

# libswd

## 0.0.1

Generated by Doxygen 1.7.1

Sun Feb 6 2011 23:08:40



# Contents

<b>1</b>	<b>Serial Wire Debug Open Library.</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	swd_ahbap_t Struct Reference . . . . .	7
4.1.1	Member Data Documentation . . . . .	8
4.1.1.1	bd0 . . . . .	8
4.1.1.2	bd1 . . . . .	8
4.1.1.3	bd2 . . . . .	8
4.1.1.4	bd3 . . . . .	8
4.1.1.5	controlstatus . . . . .	8
4.1.1.6	dromt . . . . .	8
4.1.1.7	drw . . . . .	8
4.1.1.8	idr . . . . .	8
4.1.1.9	tar . . . . .	8
4.2	swd_cmd_t Struct Reference . . . . .	8
4.2.1	Member Data Documentation . . . . .	9
4.2.1.1	"@1 . . . . .	9
4.2.1.2	ack . . . . .	9
4.2.1.3	bits . . . . .	9
4.2.1.4	cmdtype . . . . .	9
4.2.1.5	control . . . . .	9
4.2.1.6	done . . . . .	9
4.2.1.7	misobit . . . . .	9

4.2.1.8	misodata	9
4.2.1.9	mosibit	9
4.2.1.10	mosidata	9
4.2.1.11	next	9
4.2.1.12	parity	9
4.2.1.13	prev	9
4.2.1.14	request	9
4.2.1.15	TRNnMOSI	9
4.3	swd_context_config_t Struct Reference	9
4.3.1	Member Data Documentation	10
4.3.1.1	initialized	10
4.3.1.2	loglevel	10
4.3.1.3	maxcmdqlen	10
4.3.1.4	trnlen	10
4.4	swd_ctx_t Struct Reference	10
4.4.1	Member Data Documentation	10
4.4.1.1	cmdq	10
4.4.1.2	config	10
4.4.1.3	driver	10
4.4.1.4	misoahbap	10
4.4.1.5	misoswdp	10
4.4.1.6	mosiahbap	10
4.4.1.7	mosiswdp	10
4.5	swd_driver_t Struct Reference	11
4.5.1	Member Data Documentation	11
4.5.1.1	device	11
4.6	swd_swdp_t Struct Reference	11
4.6.1	Member Data Documentation	11
4.6.1.1	abort	11
4.6.1.2	ack	11
4.6.1.3	ctrlstat	11
4.6.1.4	idcode	11
4.6.1.5	rdbuf	11
4.6.1.6	select	11
4.6.1.7	wcr	11

5.1	libswd.c File Reference . . . . .	13
5.1.1	Function Documentation . . . . .	14
5.1.1.1	swd_bin32_bitswap . . . . .	14
5.1.1.2	swd_bin32_parity_even . . . . .	15
5.1.1.3	swd_bin32_print . . . . .	15
5.1.1.4	swd_bin32_string . . . . .	15
5.1.1.5	swd_bin8_bitswap . . . . .	15
5.1.1.6	swd_bin8_parity_even . . . . .	16
5.1.1.7	swd_bin8_print . . . . .	16
5.1.1.8	swd_bin8_string . . . . .	16
5.1.1.9	swd_bus_setdir_miso . . . . .	16
5.1.1.10	swd_bus_setdir_mosi . . . . .	17
5.1.1.11	swd_cmd_append_mosi_n_data_ap . . . . .	17
5.1.1.12	swd_cmd_append_mosi_n_data_p . . . . .	17
5.1.1.13	swd_cmd_queue_append . . . . .	17
5.1.1.14	swd_cmd_queue_append_jtag2swd . . . . .	18
5.1.1.15	swd_cmd_queue_append_miso_ack . . . . .	18
5.1.1.16	swd_cmd_queue_append_miso_data . . . . .	18
5.1.1.17	swd_cmd_queue_append_miso_data_p . . . . .	18
5.1.1.18	swd_cmd_queue_append_miso_n_data_p . . . . .	19
5.1.1.19	swd_cmd_queue_append_miso_nbit . . . . .	19
5.1.1.20	swd_cmd_queue_append_miso_parity . . . . .	19
5.1.1.21	swd_cmd_queue_append_miso_trn . . . . .	19
5.1.1.22	swd_cmd_queue_append_mosi_control . . . . .	20
5.1.1.23	swd_cmd_queue_append_mosi_data . . . . .	20
5.1.1.24	swd_cmd_queue_append_mosi_data_ap . . . . .	20
5.1.1.25	swd_cmd_queue_append_mosi_data_p . . . . .	21
5.1.1.26	swd_cmd_queue_append_mosi_nbit . . . . .	21
5.1.1.27	swd_cmd_queue_append_mosi_parity . . . . .	21
5.1.1.28	swd_cmd_queue_append_mosi_request . . . . .	21
5.1.1.29	swd_cmd_queue_append_mosi_trn . . . . .	22
5.1.1.30	swd_cmd_queue_append_swd2jtag . . . . .	22
5.1.1.31	swd_cmd_queue_append_swdpreset . . . . .	22
5.1.1.32	swd_cmd_queue_find_root . . . . .	22
5.1.1.33	swd_cmd_queue_find_tail . . . . .	23
5.1.1.34	swd_cmd_queue_flush . . . . .	23

5.1.1.35	swd_cmd_queue_free	23
5.1.1.36	swd_cmd_queue_free_head	23
5.1.1.37	swd_cmd_queue_free_tail	24
5.1.1.38	swd_cmd_queue_init	24
5.1.1.39	swd_deinit	24
5.1.1.40	swd_deinit_cmdq	24
5.1.1.41	swd_deinit_ctx	25
5.1.1.42	swd_error_string	25
5.1.1.43	swd_idcode	25
5.1.1.44	swd_init	25
5.1.1.45	swd_log	25
5.1.1.46	swd_miso_ack	25
5.1.1.47	swd_miso_data_p	26
5.1.1.48	swd_mosi_data_ap	26
5.1.1.49	swd_mosi_data_p	26
5.1.1.50	swd_mosi_request	27
5.2	libswd.h File Reference	27
5.2.1	Define Documentation	33
5.2.1.1	AHB_AP_BD0	33
5.2.1.2	AHB_AP_BD1	33
5.2.1.3	AHB_AP_BD2	33
5.2.1.4	AHB_AP_BD3	33
5.2.1.5	AHB_AP_CONTROLSTATUS	33
5.2.1.6	AHB_AP_DROMT	33
5.2.1.7	AHB_AP_DRW	33
5.2.1.8	AHB_AP_IDR	33
5.2.1.9	AHB_AP_TAR	33
5.2.1.10	SWD_ABORT_BITNUM_DAPABORT	33
5.2.1.11	SWD_ABORT_BITNUM_DORUNERRCLR	33
5.2.1.12	SWD_ABORT_BITNUM_DSTKCMPLR	33
5.2.1.13	SWD_ABORT_BITNUM_DSTKERRCLR	33
5.2.1.14	SWD_ABORT_BITNUM_DWDERRCLR	33
5.2.1.15	SWD_ACK_BITLEN	33
5.2.1.16	SWD_ACK_FAULT	33
5.2.1.17	SWD_ACK_OK	33
5.2.1.18	SWD_ACK_WAIT	33

5.2.1.19	SWD_ADDR_MAXVAL	33
5.2.1.20	SWD_ADDR_MINVAL	33
5.2.1.21	SWD_CMDQLEN_DEFAULT	33
5.2.1.22	SWD_CTRLSTAT_BITNUM_OCDBGPWRUPACK	33
5.2.1.23	SWD_CTRLSTAT_BITNUM_OCDBGPWRUPREQ	33
5.2.1.24	SWD_CTRLSTAT_BITNUM_OCDBGGRSTACK	33
5.2.1.25	SWD_CTRLSTAT_BITNUM_OCDBGGRSTREQ	33
5.2.1.26	SWD_CTRLSTAT_BITNUM_OCSYSPWRUPACK	33
5.2.1.27	SWD_CTRLSTAT_BITNUM_OCSYSPWRUPREQ	33
5.2.1.28	SWD_CTRLSTAT_BITNUM_OMASKLANE	33
5.2.1.29	SWD_CTRLSTAT_BITNUM_OREADOK	33
5.2.1.30	SWD_CTRLSTAT_BITNUM_ORUNDETECT	33
5.2.1.31	SWD_CTRLSTAT_BITNUM_OSTICKYCMP	33
5.2.1.32	SWD_CTRLSTAT_BITNUM_OSTICKYERR	33
5.2.1.33	SWD_CTRLSTAT_BITNUM_OSTICKYORUN	33
5.2.1.34	SWD_CTRLSTAT_BITNUM_OTRNCNT	33
5.2.1.35	SWD_CTRLSTAT_BITNUM_OTRNMODE	33
5.2.1.36	SWD_CTRLSTAT_BITNUM_OWDATAERR	33
5.2.1.37	SWD_DATA_BITLEN	33
5.2.1.38	SWD_DATA_BYTESIZE	33
5.2.1.39	SWD_DATA_MAXBITCOUNT	33
5.2.1.40	SWD_DP_ADDR_ABORT	33
5.2.1.41	SWD_DP_ADDR_CRTLSTAT	33
5.2.1.42	SWD_DP_ADDR_IDCODE	33
5.2.1.43	SWD_DP_ADDR_RDBUF	33
5.2.1.44	SWD_DP_ADDR_RESEND	33
5.2.1.45	SWD_DP_ADDR_SELECT	33
5.2.1.46	SWD_DP_ADDR_WCR	33
5.2.1.47	SWD_MASKLANE_0	33
5.2.1.48	SWD_MASKLANE_1	33
5.2.1.49	SWD_MASKLANE_2	33
5.2.1.50	SWD_MASKLANE_3	33
5.2.1.51	SWD_REQUEST_A2_BITNUM	33
5.2.1.52	SWD_REQUEST_A3_BITNUM	33
5.2.1.53	SWD_REQUEST_ADDR_BITNUM	33
5.2.1.54	SWD_REQUEST_APhDP_BITNUM	33

