters

<i>chan_num</i>	Channel number starting at zero.
<i>buf_size</i>	The size of the buffer in bytes.
<i>buf</i>	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

<i>chan_num</i>	Channel number starting at zero.
<i>count</i>	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

<i>chan</i>	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

<i>chan</i>	Channel number
<i>smpl</i>	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

<i>chan</i>	Channel number starting at zero
<i>count</i>	Maximum number of samples to read
<i>dst</i>	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

<i>pkt</i>	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

<i>evt</i>	Event that just happened
<i>data</i>	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

<i>chan</i>	Number of channel starting at zero.
<i>cfg</i>	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

<i>channel</i>	Channel number starting at zero
<i>gain</i>	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

<i>holdoff_time</i>	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

<i>sps</i>	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

<i>cfg</i>	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

<i>cfg</i>	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

<i>cb_↵_f</i>	Call back function to call at end of hold-off time.
<i>para</i>	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

<i>cb</i>	Call back function to call when the timer has elapsed.
<i>t_ms</i>	Time in milliseconds
<i>is_single_shot</i>	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

<i>cfg</i>	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

<i>num</i>	Channel number
<i>lvl</i>	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

<i>rate</i>	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_smpl_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

<i>cfg</i>	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

<i>t_ms</i>	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.

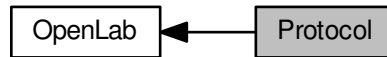
If the hold-off time has already elapsed it will return immediately.

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_PROT_CODE_SD` = 8,
`OLB_PROT_CODE_HWV` = 9, `OLB_PROT_CODE_SWV` = 10, `OLB_PROT_CODE_ESD` = 11, `OLB_PROT_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

<i>buf</i>	Pointer to memory area that holds new packet data.
<i>count</i>	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

<i>pkt</i>	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

<i>pkt</i>	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

<i>pkt</i>	Pointer to <code>Olb_protocol_packet_t</code> 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

<i>dev</i>	Device code to register
<i>hdl↔ _f</i>	Handler function

5.5.3.8 `void olb_prot_send_ack (Olb_protocol_packet_t *const pkt)`

Send an OpenLab acknowledge message

Parameters

<i>msg</i>	Pointer to <code>Olb_protocol_packet_t</code> 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

<i>msg</i>	Pointer to <code>Olb_protocol_packet_t</code> object
<i>err</i>	Error code to send

5.5.3.10 `void olb_prot_send_pkt (Olb_protocol_packet_t *const pkt)`

Send an OpenLab message

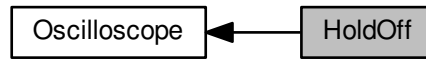
Sends the message as provided. This function will not do any changes to it except calculating the checksums. Therefore the user has to set all fields except the checksums correctly before calling this function.

Parameters

<i>msg</i>	Pointer to <code>Olb_protocol_packet_t</code> 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