5.2.1.55	SWD_REQUEST_BITLEN . . . . .	33
5.2.1.56	SWD_REQUEST_PARITY_BITNUM . . . . .	33
5.2.1.57	SWD_REQUEST_PARK_BITNUM . . . . .	33
5.2.1.58	SWD_REQUEST_PARK_VAL . . . . .	33
5.2.1.59	SWD_REQUEST_RnW_BITNUM . . . . .	33
5.2.1.60	SWD_REQUEST_START_BITNUM . . . . .	33
5.2.1.61	SWD_REQUEST_START_VAL . . . . .	33
5.2.1.62	SWD_REQUEST_STOP_BITNUM . . . . .	33
5.2.1.63	SWD_REQUEST_STOP_VAL . . . . .	33
5.2.1.64	SWD_SELECT_BITNUM_APBANKSEL . . . . .	33
5.2.1.65	SWD_SELECT_BITNUM_APSEL . . . . .	33
5.2.1.66	SWD_SELECT_BITNUM_CTRLSEL . . . . .	33
5.2.1.67	SWD_TURNROUND_1 . . . . .	33
5.2.1.68	SWD_TURNROUND_2 . . . . .	33
5.2.1.69	SWD_TURNROUND_3 . . . . .	33
5.2.1.70	SWD_TURNROUND_4 . . . . .	33
5.2.1.71	SWD_TURNROUND_DEFAULT . . . . .	33
5.2.1.72	SWD_TURNROUND_MAX . . . . .	33
5.2.1.73	SWD_TURNROUND_MIN . . . . .	33
5.2.1.74	SWD_WCR_BITNUM_PRESCALER . . . . .	33
5.2.1.75	SWD_WCR_BITNUM_TURNROUND . . . . .	33
5.2.1.76	SWD_WCR_BITNUM_WIREMODE . . . . .	33
5.2.2	Typedef Documentation . . . . .	33
5.2.2.1	swd_cmd_t . . . . .	33
5.2.2.2	swd_cmdtype_t . . . . .	33
5.2.2.3	swd_debuglevel_t . . . . .	33
5.2.2.4	swd_direction_t . . . . .	33
5.2.2.5	swd_error_code_t . . . . .	33
5.2.2.6	swd_loglevel_t . . . . .	33
5.2.2.7	swd_operation_t . . . . .	33
5.2.3	Enumeration Type Documentation . . . . .	33
5.2.3.1	swd_bool_t . . . . .	33
5.2.3.2	SWD_CMDTYPE . . . . .	34
5.2.3.3	SWD_DEBUGLEVEL . . . . .	34
5.2.3.4	SWD_DIRECTION . . . . .	34
5.2.3.5	SWD_ERROR_CODE . . . . .	34



5.2.3.6	SWD_LOGLEVEL	35
5.2.3.7	SWD_OPERATION	36
5.2.4	Function Documentation	36
5.2.4.1	swd_bin32_bitswap	36
5.2.4.2	swd_bin32_parity_even	36
5.2.4.3	swd_bin32_print	36
5.2.4.4	swd_bin32_string	37
5.2.4.5	swd_bin8_bitswap	37
5.2.4.6	swd_bin8_parity_even	37
5.2.4.7	swd_bin8_print	38
5.2.4.8	swd_bin8_string	38
5.2.4.9	swd_bit8_gen_request	38
5.2.4.10	swd_bus_setdir_miso	38
5.2.4.11	swd_bus_setdir_mosi	38
5.2.4.12	swd_cmd_append_mosi_n_data_ap	39
5.2.4.13	swd_cmd_append_mosi_n_data_p	39
5.2.4.14	swd_cmd_queue_append	39
5.2.4.15	swd_cmd_queue_append_jtag2swd	39
5.2.4.16	swd_cmd_queue_append_miso_ack	40
5.2.4.17	swd_cmd_queue_append_miso_data	40
5.2.4.18	swd_cmd_queue_append_miso_data_p	40
5.2.4.19	swd_cmd_queue_append_miso_n_data_p	40
5.2.4.20	swd_cmd_queue_append_miso_nbit	41
5.2.4.21	swd_cmd_queue_append_miso_parity	41
5.2.4.22	swd_cmd_queue_append_miso_trn	41
5.2.4.23	swd_cmd_queue_append_mosi_control	42
5.2.4.24	swd_cmd_queue_append_mosi_data	42
5.2.4.25	swd_cmd_queue_append_mosi_data_ap	42
5.2.4.26	swd_cmd_queue_append_mosi_data_p	42
5.2.4.27	swd_cmd_queue_append_mosi_nbit	43
5.2.4.28	swd_cmd_queue_append_mosi_parity	43
5.2.4.29	swd_cmd_queue_append_mosi_request	43
5.2.4.30	swd_cmd_queue_append_mosi_trn	44
5.2.4.31	swd_cmd_queue_append_swd2jtag	44
5.2.4.32	swd_cmd_queue_append_swdpreset	44
5.2.4.33	swd_cmd_queue_find_end	44

5.2.4.34	swd_cmd_queue_find_root	44
5.2.4.35	swd_cmd_queue_flush	45
5.2.4.36	swd_cmd_queue_free	45
5.2.4.37	swd_cmd_queue_free_head	45
5.2.4.38	swd_cmd_queue_free_tail	45
5.2.4.39	swd_cmd_queue_init	46
5.2.4.40	swd_deinit	46
5.2.4.41	swd_deinit_cmdq	46
5.2.4.42	swd_deinit_ctx	46
5.2.4.43	swd_drv_miso_32	47
5.2.4.44	swd_drv_miso_8	47
5.2.4.45	swd_drv_miso_trn	47
5.2.4.46	swd_drv_mosi_32	47
5.2.4.47	swd_drv_mosi_8	47
5.2.4.48	swd_drv_mosi_trn	47
5.2.4.49	swd_error_string	47
5.2.4.50	swd_idcode	47
5.2.4.51	swd_init	47
5.2.4.52	swd_log	47
5.2.4.53	swd_miso_ack	47
5.2.4.54	swd_miso_data_p	48
5.2.4.55	swd_mosi_data_ap	48
5.2.4.56	swd_mosi_data_p	48
5.2.4.57	swd_mosi_request	49
5.2.4.58	swd_transfer_cmd	49
5.2.4.59	swd_transmit	49
5.3	libswd_drv_dummy.c File Reference	49
5.3.1	Function Documentation	50
5.3.1.1	swd_drv_miso_32	50
5.3.1.2	swd_drv_miso_8	50
5.3.1.3	swd_drv_miso_trn	50
5.3.1.4	swd_drv_mosi_32	50
5.3.1.5	swd_drv_mosi_8	50
5.3.1.6	swd_drv_mosi_trn	50
5.4	libswd_drv_urjtag.c File Reference	50
5.4.1	Function Documentation	50

---

5.4.1.1	swd_drv_miso_32 . . . . .	50
5.4.1.2	swd_drv_miso_8 . . . . .	50
5.4.1.3	swd_drv_miso_trn . . . . .	50
5.4.1.4	swd_drv_mosi_32 . . . . .	50
5.4.1.5	swd_drv_mosi_8 . . . . .	50
5.4.1.6	swd_drv_mosi_trn . . . . .	50
5.5	libswd_test.c File Reference . . . . .	50
5.5.1	Function Documentation . . . . .	51
5.5.1.1	main . . . . .	51



# Chapter 1

## Serial Wire Debug Open Library.

### 1.1 Introduction

Welcome to the source code documentation repository. LibSWD is an Open-Source framework to deal with Serial Wire Debug. It is released under 3-clause BSD license. For more information please visit project website at <http://libswd.sf.net>

### 1.2 What is this about

Serial Wire Debug is an alternative to JTAG (IEEE1149.1) transport layer to access Debug Access Port in ARM-Cortex's based devices. LibSWD provides both bitstream generation and high/low level bus operations. Every bus operation such as request, turnaround, acknowledge, data and parity packet is represented by a `swd_cmd_t` element that can extend command queue (a standard bidirectional queue) that later can be flushed into real hardware using simple set of interface-specific driver functions. This way LibSWD is almost standalone and can be easily integrated into existing utilities for low-level access and only requires in return to define drivers that controls the interface interconnecting host and target. Such drivers are application specific therefore located in external file crafted for that application and its hardware.



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">swd_ahbap_t</a> (Most actual Advanced High Bandwidth Access Peripheral Bus Registers ) . . . .	7
<a href="#">swd_cmd_t</a> (SWD Command Element Structure ) . . . . .	8
<a href="#">swd_context_config_t</a> (Context configuration structure ) . . . . .	9
<a href="#">swd_ctx_t</a> (SWD Context Structure definition ) . . . . .	10
<a href="#">swd_driver_t</a> (Interface Driver structure ) . . . . .	11
<a href="#">swd_swdp_t</a> (Most actual Serial Wire Debug Port Registers ) . . . . .	11





# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">libswd.c</a>	13
<a href="#">libswd.h</a>	27



# Chapter 4

## Class Documentation

### 4.1 swd\_ahbap\_t Struct Reference

Most actual Advanced High Bandwidth Access Peripheral Bus Registers.

```
#include <libswd.h>
```

#### Public Attributes

- int [controlstatus](#)  
*Last known CONTROLSTATUS register value.*
- int [tar](#)  
*Last known TAR register value.*
- int [drw](#)  
*Last known DRW register value.*
- int [bd0](#)  
*Last known BD0 register value.*
- int [bd1](#)  
*Last known BD1 register value.*
- int [bd2](#)  
*Last known BD2 register value.*
- int [bd3](#)  
*Last known BD3 register value.*
- int [dromt](#)  
*Last known DROMT register value.*
- int [idr](#)  
*Last known IDR register value.*

### 4.1.1 Detailed Description

Most actual Advanced High Bandwidth Access Peripheral Bus Reisters.

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

- [libswd.h](#)

## 4.2 swd\_cmd\_t Struct Reference

SWD Command Element Structure.

```
#include <libswd.h>
```

### Public Attributes

- union {
  - char [TRNnMOSI](#)  
*< Payload data union.*
  - char [request](#)  
*Request header data.*
  - char [ack](#)  
*Acknowledge response from target.*
  - int [misodata](#)  
*Data read from target (MISO).*
  - int [mosidata](#)  
*Data written to target (MOSI).*
  - char [misobit](#)  
*Single bit read from target (bit-per-char).*
  - char [mosibit](#)  
*Single bit written to target (bit-per-char).*
  - char [parity](#)  
*Parity bit for data payload.*
  - char [control](#)  
*Control transfer data (one byte).*
- };
- char [bits](#)  
*Payload bit count == clk pulses on the bus.*
- char [cmdtype](#)  
*Command type as defined by swd\_cmdtype\_t.*
- char [done](#)  
*Non-zero if operation already executed.*
- struct [swd\\_cmd\\_t](#) \* [prev](#)
- struct [swd\\_cmd\\_t](#) \* [next](#)  
*Pointer to the previous/next command.*

### 4.2.1 Detailed Description

SWD Command Element Structure. In libswd each operation is split into separate commands (request, trn, ack, data, parity) that can be appended to the command queue and later executed. This organization allows better granularity for tracing bugs and makes possible to compose complete bus/target operations made of simple commands.

### 4.2.2 Member Data Documentation

#### 4.2.2.1 char swd\_cmd\_t::TRNnMOSI

< Payload data union.

Holds/sets bus direction: MOSI when zero, MISO for other.