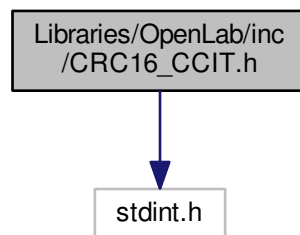
Chapter 7

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

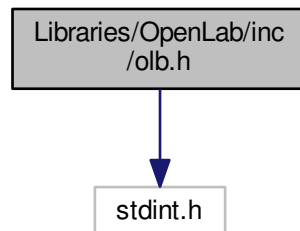
1 Jan 2016

Author

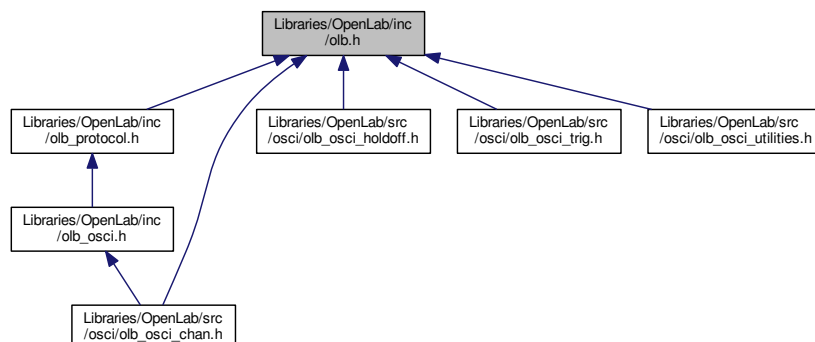
Harald Schloffer

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

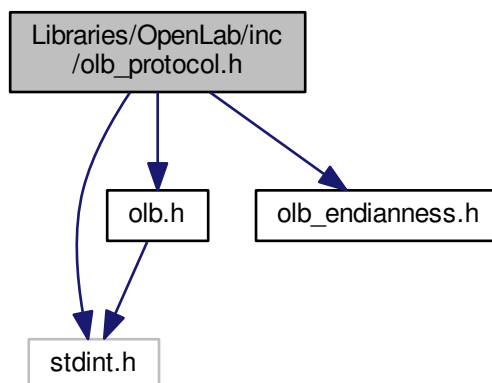
Jul 11, 2016

Author

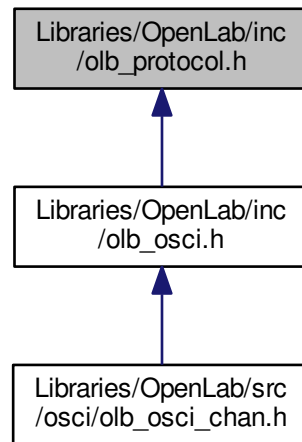
Harald Schloffer

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_PROT_CODE_SD = 8,`
`OLB_PROT_CODE_HWV = 9, OLB_PROT_CODE_SWV = 10, OLB_PROT_CODE_ESD = 11, OLB_PROT_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_SLICE_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_SLICE_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_TRANSFER_SLICE_1
• #define OLB_OSCI_TRIG_PWM_CH0_CCU_SLICE_COMPARE_CHANNEL XMC_CCU8_SLICE_COMPARE_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_TRANSFER_SLICE_0`
- `#define OLB_OSCI_TRIG_PWM_CH1_CCU_SLICE_COMPARE_CHANNEL XMC_CCU8_SLICE_COMPARE_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

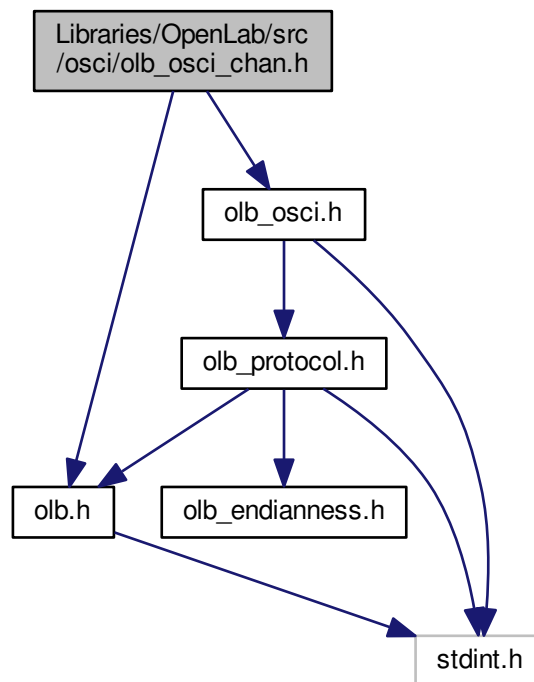
Aug 15, 2016

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

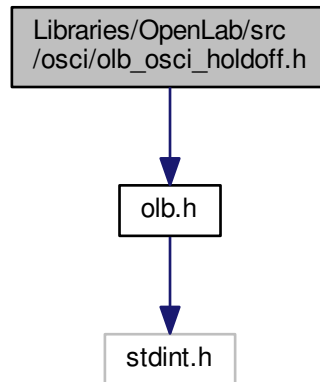
Author

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

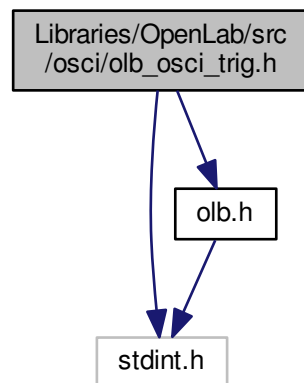
Author

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_smpl_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

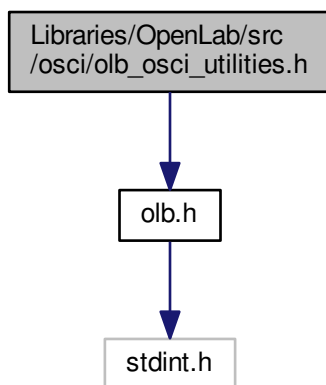
Author

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

__attribute__
Protocol, [35](#)

CRC16_CCIT, [39](#)

Fifo8, [13](#)
 olb_fifo8_clear, [13](#)
 olb_fifo8_count, [13](#)
 olb_fifo8_get_mem, [14](#)
 olb_fifo8_init, [14](#)
 olb_fifo8_pop, [14](#)
 olb_fifo8_push, [14](#)

fill_count
 Olb_fifo8_t, [39](#)

HoldOff, [38](#)

Libraries/OpenLab/inc/CRC16_CCIT.h, [43](#)
Libraries/OpenLab/inc/olb.h, [44](#)
Libraries/OpenLab/inc/olb_protocol.h, [45](#)
Libraries/OpenLab/src/osci/olb_osci_cfg.h, [47](#)
Libraries/OpenLab/src/osci/olb_osci_chan.h, [50](#)
Libraries/OpenLab/src/osci/olb_osci_holdoff.h, [51](#)
Libraries/OpenLab/src/osci/olb_osci_trig.h, [52](#)
Libraries/OpenLab/src/osci/olb_osci_utilities.h, [53](#)

mem
 Olb_fifo8_t, [39](#)

OLB_EAGAIN
 OpenLab, [10](#)

OLB_EFAIL
 OpenLab, [10](#)

OLB_ENOT_SUPPORTED
 OpenLab, [10](#)

OLB_EOVERFLOW
 OpenLab, [10](#)

OLB_EOK
 OpenLab, [10](#)

OLB_ETOUT
 OpenLab, [10](#)

OLB_EUNDERFLOW
 OpenLab, [10](#)

OLB_EUNKNOWN_COMMAND
 OpenLab, [10](#)

OLB_EWARNING
 OpenLab, [10](#)

OLB_OSCI_COUPLING_AC
 Oscilloscope, [20](#)

OLB_OSCI_COUPLING_DC
 Oscilloscope, [20](#)

OLB_OSCI_COUPLING_OFF
 Oscilloscope, [20](#)

OLB_OSCI_EVT_SAMPLING_COMPLETE
 Oscilloscope, [20](#)

OLB_OSCI_EVT_TRIGGER_DETECT
 Oscilloscope, [20](#)

OLB_OSCI_MODE_REALT
 Oscilloscope, [20](#)

OLB_OSCI_MODE_RETS
 Oscilloscope, [20](#)

OLB_OSCI_MODE_SETS
 Oscilloscope, [20](#)

OLB_OSCI_TRIG_ACT_FLANK_BOTH
 Oscilloscope, [20](#)

OLB_OSCI_TRIG_ACT_FLANK_FALLING
 Oscilloscope, [20](#)

OLB_OSCI_TRIG_ACT_FLANK_NONE
 Oscilloscope, [20](#)

OLB_OSCI_TRIG_ACT_FLANK_RISING
 Oscilloscope, [20](#)

OLB_OSCI_TRIG_MODE_AUTO
 Oscilloscope, [21](#)

OLB_OSCI_TRIG_MODE_NORMAL
 Oscilloscope, [21](#)

OLB_PROT_CODE_ACK
 Protocol, [34](#)

OLB_PROT_CODE_ESD
 Protocol, [34](#)

OLB_PROT_CODE_GHWV
 Protocol, [34](#)

OLB_PROT_CODE_GSWV
 Protocol, [34](#)

OLB_PROT_CODE_HWV
 Protocol, [34](#)

OLB_PROT_CODE_NACK
 Protocol, [34](#)

OLB_PROT_CODE_SCS
 Protocol, [34](#)

OLB_PROT_CODE_SLB
 Protocol, [34](#)

OLB_PROT_CODE_SSM
 Protocol, [34](#)

OLB_PROT_CODE_STB
 Protocol, [34](#)