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

- [libswd.h](#)

## 4.3 swd\_context\_config\_t Struct Reference

Context configuration structure.

```
#include <libswd.h>
```

### Public Attributes

- char [initialized](#)  
*Context must be initialized prior use.*
- char [trnlen](#)  
*How many CLK cycles will TRN use.*
- int [maxcmdqlen](#)  
*How long command queue can be.*
- [swd\\_loglevel\\_t](#) [loglevel](#)  
*Holds Logging Level setting.*

### 4.3.1 Detailed Description

Context configuration structure.

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

- [libswd.h](#)

## 4.4 swd\_ctx\_t Struct Reference

SWD Context Structure definition.

```
#include <libswd.h>
```

### Public Attributes

- [swd\\_cmd\\_t \\* cmdq](#)  
*Command queue, stores all bus operations.*
- [swd\\_context\\_config\\_t config](#)  
*Target specific configuration.*
- [swd\\_driver\\_t \\* driver](#)  
*Pointer to the interface driver structure.*
- [swd\\_swdp\\_t misoswdp](#)  
*Last known read from the SW-DP register.*
- [swd\\_swdp\\_t mosiswdp](#)  
*Last known write to the SW-DP register.*
- [swd\\_ahbap\\_t misoahbap](#)  
*Last known read from AHB-AP register.*
- [swd\\_ahbap\\_t mosiahbap](#)  
*Last known write to the AHB-AP register.*

### 4.4.1 Detailed Description

SWD Context Structure definition. It stores all the information about the library, drivers and interface configuration, target status along with DAP/AHBAP data/instruction internal registers, and the command queue. Bus operations are stored on the command queue. There may be more than one context in use by a host software, each one for single interface-target pair. Most of the target operations made with libswd are required to pass [swd\\_ctx\\_t](#) pointer structure that also remembers last known state of the target's internal registers.

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

- [libswd.h](#)

## 4.5 swd\_driver\_t Struct Reference

Interface Driver structure.

```
#include <libswd.h>
```

## Public Attributes

- void \* **device**

### 4.5.1 Detailed Description

Interface Driver structure. It holds pointer to the driver structure that keeps driver information necessary to work with the physical interface.

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

- [libswd.h](#)

## 4.6 swd\_swdp\_t Struct Reference

Most actual Serial Wire Debug Port Registers.

```
#include <libswd.h>
```

## Public Attributes

- char [ack](#)  
*Last known state of ACK response.*
- int [idcode](#)  
*Target's IDCODE register value.*
- int [abort](#)  
*Last known ABORT register value.*
- int [ctrlstat](#)  
*Last known CTRLSTAT register value.*
- int [wcr](#)  
*Last known WCR register value.*
- int [select](#)  
*Last known SELECT register value.*
- int [rdbuf](#)  
*Last known RDBUF register (payload data) value.*

### 4.6.1 Detailed Description

Most actual Serial Wire Debug Port Registers.

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

- [libswd.h](#)





# Chapter 5

## File Documentation

### 5.1 libswd.c File Reference

```
#include <urjtag/libswd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

#### Functions

- int [swd\\_bin8\\_parity\\_even](#) (char \*data, char \*parity)  
*Data parity calculator, calculates even parity on char type.*
- int [swd\\_bin32\\_parity\\_even](#) (int \*data, char \*parity)  
*Data parity calculator, calculates even parity on integer type.*
- int [swd\\_bin8\\_print](#) (char \*data)  
*Prints binary data of a char value on the screen.*
- int [swd\\_bin32\\_print](#) (int \*data)  
*Prints binary data of an integer value on the screen.*
- char \* [swd\\_bin8\\_string](#) (char \*data)  
*Generates string containing binary data of a char value.*
- char \* [swd\\_bin32\\_string](#) (int \*data)  
*Generates string containing binary data of an integer value.*
- int [swd\\_bin8\\_bitswap](#) (unsigned char \*buffer, int bitcount)  
*Bit swap helper function that reverse bit order in char \*buffer.*
- int [swd\\_bin32\\_bitswap](#) (unsigned int \*buffer, int bitcount)  
*Bit swap helper function that reverse bit order in int \*buffer.*

- `int swd_cmd_queue_init (swd_cmd_t *cmdq)`  
*Initialize new queue element in memory that becomes a queue root.*
- `swd_cmd_t * swd_cmd_queue_find_root (swd_cmd_t *cmdq)`  
*Find queue root (first element).*
- `swd_cmd_t * swd_cmd_queue_find_tail (swd_cmd_t *cmdq)`  
*Find queue tail (last element).*
- `int swd_cmd_queue_append (swd_cmd_t *cmdq, swd_cmd_t *cmd)`  
*Append element pointed by \*cmd at the end of the quque pointed by \*cmdq.*
- `int swd_cmd_queue_free (swd_cmd_t *cmdq)`  
*Free queue pointed by \*cmdq element.*
- `int swd_cmd_queue_free_head (swd_cmd_t *cmdq)`  
*Free queue head up to \*cmdq element.*
- `int swd_cmd_queue_free_tail (swd_cmd_t *cmdq)`  
*Free queue tail starting after \*cmdq element.*
- `int swd_cmd_queue_append_mosi_request (swd_ctx_t *swdctx, char *request)`  
*Appends command queue with SWD Request packet header.*
- `int swd_cmd_queue_append_mosi_trn (swd_ctx_t *swdctx)`  
*Append command queue with Turnaround activating MOSI mode.*
- `int swd_cmd_queue_append_miso_trn (swd_ctx_t *swdctx)`  
*Append command queue with Turnaround activating MISO mode.*
- `int swd_cmd_queue_append_miso_nbit (swd_ctx_t *swdctx, char **data, int count)`  
*Append command queue with bus binary read bit-by-bit operation.*
- `int swd_cmd_queue_append_mosi_nbit (swd_ctx_t *swdctx, char *data, int count)`  
*Append command queue with bus binary write bit-by-bit operation.*
- `int swd_cmd_queue_append_mosi_parity (swd_ctx_t *swdctx, char *parity)`  
*Append command queue with parity bit write.*
- `int swd_cmd_queue_append_miso_parity (swd_ctx_t *swdctx, char *parity)`  
*Append command queue with parity bit read.*
- `int swd_cmd_queue_append_miso_data (swd_ctx_t *swdctx, int *data)`  
*Append command queue with data read.*
- `int swd_cmd_queue_append_miso_data_p (swd_ctx_t *swdctx, int *data, char *parity)`  
*Append command queue with data and parity read.*
- `int swd_cmd_queue_append_miso_n_data_p (swd_ctx_t *swdctx, int **data, char **parity, int count)`

*Append command queue with series of data and parity read.*

- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data](#) (swd\_ctx\_t \*swdctx, int \*data)  
*Append command queue with data and parity write.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data\\_ap](#) (swd\_ctx\_t \*swdctx, int \*data)  
*Append command queue with data and automatic parity write.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*data, char \*parity)  
*Append command queue with data and provided parity write.*
- int [swd\\_cmd\\_append\\_mosi\\_n\\_data\\_ap](#) (swd\_ctx\_t \*swdctx, int \*\*data, int count)  
*Append command queue with series of data and automatic parity writes.*
- int [swd\\_cmd\\_append\\_mosi\\_n\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*\*data, char \*\*parity, int count)  
*Append command queue with series of data and provided parity writes.*
- int [swd\\_cmd\\_queue\\_append\\_miso\\_ack](#) (swd\_ctx\_t \*swdctx, char \*ack)  
*Append queue with ACK read.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_control](#) (swd\_ctx\_t \*swdctx, char \*ctlmsg, int len)  
*Append command queue with len-octet size control sequence.*
- int [swd\\_cmd\\_queue\\_append\\_swdpreset](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with SW-DP-RESET sequence.*
- int [swd\\_cmd\\_queue\\_append\\_jtag2swd](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with JTAG-TO-SWD DAP-switch sequence.*
- int [swd\\_cmd\\_queue\\_append\\_swdd2jtag](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with SWD-TO-JTAG DAP-switch sequence.*
- int [swd\\_bus\\_setdir\\_mosi](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with TRN WRITE/MOSI.*
- int [swd\\_bus\\_setdir\\_miso](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with TRN READ/MISO.*
- int [swd\\_cmd\\_queue\\_flush](#) (swd\_ctx\_t \*swdctx, swd\_operation\_t operation)  
*Flush command queue contents into interface driver.*
- int [swd\\_mosi\\_request](#) (swd\_ctx\_t \*swdctx, swd\_operation\_t operation, char \*APnDP, char \*RnW, char \*addr)  
*Perform Request.*
- int [swd\\_miso\\_ack](#) (swd\_ctx\_t \*swdctx, swd\_operation\_t operation, char \*ack)  
*Perform ACK read into \*ack and verify received data.*
- int [swd\\_mosi\\_data\\_p](#) (swd\_ctx\_t \*swdctx, swd\_operation\_t operation, int \*data, char \*parity)  
*Perform (MOSI) data write with provided parity value.*

- int `swd_mosi_data_ap` (`swd_ctx_t` \*swdctx, `swd_operation_t` operation, int \*data)  
*Perform (MOSI) data write with automatic parity calculation.*
- int `swd_miso_data_p` (`swd_ctx_t` \*swdctx, `swd_operation_t` operation, int \*data, char \*parity)  
*Perform (MISO) data read.*
- int `swd_idcode` (`swd_ctx_t` \*swdctx, `swd_operation_t` operation, int \*idcode, char \*ack, char \*parity)  
*Read target's IDCODE register value.*
- int `swd_log` (`swd_loglevel_t` loglevel, char \*msg)
- char \* `swd_error_string` (`swd_error_code_t` error)
- `swd_ctx_t` \* `swd_init` (void)  
*LibSWD initialization routine.*
- int `swd_deinit_ctx` (`swd_ctx_t` \*swdctx)  
*De-initialize selected swd context and free its memory.*
- int `swd_deinit_cmdq` (`swd_ctx_t` \*swdctx)  
*De-initialize command queue and free its memory on selected swd context.*
- int `swd_deinit` (`swd_ctx_t` \*swdctx)  
*De-initialize selected swd context and its command queue.*

### 5.1.1 Detailed Description

### 5.1.2 Function Documentation

#### 5.1.2.1 int `swd_bin32_bitswap` ( unsigned int \* *buffer*, int *bitcount* )

Bit swap helper function that reverse bit order in int \*buffer.

Most Significant Bit becomes Least Significant Bit. It is possible to swap only n-bits from int (32-bit) \*buffer.

#### Parameters

**\*buffer** unsigned char (32-bit) data pointer.

**bitcount** how many bits to swap.

#### Returns

swapped bit count (positive) or error code (negative).

#### 5.1.2.2 int `swd_bin32_parity_even` ( int \* *data*, char \* *parity* )

Data parity calculator, calculates even parity on integer type.

**Parameters**

*\*data* source data pointer.  
*\*parity* resulting data pointer.

**Returns**

negative value on error, 0 or 1 as parity result.

**5.1.2.3 int swd\_bin32\_print ( int \* data )**

Prints binary data of an integer value on the screen.

**Parameters**

*\*data* source data pointer.

**Returns**

number of characters printed.

**5.1.2.4 char\* swd\_bin32\_string ( int \* data )**

Generates string containing binary data of an integer value.

**Parameters**

*\*data* source data pointer.

**Returns**

pointer to the resulting string.

**5.1.2.5 int swd\_bin8\_bitswap ( unsigned char \* buffer, int bitcount )**

Bit swap helper function that reverse bit order in char \*buffer.

Most Significant Bit becomes Least Significant Bit. It is possible to swap only n-bits from char (8-bit) \*buffer.

**Parameters**

*\*buffer* unsigned char (8-bit) data pointer.  
*bitcount* how many bits to swap.

**Returns**

swapped bit count (positive) or error code (negative).

#### 5.1.2.6 int swd\_bin8\_parity\_even ( char \* *data*, char \* *parity* )

Data parity calculator, calculates even parity on char type.

##### Parameters

- \*data* source data pointer.
- \*parity* resulting data pointer.

##### Returns

negative value on error, 0 or 1 as parity result.

#### 5.1.2.7 int swd\_bin8\_print ( char \* *data* )

Prints binary data of a char value on the screen.

##### Parameters

- \*data* source data pointer.

##### Returns

number of characters printed.

#### 5.1.2.8 char\* swd\_bin8\_string ( char \* *data* )

Generates string containing binary data of a char value.

##### Parameters

- \*data* source data pointer.

##### Returns

pointer to the resulting string.

#### 5.1.2.9 int swd\_bus\_setdir\_miso ( swd\_ctx\_t \* *swdctx* )

Append command queue with TRN READ/MISO.

##### Parameters

- \*swdctx* swd context pointer.

##### Returns

number of elements appended, or SWD\_ERROR\_CODE on failure.

#### 5.1.2.10 int swd\_bus\_setdir\_mosi ( swd\_ctx\_t \* swdctx )

Append command queue with TRN WRITE/MOSI.

##### Parameters

*\*swdctx* swd context pointer.

##### Returns

number of elements appended, or SWD\_ERROR\_CODE on failure.

#### 5.1.2.11 int swd\_cmd\_append\_mosi\_n\_data\_ap ( swd\_ctx\_t \* swdctx, int \*\* data, int count )

Append command queue with series of data and automatic parity writes.

##### Parameters

*\*swdctx* swd context pointer.

*\*\*data* data value array pointer.

*count* number of (data+parity) elements to read.

##### Returns

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.12 int swd\_cmd\_append\_mosi\_n\_data\_p ( swd\_ctx\_t \* swdctx, int \*\* data, char \*\* parity, int count )

Append command queue with series of data and provided parity writes.

##### Parameters

*\*swdctx* swd context pointer.

*\*\*data* data value array pointer.

*\*\*parity* parity value array pointer.

*count* number of (data+parity) elements to read.

##### Returns

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.13 int swd\_cmd\_queue\_append ( swd\_cmd\_t \* cmdq, swd\_cmd\_t \* cmd )

Append element pointed by \*cmd at the end of the quque pointed by \*cmdq.

##### Parameters

*\*cmdq* pointer to any element on command queue

*\*cmd* pointer to the command to be appended

##### Returns

number of appended elements (one), SWD\_ERROR\_CODE on failure

**5.1.2.14 int swd\_cmd\_queue\_append\_jtag2swd ( swd\_ctx\_t \* swdctx )**

Append command queue with JTAG-TO-SWD DAP-switch sequence.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