OLB_PROT_CODE_STS
 Protocol, [34](#)

OLB_PROT_CODE_SWV
 Protocol, [34](#)

OLB_PROT_CODE_SD

- Protocol, [34](#)
- OLB_PROT_DEV_ANY
 - Protocol, [34](#)
- OLB_PROT_DEV_OSCI
 - Protocol, [34](#)
- OLB_PROT_DEV_SIG
 - Protocol, [34](#)
- OLB_PROT_ERR_FAILED
 - Protocol, [34](#)
- OLB_PROT_ERR_NOT_SUPPORTED
 - Protocol, [34](#)
- OLB_PROT_ERR_UNKNOWN
 - Protocol, [34](#)
- OLB_PROT_ERR_WARN
 - Protocol, [34](#)
- OLB_PROT_OVERFLOW
 - Protocol, [34](#)
- Olb_error_t
 - OpenLab, [10](#)
- olb_fifo8_clear
 - Fifo8, [13](#)
- olb_fifo8_count
 - Fifo8, [13](#)
- olb_fifo8_get_mem
 - Fifo8, [14](#)
- olb_fifo8_init
 - Fifo8, [14](#)
- olb_fifo8_pop
 - Fifo8, [14](#)
- olb_fifo8_push
 - Fifo8, [14](#)
- Olb_fifo8_t, [39](#)
 - fill_count, [39](#)
 - mem, [39](#)
 - size, [40](#)
 - w_i, [40](#)
- olb_init
 - OpenLab, [10](#)
- olb_osci_chan_abort
 - Oscilloscope, [21](#)
- olb_osci_chan_arm
 - Oscilloscope, [21](#)
- olb_osci_chan_disable
 - Oscilloscope, [21](#)
- olb_osci_chan_enable
 - Oscilloscope, [21](#)
- olb_osci_chan_has_samples
 - Oscilloscope, [22](#)
- olb_osci_chan_init
 - Oscilloscope, [22](#)
- Olb_osci_chan_input_cfg_t, [40](#)
- olb_osci_chan_is_busy
 - Oscilloscope, [22](#)
- olb_osci_chan_is_enabled
 - Oscilloscope, [22](#)
- olb_osci_chan_reset_smpl_data
 - Oscilloscope, [23](#)
- olb_osci_chan_set_sample_buffer
 - Oscilloscope, [23](#)
- olb_osci_chan_set_sample_count
 - Oscilloscope, [23](#)
- olb_osci_chan_set_trig_src
 - Oscilloscope, [24](#)
- olb_osci_chan_write_to_smpl_buf
 - Oscilloscope, [24](#)
- Olb_osci_coupling_t
 - Oscilloscope, [20](#)
- Olb_osci_event_t
 - Oscilloscope, [20](#)
- olb_osci_get_samples
 - Oscilloscope, [24](#)
- olb_osci_init
 - Oscilloscope, [24](#)
- olb_osci_init_1kHz
 - Oscilloscope, [25](#)
- olb_osci_init_gain
 - Oscilloscope, [25](#)
- olb_osci_init_holdoff
 - Oscilloscope, [25](#)
- Olb_osci_mode_t
 - Oscilloscope, [20](#)
- olb_osci_prot_pkt_hdl
 - Oscilloscope, [25](#)
- olb_osci_raise_evt
 - Oscilloscope, [25](#)
- Olb_osci_smpl_mode_cfg_t, [40](#)
- olb_osci_set_channel_input_cfg
 - Oscilloscope, [26](#)
- Olb_osci_set_gain
 - Oscilloscope, [26](#)
- olb_osci_set_holdoff
 - Oscilloscope, [26](#)
- olb_osci_set_sample_rate
 - Oscilloscope, [27](#)
- olb_osci_set_sampling_cfg
 - Oscilloscope, [27](#)
- olb_osci_set_trigger_cfg
 - Oscilloscope, [27](#)
- olb_osci_start
 - Oscilloscope, [28](#)
- olb_osci_start_holdoff
 - Oscilloscope, [28](#)
- olb_osci_start_holdoff_cb
 - Oscilloscope, [28](#)
- olb_osci_stop
 - Oscilloscope, [28](#)
- olb_osci_task
 - Oscilloscope, [29](#)
- olb_osci_timeout_cb
 - Oscilloscope, [29](#)
- olb_osci_tmr_global_init
 - Oscilloscope, [29](#)
- Olb_osci_trig_active_flank_t
 - Oscilloscope, [20](#)
- olb_osci_trig_arm
 - Oscilloscope, [29](#)