number of elements appended, or SWD\_ERROR\_CODE on failure.

**5.1.2.15 int swd\_cmd\_queue\_append\_miso\_ack ( swd\_ctx\_t \* swdctx, char \* ack )**

Append queue with ACK read.

**Parameters**

*\*swdctx* swd context pointer.

*\*ack* packet value pointer.

**Returns**

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.1.2.16 int swd\_cmd\_queue\_append\_miso\_data ( swd\_ctx\_t \* swdctx, int \* data )**

Append command queue with data read.

**Parameters**

*\*swdctx* swd context pointer.

*\*data* data pointer.

**Returns**

of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.1.2.17 int swd\_cmd\_queue\_append\_miso\_data\_p ( swd\_ctx\_t \* swdctx, int \* data, char \* parity )**

Append command queue with data and parity read.

**Parameters**

*\*swdctx* swd context pointer.

*\*data* data value pointer.

*\*parity* parity value pointer.

**Returns**

number of elements appended (2), or SWD\_ERROR\_CODE on failure.



**5.1.2.18** `int swd_cmd_queue_append_miso_n_data_p ( swd_ctx_t * swdctx, int ** data, char ** parity, int count )`

Append command queue with series of data and parity read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*\*data* data value array pointer.  
*\*\*parity* parity value array pointer.  
*count* number of (data+parity) elements to read.

**Returns**

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

**5.1.2.19** `int swd_cmd_queue_append_miso_nbit ( swd_ctx_t * swdctx, char ** data, int count )`

Append command queue with bus binary read bit-by-bit operation.

This function will append command to the queue for each bit, and store one bit into single char array element, so read is not constrained to 8 bits. On error memory is released and appropriate error code is returned. Important: Memory pointed by \*data must be allocated prior call!

**Parameters**

*\*swdctx* swd context pointer.  
*\*\*data* allocated data array to write result into.  
*count* number of bits to read (also the \*\*data size).

**Returns**

number of elements processed, or SWD\_ERROR\_CODE on failure.

**5.1.2.20** `int swd_cmd_queue_append_miso_parity ( swd_ctx_t * swdctx, char * parity )`

Append command queue with parity bit read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*parity* parity value pointer.

**Returns**

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.21 int swd\_cmd\_queue\_append\_miso\_trn ( swd\_ctx\_t \* swdctx )

Append command queue with Turnaround activating MISO mode.

##### Parameters

*\*swdctx* swd context pointer.

##### Returns

return number of elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.22 int swd\_cmd\_queue\_append\_mosi\_control ( swd\_ctx\_t \* swdctx, char \* ctlmsg, int len )

Append command queue with len-octet size control sequence.

This control sequence can be used for instance to send payload of packets switching DAP between JTAG and SWD mode.

##### Parameters

*\*swdctx* swd context pointer.

*\*ctlmsg* control message array pointer.

*len* number of elements to send from \*ctlmsg.

##### Returns

number of elements appended (len), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.23 int swd\_cmd\_queue\_append\_mosi\_data ( swd\_ctx\_t \* swdctx, int \* data )

Append command queue with data and parity write.

##### Parameters

*\*swdctx* swd context pointer.

*\*data* data value pointer.

##### Returns

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.24 int swd\_cmd\_queue\_append\_mosi\_data\_ap ( swd\_ctx\_t \* swdctx, int \* data )

Append command queue with data and automatic parity write.

##### Parameters

*\*swdctx* swd context pointer.

*\*data* data value pointer.

##### Returns

number of elements appended (2), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.25 `int swd_cmd_queue_append_mosi_data_p ( swd_ctx_t * swdctx, int * data, char * parity )`

Append command queue with data and provided parity write.

##### Parameters

*\*swdctx* swd context pointer.

*\*data* data value pointer.

*\*parity* parity value pointer.

##### Returns

number of elements appended (2), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.26 `int swd_cmd_queue_append_mosi_nbit ( swd_ctx_t * swdctx, char * data, int count )`

Append command queue with bus binary write bit-by-bit operation.

This function will append command to the queue for each bit and store one bit into single char array element, so read is not constrained to 8 bits. On error memory is released and appropriate error code is returned. Important: Memory pointed by *\*data* must be allocated prior call!

##### Parameters

*\*swdctx* swd context pointer.

*\*\*data* allocated data array to write result into.

*count* number of bits to read (also the *\*\*data* size).

##### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

#### 5.1.2.27 `int swd_cmd_queue_append_mosi_parity ( swd_ctx_t * swdctx, char * parity )`

Append command queue with parity bit write.

##### Parameters

*\*swdctx* swd context pointer.

*\*parity* parity value pointer.

##### Returns

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.1.2.28 `int swd_cmd_queue_append_mosi_request ( swd_ctx_t * swdctx, char * request )`

Appends command queue with SWD Request packet header.

Note that contents is not validated, so bad request can be sent as well.

**Parameters**

*\*swdctx* swd context pointer.  
*\*request* pointer to the 8-bit request payload.

**Returns**

return number elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.1.2.29 int swd\_cmd\_queue\_append\_mosi\_trn ( swd\_ctx\_t \* swdctx )**

Append command queue with Turnaround activating MOSI mode.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

return number elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.1.2.30 int swd\_cmd\_queue\_append\_swd2jtag ( swd\_ctx\_t \* swdctx )**

Append command queue with SWD-TO-JTAG DAP-switch sequence.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

number of elements appended, or SWD\_ERROR\_CODE on failure.

**5.1.2.31 int swd\_cmd\_queue\_append\_swdpreset ( swd\_ctx\_t \* swdctx )**

Append command queue with SW-DP-RESET sequence.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

number of elements appended, or SWD\_ERROR\_CODE on failure.

**5.1.2.32 swd\_cmd\_t\* swd\_cmd\_queue\_find\_root ( swd\_cmd\_t \* cmdq )**

Find queue root (first element).

**Parameters**

*\*cmdq* pointer to any queue element

**Returns**

swd\_cmd\_t\* pointer to the first element (root), NULL on failure

**5.1.2.33** `swd_cmd_t* swd_cmd_queue_find_tail ( swd_cmd_t * cmdq )`

Find queue tail (last element).

**Parameters**

*\*cmdq* pointer to any queue element

**Returns**

swd\_cmd\_t\* pointer to the last element (tail), NULL on failure

**5.1.2.34** `int swd_cmd_queue_flush ( swd_ctx_t * swdctx, swd_operation_t operation )`

Flush command queue contents into interface driver.

Operation is specified by SWD\_OPERATION and can be used to select how to flush the queue, ie. head-only, tail-only, one, all, etc.

**Parameters**

*\*swdctx* swd context pointer.

*operation* tells how to flush the queue.

**Returns**

number of commands transmitted, or SWD\_ERROR\_CODE on failure.

**5.1.2.35** `int swd_cmd_queue_free ( swd_cmd_t * cmdq )`

Free queue pointed by \*cmdq element.

**Parameters**

*\*cmdq* pointer to any element on command queue

**Returns**

number of elements destroyed, SWD\_ERROR\_CODE on failure

**5.1.2.36** `int swd_cmd_queue_free_head ( swd_cmd_t * cmdq )`

Free queue head up to \*cmdq element.

**Parameters**

*\*cmdq* pointer to the element that becomes new queue root.

**Returns**

number of elements destroyed, or SWD\_ERROR\_CODE on failure.

**5.1.2.37 int swd\_cmd\_queue\_free\_tail ( swd\_cmd\_t \* *cmdq* )**

Free queue tail starting after \*cmdq element.

**Parameters**

\**cmdq* pointer to the last element on the new queue.

**Returns**

number of elements destroyed, or SWD\_ERROR\_CODE on failure.

**5.1.2.38 int swd\_cmd\_queue\_init ( swd\_cmd\_t \* *cmdq* )**

Initialize new queue element in memory that becomes a queue root.

**Parameters**

\**cmdq* pointer to the command queue element of type [swd\\_cmd\\_t](#)

**Returns**

SWD\_OK on success, SWD\_ERROR\_CODE code on failure

**5.1.2.39 int swd\_deinit ( swd\_ctx\_t \* *swdctx* )**

De-initialize selected swd context and its command queue.

**Parameters**

\**swdctx* swd context pointer.

**Returns**

number of elements freed, or SWD\_ERROR\_CODE on failure.

**5.1.2.40 int swd\_deinit\_cmdq ( swd\_ctx\_t \* *swdctx* )**

De-initialize command queue and free its memory on selected swd context.

**Parameters**

\**swdctx* swd context pointer.

**Returns**

number of commands freed, or SWD\_ERROR\_CODE on failure.

#### 5.1.2.41 int swd\_deinit\_ctx ( swd\_ctx\_t \* swdctx )

De-initialize selected swd context and free its memory.

Note: This function will not free command queue for selected context!

##### Parameters

*\*swdctx* swd context pointer.

##### Returns

SWD\_OK on success, SWD\_ERROR\_CODE on failure.

#### 5.1.2.42 int swd\_idcode ( swd\_ctx\_t \* swdctx, swd\_operation\_t operation, int \* idcode, char \* ack, char \* parity )

Read target's IDCODE register value.

##### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform (queue or execute).

*\*idcode* resulting register value pointer.

*\*ack* resulting acknowledge response value pointer.

*\*parity* resulting data parity value pointer.

##### Returns

number of elements processed on the queue, or SWD\_ERROR\_CODE on failure.

#### 5.1.2.43 swd\_ctx\_t\* swd\_init ( void )

LibSWD initialization routine.

It should be called prior any operation made with libswd. It initializes command queue and basic parameters for context that is returned as pointer.

##### Returns

pointer to the initialized swd context.

#### 5.1.2.44 int swd\_miso\_ack ( swd\_ctx\_t \* swdctx, swd\_operation\_t operation, char \* ack )

Perform ACK read into \*ack and verify received data.

##### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform with generated request.

*\*ack* pointer to the result location.

##### Returns

number of commands processed, or SWD\_ERROR\_CODE on failure.

**5.1.2.45** `int swd_miso_data_p ( swd_ctx_t * swdctx, swd_operation_t operation, int * data, char * parity )`

Perform (MISO) data read.

#### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

*\*parity* payload parity value pointer.

#### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

**5.1.2.46** `int swd_mosi_data_ap ( swd_ctx_t * swdctx, swd_operation_t operation, int * data )`

Perform (MOSI) data write with automatic parity calculation.

#### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

#### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

**5.1.2.47** `int swd_mosi_data_p ( swd_ctx_t * swdctx, swd_operation_t operation, int * data, char * parity )`

Perform (MOSI) data write with provided parity value.

#### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

*\*parity* payload parity value pointer.

#### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.



**5.1.2.48** `int swd_mosi_request ( swd_ctx_t * swdctx, swd_operation_t operation, char * APnDP, char * RnW, char * addr )`

Perform Request.

#### Parameters

- \*swdctx* swd context pointer.
- operation* type of action to perform with generated request.
- \*APnDP* AccessPort (high) or DebugPort (low) access value pointer.
- \*RnW* Read (high) or Write (low) access value pointer.
- \*addr* target register address value pointer.

#### Returns

number of commands processed, or SWD\_ERROR\_CODE on failure.

## 5.2 libswd.h File Reference

### Classes

- struct `swd_cmd_t`  
*SWD Command Element Structure.*
- struct `swd_context_config_t`  
*Context configuration structure.*
- struct `swd_swdp_t`  
*Most actual Serial Wire Debug Port Registers.*
- struct `swd_ahbap_t`  
*Most actual Advanced High Bandwidth Access Peripheral Bus Registers.*
- struct `swd_driver_t`  
*Interface Driver structure.*
- struct `swd_ctx_t`  
*SWD Context Structure definition.*

### Defines

- #define `SWD_REQUEST_START_BITNUM` 7  
*SWD Packets Bit Fields and Values.*
- #define `SWD_REQUEST_APnDP_BITNUM` 6  
*Access Port (high) or Debug Port (low) access.*
- #define `SWD_REQUEST_RnW_BITNUM` 5

*Read (high) or Write (low) access.*

- #define SWD\_REQUEST\_ADDR\_BITNUM 4  
*LSB of the address field in request header.*
- #define SWD\_REQUEST\_A2\_BITNUM 4  
*Target Register Address bit 2.*
- #define SWD\_REQUEST\_A3\_BITNUM 3  
*Target Register Address bit 3.*
- #define SWD\_REQUEST\_PARITY\_BITNUM 2  
*Odd Parity calculated from APnDP, RnW, A[2:3].*
- #define SWD\_REQUEST\_STOP\_BITNUM 1  
*Packet Stop bit, always 0.*
- #define SWD\_REQUEST\_PARK\_BITNUM 0  
*Park wire and switch between receive/transmit.*
- #define SWD\_REQUEST\_START\_VAL 1  
*Start Bit Value is always 1.*
- #define SWD\_REQUEST\_STOP\_VAL 0  
*Stop Bit Value is always 0.*
- #define SWD\_REQUEST\_PARK\_VAL 1  
*Park bus and put outputs into Hi-Z state.*
- #define SWD\_REQUEST\_BITLEN 8  
*Number of bits in request packet header.*
- #define SWD\_ADDR\_MINVAL 0  
*Address field minimal value.*
- #define SWD\_ADDR\_MAXVAL 3  
*Address field maximal value.*
- #define SWD\_ACK\_BITLEN 3  
*Number of bits in Acknowledge packet.*
- #define SWD\_ACK\_OK\_VAL 4  
*OK code value.*
- #define SWD\_ACK\_WAIT\_VAL 2  
*WAIT code value.*
- #define SWD\_ACK\_FAULT\_VAL 1  
*FAULT code value.*

- #define [SWD\\_DP\\_ADDR\\_IDCODE](#) 0  
*IDCODE register address (RO).*
- #define [SWD\\_DP\\_ADDR\\_ABORT](#) 0  
*ABORT register address (WO).*
- #define [SWD\\_DP\\_ADDR\\_CTRLSTAT](#) 1  
*CTRLSTAT register address (R/W, CTRLSEL=b0).*
- #define [SWD\\_DP\\_ADDR\\_WCR](#) 1  
*WCR register address (R/W, CTRLSEL=b1).*
- #define [SWD\\_DP\\_ADDR\\_RESEND](#) 2  
*RESEND register address (RO).*
- #define [SWD\\_DP\\_ADDR\\_SELECT](#) 2  
*SELECT register address (WO).*
- #define [SWD\\_DP\\_ADDR\\_RDBUF](#) 3  
*RDBUF register address (RO).*
- #define [SWD\\_ABORT\\_BITNUM\\_DAPABORT](#) 0  
*SW-DP ABORT Register map.*
- #define [SWD\\_ABORT\\_BITNUM\\_DSTKMPCLR](#) 1  
*DSTKMPCLR bit number.*
- #define [SWD\\_ABORT\\_BITNUM\\_DSTKERRCLR](#) 2  
*DSTKERRCLR bit number.*
- #define [SWD\\_ABORT\\_BITNUM\\_DWDERRCLR](#) 3  
*DWDERRCLR bit number.*
- #define [SWD\\_ABORT\\_BITNUM\\_DORUNERRCLR](#) 4  
*DORUNERRCLR bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_ORUNDETECT](#) 0  
*SW-DP CTRL/STAT Register map.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OSTICKYORUN](#) 1  
*OSTICKYORUN bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OTRNMODE](#) 2  
*OTRNMODE bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OSTICKYCMP](#) 4  
*OSTICKYCMP bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OSTICKYERR](#) 5  
*OSTICKYERR bit number.*

- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OREADOK](#) 6  
*OREADOK bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OWDATAERR](#) 7  
*OWDATAERR bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OMASKLANE](#) 8  
*OMASKLANE bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OTRNCNT](#) 12  
*OTRNCNT bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCDBGRSTREQ](#) 26  
*OCDBGRSTREQ bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCDBGRSTACK](#) 27  
*OCDBGRSTACK bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCDBGWRUPREQ](#) 28  
*OCDBGWRUPREQ bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCDBGWRUPACK](#) 29  
*OCDBGWRUPACK bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCSYSPWRUPREQ](#) 30  
*OCSYSPWRUPREQ bit number.*
- #define [SWD\\_CTRLSTAT\\_BITNUM\\_OCSYSPWRUPACK](#) 31  
*OCSYSPWRUPACK bit number.*
- #define [SWD\\_MASKLANE\\_0](#) 0b0001  
*SW-DP CTRLSTAT MASKLANE available values.*
- #define [SWD\\_MASKLANE\\_1](#) 0b0010  
*Compare byte lane 1 (0x---FF--).*
- #define [SWD\\_MASKLANE\\_2](#) 0b0100  
*Compare byte lane 2 (0x--FF----).*
- #define [SWD\\_MASKLANE\\_3](#) 0b1000  
*Compare byte lane 3 (0xFF-----).*
- #define [SWD\\_SELECT\\_BITNUM\\_CTRLSEL](#) 0  
*SW-DP SELECT Register map.*
- #define [SWD\\_SELECT\\_BITNUM\\_APBANKSEL](#) 4  
*APBANKSEL bit number.*
- #define [SWD\\_SELECT\\_BITNUM\\_APSEL](#) 24

*APSEL bit number.*

- #define SWD\_WCR\_BITNUM\_PRESCALER 0

*SW-DP WCR Register map.*

- #define SWD\_WCR\_BITNUM\_WIREMODE 6
- #define SWD\_WCR\_BITNUM\_TURNROUND 8
- #define SWD\_TURNROUND\_1 0

*SW-DP WCR TURNROUND available values.*

- #define SWD\_TURNROUND\_2 1
- #define SWD\_TURNROUND\_3 2
- #define SWD\_TURNROUND\_4 3
- #define SWD\_TURNROUND\_MIN SWD\_TURNROUND\_1
- #define SWD\_TURNROUND\_MAX SWD\_TURNROUND\_4
- #define SWD\_TURNROUND\_DEFAULT SWD\_TURNROUND\_1
- #define AHB\_AP\_CONTROLSTATUS 0x00

*AHB-AP Registers Map.*

- #define AHB\_AP\_TAR 0x04

*R/W, 32bit, reset value: 0x00000000.*

- #define AHB\_AP\_DRW 0x0C

*R/W, 32bit.*

- #define AHB\_AP\_BD0 0x10

*R/W, 32bit.*

- #define AHB\_AP\_BD1 0x14

*R/W, 32bit.*

- #define AHB\_AP\_BD2 0x18

*R/W, 32bit.*

- #define AHB\_AP\_BD3 0x1C

*R/W, 32bit.*

- #define AHB\_AP\_DROMT 0xF8

*RO, 32bit, reset value: 0xE00FF000.*

- #define AHB\_AP\_IDR 0xFC

*RO, 32bit, reset value: 0x24770001.*

- #define SWD\_DATA\_MAXBITCOUNT 32

*SWD queue and payload data definitions.*

- #define SWD\_DATA\_BYTESIZE 8

*How many bits are there in a byte.*

- #define SWD\_DATA\_BITLEN 32

*How many bits are there in data payload.*

- #define [SWD\\_CMDQLEN\\_DEFAULT](#) 1024;  
*How long is the command queue by default.*

## Typedefs

- typedef enum [SWD\\_ERROR\\_CODE](#) [swd\\_error\\_code\\_t](#)  
*Status and Error Codes definitions.*
- typedef enum [SWD\\_LOGLEVEL](#) [swd\\_loglevel\\_t](#)  
*Logging Level Codes definition.*
- typedef enum [SWD\\_CMDTYPE](#) [swd\\_cmdtype\\_t](#)  
*SWD Command Codes definitions.*
- typedef enum [SWD\\_SHIFTDIR](#) [swd\\_shiftdir\\_t](#)  
*What is the shift direction LSB-first or MSB-first.*
- typedef enum [SWD\\_OPERATION](#) [swd\\_operation\\_t](#)  
*Command Queue operations codes.*
- typedef struct [swd\\_cmd\\_t](#) [swd\\_cmd\\_t](#)  
*SWD Command Element Structure.*

## Enumerations

- enum [SWD\\_ERROR\\_CODE](#) {  
[SWD\\_OK](#) = 0, [SWD\\_ERROR\\_GENERAL](#) = -1, [SWD\\_ERROR\\_NULLPOINTER](#) = -2, [SWD\\_ERROR\\_NULLQUEUE](#) = -3,  
[SWD\\_ERROR\\_NULLTRN](#) = -4, [SWD\\_ERROR\\_PARAM](#) = -5, [SWD\\_ERROR\\_OUTOFMEM](#) = -6,  
[SWD\\_ERROR\\_RESULT](#) = -7,  
[SWD\\_ERROR\\_RANGE](#) = -8, [SWD\\_ERROR\\_DEFINITION](#) = -9, [SWD\\_ERROR\\_NULLCONTEXT](#) = -10, [SWD\\_ERROR\\_QUEUE](#) = -11,  
[SWD\\_ERROR\\_ADDR](#) = -12, [SWD\\_ERROR\\_APnDP](#) = -13, [SWD\\_ERROR\\_RnW](#) = -14, [SWD\\_ERROR\\_PARITY](#) = -15,  
[SWD\\_ERROR\\_ACK](#) = -16, [SWD\\_ERROR\\_ACKUNKNOWN](#) = -19, [SWD\\_ERROR\\_ACKNOTDONE](#) = -20, [SWD\\_ERROR\\_ACKMISSING](#) = -21,  
[SWD\\_ERROR\\_ACKMISMATCH](#) = -22, [SWD\\_ERROR\\_ACKORDER](#) = -23, [SWD\\_ERROR\\_BADOPCODE](#) = -24, [SWD\\_ERROR\\_NODATACMD](#) = -25,  
[SWD\\_ERROR\\_DATAADDR](#) = -26, [SWD\\_ERROR\\_NOPARITYCMD](#) = -27, [SWD\\_ERROR\\_PARITYADDR](#) = -28, [SWD\\_ERROR\\_NOTDONE](#) = -29,  
[SWD\\_ERROR\\_QUEUEEROOT](#) = -30, [SWD\\_ERROR\\_BADCMDTYPE](#) = -31, [SWD\\_ERROR\\_BADCMDDATA](#) = -32, [SWD\\_ERROR\\_TURNAROUND](#) = -33,  
[SWD\\_ERROR\\_DRIVER](#) = -34, [SWD\\_ERROR\\_ACK\\_WAIT](#) = -35, [SWD\\_ERROR\\_ACK\\_FAULT](#) = -36, [SWD\\_ERROR\\_QUEUEENOTFREE](#) = -37,  
[SWD\\_ERROR\\_TRANSPORT](#) = -38 }

*Status and Error Codes definitions.*

- enum `SWD_LOGLEVEL` {  
`SWD_LOGLEVEL_SILENT` = 0, `SWD_LOGLEVEL_INFO` = 1, `SWD_LOGLEVEL_WARNING`  
= 2, `SWD_LOGLEVEL_ERROR` = 3,  
`SWD_LOGLEVEL_DEBUG` = 4 }

*Logging Level Codes definition.*

- enum `SWD_CMDTYPE` {  
`SWD_CMDTYPE_MOSI_DATA` = -7, `SWD_CMDTYPE_MOSI_REQUEST` = -6, `SWD_-`  
`CMDTYPE_MOSI_TRN` = -5, `SWD_CMDTYPE_MOSI_PARITY` = -4,  
`SWD_CMDTYPE_MOSI_BITBANG` = -3, `SWD_CMDTYPE_MOSI_CONTROL` = -2, `SWD_-`  
`CMDTYPE_MOSI` = -1, `SWD_CMDTYPE_UNDEFINED` = 0,  
`SWD_CMDTYPE_MISO` = 1, `SWD_CMDTYPE_MISO_ACK` = 2, `SWD_CMDTYPE_MISO_-`  
`BITBANG` = 3, `SWD_CMDTYPE_MISO_PARITY` = 4,  
`SWD_CMDTYPE_MISO_TRN` = 5, `SWD_CMDTYPE_MISO_DATA` = 6 }

*SWD Command Codes definitions.*

- enum `SWD_SHIFTDIR` { `SWD_DIR_LSBFIRST` = 0, `SWD_DIR_MSBFIRST` = 1 }

*What is the shift direction LSB-first or MSB-first.*

- enum `SWD_OPERATION` {  
`SWD_OPERATION_FIRST` = 1, `SWD_OPERATION_QUEUE_APPEND` = 1, `SWD_-`  
`OPERATION_TRANSMIT_HEAD` = 2, `SWD_OPERATION_TRANSMIT_TAIL` = 3,  
`SWD_OPERATION_TRANSMIT_ALL` = 4, `SWD_OPERATION_TRANSMIT_ONE` = 5, `SWD_-`  
`OPERATION_TRANSMIT_LAST` = 6, `SWD_OPERATION_QUEUE` = 7,  
`SWD_OPERATION_EXECUTE` = 8, `SWD_OPERATION_LAST` = 8 }

*Command Queue operations codes.*

- enum `swd_bool_t` { `SWD_FALSE` = 0, `SWD_TRUE` = 1 }

*Boolean values definition.*

## Functions

- int `swd_bin8_parity_even` (char \*data, char \*parity)  
*Data parity calculator, calculates even parity on char type.*
- int `swd_bin32_parity_even` (int \*data, char \*parity)  
*Data parity calculator, calculates even parity on integer type.*
- int `swd_bin8_print` (char \*data)  
*Prints binary data of a char value on the screen.*
- int `swd_bin32_print` (int \*data)  
*Prints binary data of an integer value on the screen.*