- Olb_osci_trig_cfg_t, [41](#)
- olb_osci_trig_init
 - Oscilloscope, [30](#)
- Olb_osci_trig_mode_t
 - Oscilloscope, [20](#)
- olb_osci_trig_set_cfg
 - Oscilloscope, [30](#)
- olb_osci_trig_set_lvl
 - Oscilloscope, [30](#)
- olb_osci_trig_set_sample_rate
 - Oscilloscope, [30](#)
- olb_osci_trig_set_smpl_mode
 - Oscilloscope, [31](#)
- olb_osci_trig_stop
 - Oscilloscope, [31](#)
- olb_osci_wait
 - Oscilloscope, [31](#)
- olb_osci_wait_holdoff
 - Oscilloscope, [32](#)
- olb_prot_decode
 - Protocol, [35](#)
- olb_prot_dispatch
 - Protocol, [35](#)
- olb_prot_ghwv_handler
 - Protocol, [35](#)
- olb_prot_gswv_handler
 - Protocol, [35](#)
- olb_prot_init
 - Protocol, [36](#)
- olb_prot_register_hdl
 - Protocol, [36](#)
- olb_prot_send_ack
 - Protocol, [36](#)
- olb_prot_send_nack
 - Protocol, [36](#)
- olb_prot_send_pkt
 - Protocol, [37](#)
- Olb_protocol_code_t
 - Protocol, [34](#)
- Olb_protocol_dev_type_t
 - Protocol, [34](#)
- Olb_protocol_err_t
 - Protocol, [34](#)
- olb_receive
 - OpenLab, [10](#)
- olb_send
 - OpenLab, [11](#)
- olb_task
 - OpenLab, [11](#)
- olb_wait
 - OpenLab, [11](#)
- OpenLab, [9](#)
 - OLB_EAGAIN, [10](#)
 - OLB_EFAIL, [10](#)
 - OLB_ENOT_SUPPORTED, [10](#)
 - OLB_EOVERFLOW, [10](#)
 - OLB_EOK, [10](#)
 - OLB_ETOUT, [10](#)
 - OLB_EUNDERFLOW, [10](#)
 - OLB_EUNKNOWN_COMMAND, [10](#)
 - OLB_EWARNING, [10](#)
 - Olb_error_t, [10](#)
 - olb_init, [10](#)
 - olb_receive, [10](#)
 - olb_send, [11](#)
 - olb_task, [11](#)
 - olb_wait, [11](#)
- Oscilloscope, [16](#)
 - OLB_OSCI_COUPLING_AC, [20](#)
 - OLB_OSCI_COUPLING_DC, [20](#)
 - OLB_OSCI_COUPLING_OFF, [20](#)
 - OLB_OSCI_EVT_SAMPLING_COMPLETE, [20](#)
 - OLB_OSCI_EVT_TRIGGER_DETECT, [20](#)
 - OLB_OSCI_MODE_REALT, [20](#)
 - OLB_OSCI_MODE_RETS, [20](#)
 - OLB_OSCI_MODE_SETS, [20](#)
 - OLB_OSCI_TRIG_ACT_FLANK_BOTH, [20](#)
 - OLB_OSCI_TRIG_ACT_FLANK_FALLING, [20](#)
 - OLB_OSCI_TRIG_ACT_FLANK_NONE, [20](#)
 - OLB_OSCI_TRIG_ACT_FLANK_RISING, [20](#)
 - OLB_OSCI_TRIG_MODE_AUTO, [21](#)
 - OLB_OSCI_TRIG_MODE_NORMAL, [21](#)
 - olb_osci_chan_abort, [21](#)
 - olb_osci_chan_arm, [21](#)
 - olb_osci_chan_disable, [21](#)
 - olb_osci_chan_enable, [21](#)
 - olb_osci_chan_has_samples, [22](#)
 - olb_osci_chan_init, [22](#)
 - olb_osci_chan_is_busy, [22](#)
 - olb_osci_chan_is_enabled, [22](#)
 - olb_osci_chan_reset_smpl_data, [23](#)
 - olb_osci_chan_set_sample_buffer, [23](#)
 - olb_osci_chan_set_sample_count, [23](#)
 - olb_osci_chan_set_trig_src, [24](#)
 - olb_osci_chan_write_to_smpl_buf, [24](#)
 - Olb_osci_coupling_t, [20](#)
 - Olb_osci_event_t, [20](#)
 - olb_osci_get_samples, [24](#)
 - olb_osci_init, [24](#)
 - olb_osci_init_1kHz, [25](#)
 - olb_osci_init_gain, [25](#)
 - olb_osci_init_holdoff, [25](#)
 - Olb_osci_mode_t, [20](#)
 - olb_osci_prot_pkt_hdl, [25](#)
 - olb_osci_raise_evt, [25](#)
 - olb_osci_set_channel_input_cfg, [26](#)
 - Olb_osci_set_gain, [26](#)
 - olb_osci_set_holdoff, [26](#)
 - olb_osci_set_sample_rate, [27](#)
 - olb_osci_set_sampling_cfg, [27](#)
 - olb_osci_set_trigger_cfg, [27](#)
 - olb_osci_start, [28](#)
 - olb_osci_start_holdoff, [28](#)
 - olb_osci_start_holdoff_cb, [28](#)
 - olb_osci_stop, [28](#)
 - olb_osci_task, [29](#)