- char \* `swd_bin8_string` (char \*data)

*Generates string containing binary data of a char value.*

- `char * swd_bin32_string (int *data)`  
*Generates string containing binary data of an integer value.*
- `int swd_bin8_bitswap (unsigned char *buffer, int bitcount)`  
*Bit swap helper function that reverse bit order in char \*buffer.*
- `int swd_bin32_bitswap (unsigned int *buffer, int bitcount)`  
*Bit swap helper function that reverse bit order in int \*buffer.*
- `int swd_cmd_queue_init (swd_cmd_t *cmdq)`  
*Initialize new queue element in memory that becomes a queue root.*
- `swd_cmd_t * swd_cmd_queue_find_root (swd_cmd_t *cmdq)`  
*Find queue root (first element).*
- `swd_cmd_t * swd_cmd_queue_find_tail (swd_cmd_t *cmdq)`  
*Find queue tail (last element).*
- `int swd_cmd_queue_append (swd_cmd_t *cmdq, swd_cmd_t *cmd)`  
*Append element pointed by \*cmd at the end of the quque pointed by \*cmdq.*
- `int swd_cmd_queue_free (swd_cmd_t *cmdq)`  
*Free queue pointed by \*cmdq element.*
- `int swd_cmd_queue_free_head (swd_cmd_t *cmdq)`  
*Free queue head up to \*cmdq element.*
- `int swd_cmd_queue_free_tail (swd_cmd_t *cmdq)`  
*Free queue tail starting after \*cmdq element.*
- `int swd_cmd_queue_append_mosi_request (swd_ctx_t *swdctx, char *request)`  
*Appends command queue with SWD Request packet header.*
- `int swd_cmd_queue_append_mosi_trn (swd_ctx_t *swdctx)`  
*Append command queue with Turnaround activating MOSI mode.*
- `int swd_cmd_queue_append_miso_trn (swd_ctx_t *swdctx)`  
*Append command queue with Turnaround activating MISO mode.*
- `int swd_cmd_queue_append_miso_nbit (swd_ctx_t *swdctx, char **data, int count)`  
*Append command queue with bus binary read bit-by-bit operation.*
- `int swd_cmd_queue_append_mosi_nbit (swd_ctx_t *swdctx, char *data, int count)`  
*Append command queue with bus binary write bit-by-bit operation.*
- `int swd_cmd_queue_append_mosi_parity (swd_ctx_t *swdctx, char *parity)`  
*Append command queue with parity bit write.*



- int [swd\\_cmd\\_queue\\_append\\_miso\\_parity](#) (swd\_ctx\_t \*swdctx, char \*parity)  
*Append command queue with parity bit read.*
- int [swd\\_cmd\\_queue\\_append\\_miso\\_data](#) (swd\_ctx\_t \*swdctx, int \*data)  
*Append command queue with data read.*
- int [swd\\_cmd\\_queue\\_append\\_miso\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*data, char \*parity)  
*Append command queue with data and parity read.*
- int [swd\\_cmd\\_queue\\_append\\_miso\\_n\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*\*data, char \*\*parity, int count)  
*Append command queue with series of data and parity read.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data](#) (swd\_ctx\_t \*swdctx, int \*data)  
*Append command queue with data and parity write.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data\\_ap](#) (swd\_ctx\_t \*swdctx, int \*data)  
*Append command queue with data and automatic parity write.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*data, char \*parity)  
*Append command queue with data and provided parity write.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_n\\_data\\_ap](#) (swd\_ctx\_t \*swdctx, int \*\*data, int count)  
*Append command queue with series of data and automatic parity writes.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_n\\_data\\_p](#) (swd\_ctx\_t \*swdctx, int \*\*data, char \*\*parity, int count)  
*Append command queue with series of data and provided parity writes.*
- int [swd\\_cmd\\_queue\\_append\\_miso\\_ack](#) (swd\_ctx\_t \*swdctx, char \*ack)  
*Append queue with ACK read.*
- int [swd\\_cmd\\_queue\\_append\\_mosi\\_control](#) (swd\_ctx\_t \*swdctx, char \*ctlmsg, int len)  
*Append command queue with len-octet size control seruence.*
- int [swd\\_cmd\\_queue\\_append\\_swdpreset](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with SW-DP-RESET sequence.*
- int [swd\\_cmd\\_queue\\_append\\_jtag2swd](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with JTAG-TO-SWD DAP-switch sequence.*
- int [swd\\_cmd\\_queue\\_append\\_swid2jtag](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with SWD-TO-JTAG DAP-switch sequence.*
- int [swd\\_bus\\_setdir\\_mosi](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with TRN WRITE/MOSI.*
- int [swd\\_bus\\_setdir\\_miso](#) (swd\_ctx\_t \*swdctx)  
*Append command queue with TRN READ/MISO.*

- `int swd_bit8_gen_request (swd_ctx_t *swdctx, char *APnDP, char *RnW, char *addr, char *request)`
- `int swd_transmit (swd_ctx_t *swdctx, swd_cmd_t *cmd)`
- `int swd_cmd_queue_flush (swd_ctx_t *swdctx, swd_operation_t operation)`  
*Flush command queue contents into interface driver.*
- `int swd_mosi_request (swd_ctx_t *swdctx, swd_operation_t operation, char *APnDP, char *RnW, char *addr)`  
*Perform Request.*
- `int swd_miso_ack (swd_ctx_t *swdctx, swd_operation_t operation, char *ack)`  
*Perform ACK read into \*ack and verify received data.*
- `int swd_mosi_data_p (swd_ctx_t *swdctx, swd_operation_t operation, int *data, char *parity)`  
*Perform (MOSI) data write with provided parity value.*
- `int swd_mosi_data_ap (swd_ctx_t *swdctx, swd_operation_t operation, int *data)`  
*Perform (MOSI) data write with automatic parity calculation.*
- `int swd_miso_data_p (swd_ctx_t *swdctx, swd_operation_t operation, int *data, char *parity)`  
*Perform (MISO) data read.*
- `int swd_idcode (swd_ctx_t *swdctx, swd_operation_t operation, int *idcode, char *ack, char *parity)`  
*Read target's IDCODE register value.*
- `int swd_log (swd_loglevel_t loglevel, char *msg)`
- `char * swd_error_string (swd_error_code_t error)`
- `swd_ctx_t * swd_init (void)`  
*LibSWD initialization routine.*
- `int swd_deinit_ctx (swd_ctx_t *swdctx)`  
*De-initialize selected swd context and free its memory.*
- `int swd_deinit_cmdq (swd_ctx_t *swdctx)`  
*De-initialize command queue and free its memory on selected swd context.*
- `int swd_deinit (swd_ctx_t *swdctx)`  
*De-initialize selected swd context and its command queue.*
- `int swd_drv_mosi_8 (swd_ctx_t *swdctx, char *data, int bits, int direction)`
- `int swd_drv_mosi_32 (swd_ctx_t *swdctx, int *data, int bits, int direction)`
- `int swd_drv_miso_8 (swd_ctx_t *swdctx, char *data, int bits, int direction)`
- `int swd_drv_miso_32 (swd_ctx_t *swdctx, int *data, int bits, int direction)`
- `int swd_drv_mosi_trn (swd_ctx_t *swdctx, int clks)`
- `int swd_drv_miso_trn (swd_ctx_t *swdctx, int clks)`

## 5.2.1 Detailed Description

## 5.2.2 Define Documentation

### 5.2.2.1 #define AHB\_AP\_BD0 0x10

R/W, 32bit.

R/W, 32bit

### 5.2.2.2 #define AHB\_AP\_BD1 0x14

R/W, 32bit.

R/W, 32bit

### 5.2.2.3 #define AHB\_AP\_BD2 0x18

R/W, 32bit.

R/W, 32bit

### 5.2.2.4 #define AHB\_AP\_BD3 0x1C

R/W, 32bit.

R/W, 32bit

### 5.2.2.5 #define AHB\_AP\_CONTROLSTATUS 0x00

AHB-AP Registers Map.

TODO!!!! R/W, 32bit, reset value: 0x43800042 R/W, 32bit, reset value: 0x43800042

### 5.2.2.6 #define AHB\_AP\_DROMT 0xF8

RO, 32bit, reset value: 0xE00FF000.

RO, 32bit, reset value: 0xE00FF000

### 5.2.2.7 #define AHB\_AP\_DRW 0x0C

R/W, 32bit.

R/W, 32bit

### 5.2.2.8 #define AHB\_AP\_IDR 0xFC

RO, 32bit, reset value: 0x24770001.

RO, 32bit, reset value: 0x24770001

**5.2.2.9 #define AHB\_AP\_TAR 0x04**

R/W, 32bit, reset value: 0x00000000.

R/W, 32bit, reset value: 0x00000000

**5.2.2.10 #define SWD\_ABORT\_BITNUM\_DAPABORT 0**

SW-DP ABORT Register map.

DAPABORT bit number.

**5.2.2.11 #define SWD\_CTRLSTAT\_BITNUM\_ORUNDETECT 0**

SW-DP CTRL/STAT Register map.

ORUNDETECT bit number.

**5.2.2.12 #define SWD\_DATA\_MAXBITCOUNT 32**

SWD queue and payload data definitions.

What is the maximal bit length of the data.

**5.2.2.13 #define SWD\_MASKLANE\_0 0b0001**

SW-DP CTRLSTAT MASKLANE available values.

Compare byte lane 0 (0x-----FF)

**5.2.2.14 #define SWD\_REQUEST\_START\_BITNUM 7**

SWD Packets Bit Fields and Values.

Packet Start bit, always set to 1.

**5.2.2.15 #define SWD\_SELECT\_BITNUM\_CTRLSEL 0**

SW-DP SELECT Register map.

CTRLSEL bit number.

**5.2.2.16 #define SWD\_TURNROUND\_1 0**

SW-DP WCR TURNROUND available values.

TRN takes one CLK cycle. TRN takes one CLK cycle.

**5.2.2.17 #define SWD\_TURNROUND\_2 1**

TRN takes two CLK cycles.

**5.2.2.18 #define SWD\_TURNROUND\_3 2**

TRN takes three CLK cycles.

**5.2.2.19 #define SWD\_TURNROUND\_4 3**

TRN takes four CLK cycles. ????

**5.2.2.20 #define SWD\_TURNROUND\_DEFAULT SWD\_TURNROUND\_1**

Default TRN length is one CLK.

**5.2.2.21 #define SWD\_TURNROUND\_MAX SWD\_TURNROUND\_4**

longest TRN time.

**5.2.2.22 #define SWD\_TURNROUND\_MIN SWD\_TURNROUND\_1**

shortest TRN time.

**5.2.2.23 #define SWD\_WCR\_BITNUM\_PRESCALER 0**

SW-DP WCR Register map.

PRESCALER bit number. PRESCALER bit number.

**5.2.2.24 #define SWD\_WCR\_BITNUM\_TURNROUND 8**

TURNROUND bit number.

**5.2.2.25 #define SWD\_WCR\_BITNUM\_WIREMODE 6**

WIREMODE bit number.

**5.2.3 Typedef Documentation****5.2.3.1 typedef struct swd\_cmd\_t swd\_cmd\_t**

SWD Command Element Structure.

In libswd each operation is split into separate commands (request, trn, ack, data, parity) that can be appended to the command queue and later executed. This organization allows better granularity for tracing bugs and makes possible to compose complete bus/target operations made of simple commands.

**5.2.3.2 typedef enum SWD\_CMDTYPE swd\_cmdtype\_t**

SWD Command Codes definitions.

Available values: MISO>0, MOSI<0, undefined=0. To check command direction (read/write) multiply tested value with one of the MOSI or MISO commands

- result is positive for equal direction and negative if direction differs. Command Type codes definition, use this to see names in debugger.

### 5.2.3.3 typedef enum SWD\_ERROR\_CODE swd\_error\_code\_t

Status and Error Codes definitions.

Error Codes definition, use this to have its name on debugger.

### 5.2.3.4 typedef enum SWD\_LOGLEVEL swd\_loglevel\_t

Logging Level Codes definition.

Logging Level codes definition, use this to have its name on debugger.

### 5.2.3.5 typedef enum SWD\_OPERATION swd\_operation\_t

Command Queue operations codes.

### 5.2.3.6 typedef enum SWD\_SHIFTDIR swd\_shiftdir\_t

What is the shift direction LSB-first or MSB-first.

## 5.2.4 Enumeration Type Documentation

### 5.2.4.1 enum swd\_bool\_t

Boolean values definition.

**Enumerator:**

*SWD\_FALSE* False is 0.

*SWD\_TRUE* True is 1.

### 5.2.4.2 enum SWD\_CMDTYPE

SWD Command Codes definitions.

Available values: MISO>0, MOSI<0, undefined=0. To check command direction (read/write) multiply tested value with one of the MOSI or MISO commands