- [olb_osci_timeout_cb](#), 29
- [olb_osci_tmr_global_init](#), 29
- [Olb_osci_trig_active_flank_t](#), 20
- [olb_osci_trig_arm](#), 29
- [olb_osci_trig_init](#), 30
- [Olb_osci_trig_mode_t](#), 20
- [olb_osci_trig_set_cfg](#), 30
- [olb_osci_trig_set_lvl](#), 30
- [olb_osci_trig_set_sample_rate](#), 30
- [olb_osci_trig_set_smpl_mode](#), 31
- [olb_osci_trig_stop](#), 31
- [olb_osci_wait](#), 31
- [olb_osci_wait_holdoff](#), 32

Protocol, 33

- [__attribute__](#), 35
- [OLB_PROT_CODE_ACK](#), 34
- [OLB_PROT_CODE_ESD](#), 34
- [OLB_PROT_CODE_GHWV](#), 34
- [OLB_PROT_CODE_GSWV](#), 34
- [OLB_PROT_CODE_HWV](#), 34
- [OLB_PROT_CODE_NACK](#), 34
- [OLB_PROT_CODE_SCS](#), 34
- [OLB_PROT_CODE_SLB](#), 34
- [OLB_PROT_CODE_SSM](#), 34
- [OLB_PROT_CODE_STB](#), 34
- [OLB_PROT_CODE_STS](#), 34
- [OLB_PROT_CODE_SWV](#), 34
- [OLB_PROT_CODE_SD](#), 34
- [OLB_PROT_DEV_ANY](#), 34
- [OLB_PROT_DEV_OSCI](#), 34
- [OLB_PROT_DEV_SIG](#), 34
- [OLB_PROT_ERR_FAILED](#), 34
- [OLB_PROT_ERR_NOT_SUPPORTED](#), 34
- [OLB_PROT_ERR_UNKNOWN](#), 34
- [OLB_PROT_ERR_WARN](#), 34
- [OLB_PROT_OVERFLOW](#), 34
- [olb_prot_decode](#), 35
- [olb_prot_dispatch](#), 35
- [olb_prot_ghwv_handler](#), 35
- [olb_prot_gswv_handler](#), 35
- [olb_prot_init](#), 36
- [olb_prot_register_hdl](#), 36
- [olb_prot_send_ack](#), 36
- [olb_prot_send_nack](#), 36
- [olb_prot_send_pkt](#), 37
- [Olb_protocol_code_t](#), 34
- [Olb_protocol_dev_type_t](#), 34
- [Olb_protocol_err_t](#), 34

size

- [Olb_fifo8_t](#), 40

Utilities, 12

w_i

- [Olb_fifo8_t](#), 40