- result is positive for equal direction and negative if direction differs. Command Type codes definition, use this to see names in debugger.

**Enumerator:**

*SWD\_CMDTYPE\_MOSI\_DATA* Contains MOSI data (from host).

***SWD\_CMDTYPE\_MOSI\_REQUEST*** Contains MOSI request packet.

***SWD\_CMDTYPE\_MOSI\_TRN*** Bus will switch into MOSI mode.

***SWD\_CMDTYPE\_MOSI\_PARITY*** Contains MOSI data parity.

***SWD\_CMDTYPE\_MOSI\_BITBANG*** Allows MOSI operation bit-by-bit.

***SWD\_CMDTYPE\_MOSI\_CONTROL*** MOSI control sequence (ie. sw-dp reset).

***SWD\_CMDTYPE\_MOSI*** Master Output Slave Input direction.

***SWD\_CMDTYPE\_UNDEFINED*** undefined command, not transmitted.

***SWD\_CMDTYPE\_MISO*** Master Input Slave Output direction.

***SWD\_CMDTYPE\_MISO\_ACK*** Contains ACK data from target.

***SWD\_CMDTYPE\_MISO\_BITBANG*** Allows MISO operation bit-by-bit.

***SWD\_CMDTYPE\_MISO\_PARITY*** Contains MISO data parity.

***SWD\_CMDTYPE\_MISO\_TRN*** Bus will switch into MISO mode.

***SWD\_CMDTYPE\_MISO\_DATA*** Contains MISO data (from target).

#### 5.2.4.3 enum SWD\_ERROR\_CODE

Status and Error Codes definitions.

Error Codes definition, use this to have its name on debugger.

##### Enumerator:

***SWD\_OK*** No error.

***SWD\_ERROR\_GENERAL*** General error.

***SWD\_ERROR\_NULLPOINTER*** Null pointer.

***SWD\_ERROR\_NULLQUEUE*** Null queue pointer.

***SWD\_ERROR\_NULLTRN*** Null TurnaRouNd pointer.

***SWD\_ERROR\_PARAM*** Bad parameter.

***SWD\_ERROR\_OUTOFMEM*** Out of memory.

***SWD\_ERROR\_RESULT*** Bad result.

***SWD\_ERROR\_RANGE*** Out of range.

***SWD\_ERROR\_DEFINITION*** Definition (internal) error.

***SWD\_ERROR\_NULLCONTEXT*** Null context pointer.

***SWD\_ERROR\_QUEUE*** Queue error.

***SWD\_ERROR\_ADDR*** Addressing error.

***SWD\_ERROR\_APnDP*** Bad APnDP value.

***SWD\_ERROR\_RnW*** Bad RnW value.

***SWD\_ERROR\_PARITY*** Parity error.

***SWD\_ERROR\_ACK*** Acknowledge error.

***SWD\_ERROR\_ACKUNKNOWN*** Unknown ACK value.

***SWD\_ERROR\_ACKNOTDONE*** ACK not yet executed on target.

***SWD\_ERROR\_ACKMISSING*** ACK command not found on the queue.

***SWD\_ERROR\_ACKMISMATCH*** Bad ACK result address.

*SWD\_ERROR\_ACKORDER* ACK not in order REQ->TRN->ACK.  
*SWD\_ERROR\_BADOPCODE* Unsupported operation requested.  
*SWD\_ERROR\_NODATACMD* Command not found on the queue.  
*SWD\_ERROR\_DATAADDR* Bad DATA result address.  
*SWD\_ERROR\_NOPARITYCMD* Parity command missing or misplaced.  
*SWD\_ERROR\_PARITYADDR* Bad PARITY command result address.  
*SWD\_ERROR\_NOTDONE* Could not end selected task.  
*SWD\_ERROR\_QUEUEROOT* Queue root not found or null.  
*SWD\_ERROR\_BADCMDTYPE* Unknown command detected.  
*SWD\_ERROR\_BADCMDDATA* Bad command data.  
*SWD\_ERROR\_TURNAROUND* Error during turnaround switch.  
*SWD\_ERROR\_DRIVER* Driver error.  
*SWD\_ERROR\_ACK\_WAIT* Received ACK WAIT.  
*SWD\_ERROR\_ACK\_FAULT* Received ACK FAULT.  
*SWD\_ERROR\_QUEUENOTFREE* Cannot free resources, queue not empty.  
*SWD\_ERROR\_TRANSPORT* Transport type unknown or undefined.

#### 5.2.4.4 enum SWD\_LOGLEVEL

Logging Level Codes definition.

Logging Level codes definition, use this to have its name on debugger.

**Enumerator:**

*SWD\_LOGLEVEL\_SILENT* Remain silent.  
*SWD\_LOGLEVEL\_INFO* Log only informational messages.  
*SWD\_LOGLEVEL\_WARNING* also log warnings.  
*SWD\_LOGLEVEL\_ERROR* also log errors.  
*SWD\_LOGLEVEL\_DEBUG* Log everything including detailed details.

#### 5.2.4.5 enum SWD\_OPERATION

Command Queue operations codes.

**Enumerator:**

*SWD\_OPERATION\_FIRST* First operation to know its code.  
*SWD\_OPERATION\_QUEUE\_APPEND* Append command to the queue.  
*SWD\_OPERATION\_TRANSMIT\_HEAD* Transmit root..current (head).  
*SWD\_OPERATION\_TRANSMIT\_TAIL* Transmit current..last (tail).  
*SWD\_OPERATION\_TRANSMIT\_ALL* Transmit all commands on the queue.  
*SWD\_OPERATION\_TRANSMIT\_ONE* Transmit only current command.  
*SWD\_OPERATION\_TRANSMIT\_LAST* Transmit last command on the queue.  
*SWD\_OPERATION\_QUEUE* Only queue provided commands.  
*SWD\_OPERATION\_EXECUTE* Execute provided commands.  
*SWD\_OPERATION\_LAST* Last operation to know its code.



#### 5.2.4.6 enum SWD\_SHIFTDIR

What is the shift direction LSB-first or MSB-first.

##### Enumerator:

***SWD\_DIR\_LSBFIRST*** Data is shifted in/out right (LSB-first).

***SWD\_DIR\_MSBFIRST*** Data is shifted in/out left (MSB-first).

### 5.2.5 Function Documentation

#### 5.2.5.1 int swd\_bin32\_bitswap ( unsigned int \* *buffer*, int *bitcount* )

Bit swap helper function that reverse bit order in int \*buffer.

Most Significant Bit becomes Least Significant Bit. It is possible to swap only n-bits from int (32-bit) \*buffer.

##### Parameters

***\*buffer*** unsigned char (32-bit) data pointer.

***bitcount*** how many bits to swap.

##### Returns

swapped bit count (positive) or error code (negative).

#### 5.2.5.2 int swd\_bin32\_parity\_even ( int \* *data*, char \* *parity* )

Data parity calculator, calculates even parity on integer type.

##### Parameters

***\*data*** source data pointer.

***\*parity*** resulting data pointer.

##### Returns

negative value on error, 0 or 1 as parity result.

#### 5.2.5.3 int swd\_bin32\_print ( int \* *data* )

Prints binary data of an integer value on the screen.

##### Parameters

***\*data*** source data pointer.

##### Returns

number of characters printed.

**5.2.5.4 char\* swd\_bin32\_string ( int \* *data* )**

Generates string containing binary data of an integer value.

**Parameters**

*\*data* source data pointer.

**Returns**

pointer to the resulting string.

**5.2.5.5 int swd\_bin8\_bitswap ( unsigned char \* *buffer*, int *bitcount* )**

Bit swap helper function that reverse bit order in char \*buffer.

Most Significant Bit becomes Least Significant Bit. It is possible to swap only n-bits from char (8-bit) \*buffer.

**Parameters**

*\*buffer* unsigned char (8-bit) data pointer.

*bitcount* how many bits to swap.

**Returns**

swapped bit count (positive) or error code (negative).

**5.2.5.6 int swd\_bin8\_parity\_even ( char \* *data*, char \* *parity* )**

Data parity calculator, calculates even parity on char type.

**Parameters**

*\*data* source data pointer.

*\*parity* resulting data pointer.

**Returns**

negative value on error, 0 or 1 as parity result.

**5.2.5.7 int swd\_bin8\_print ( char \* *data* )**

Prints binary data of a char value on the screen.

**Parameters**

*\*data* source data pointer.

**Returns**

number of characters printed.

#### 5.2.5.8 char\* swd\_bin8\_string ( char \* *data* )

Generates string containing binary data of a char value.

##### Parameters

*\*data* source data pointer.

##### Returns

pointer to the resulting string.

#### 5.2.5.9 int swd\_bus\_setdir\_miso ( swd\_ctx\_t \* *swdctx* )

Append command queue with TRN READ/MISO.

##### Parameters

*\*swdctx* swd context pointer.

##### Returns

number of elements appended, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.10 int swd\_bus\_setdir\_mosi ( swd\_ctx\_t \* *swdctx* )

Append command queue with TRN WRITE/MOSI.

##### Parameters

*\*swdctx* swd context pointer.

##### Returns

number of elements appended, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.11 int swd\_cmd\_append\_mosi\_n\_data\_ap ( swd\_ctx\_t \* *swdctx*, int \*\* *data*, int *count* )

Append command queue with series of data and automatic parity writes.

##### Parameters

*\*swdctx* swd context pointer.

*\*\*data* data value array pointer.

*count* number of (data+parity) elements to read.

##### Returns

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

### 5.2.5.12 `int swd_cmd_append_mosi_n_data_p ( swd_ctx_t * swdctx, int ** data, char ** parity, int count )`

Append command queue with series of data and provided parity writes.

#### Parameters

- \*swdctx* swd context pointer.
- \*\*data* data value array pointer.
- \*\*parity* parity value array pointer.
- count* number of (data+parity) elements to read.

#### Returns

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

### 5.2.5.13 `int swd_cmd_queue_append ( swd_cmd_t * cmdq, swd_cmd_t * cmd )`

Append element pointed by \*cmd at the end of the quque pointed by \*cmdq.

#### Parameters

- \*cmdq* pointer to any element on command queue
- \*cmd* pointer to the command to be appended

#### Returns

number of appended elements (one), SWD\_ERROR\_CODE on failure

### 5.2.5.14 `int swd_cmd_queue_append_jtag2swd ( swd_ctx_t * swdctx )`

Append command queue with JTAG-TO-SWD DAP-switch sequence.

#### Parameters

- \*swdctx* swd context pointer.

#### Returns

number of elements appended, or SWD\_ERROR\_CODE on failure.

### 5.2.5.15 `int swd_cmd_queue_append_miso_ack ( swd_ctx_t * swdctx, char * ack )`

Append queue with ACK read.

#### Parameters

- \*swdctx* swd context pointer.
- \*ack* packet value pointer.

#### Returns

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.2.5.16 int swd\_cmd\_queue\_append\_miso\_data ( swd\_ctx\_t \* swdctx, int \* data )**

Append command queue with data read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*data* data pointer.

**Returns**

of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.2.5.17 int swd\_cmd\_queue\_append\_miso\_data\_p ( swd\_ctx\_t \* swdctx, int \* data, char \* parity )**

Append command queue with data and parity read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*data* data value pointer.  
*\*parity* parity value pointer.

**Returns**

number of elements appended (2), or SWD\_ERROR\_CODE on failure.

**5.2.5.18 int swd\_cmd\_queue\_append\_miso\_n\_data\_p ( swd\_ctx\_t \* swdctx, int \*\* data, char \*\* parity, int count )**

Append command queue with series of data and parity read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*\*data* data value array pointer.  
*\*\*parity* parity value array pointer.  
*count* number of (data+parity) elements to read.

**Returns**

number of elements appended (2\*count), or SWD\_ERROR\_CODE on failure.

**5.2.5.19 int swd\_cmd\_queue\_append\_miso\_nbit ( swd\_ctx\_t \* swdctx, char \*\* data, int count )**

Append command queue with bus binary read bit-by-bit operation.

This function will append command to the queue for each bit, and store one bit into single char array element, so read is not constrained to 8 bits. On error memory is released and appropriate error code is returned. Important: Memory pointed by *\*data* must be allocated prior call!

**Parameters**

*\*swdctx* swd context pointer.  
*\*\*data* allocated data array to write result into.  
*count* number of bits to read (also the *\*\*data* size).

**Returns**

number of elements processed, or SWD\_ERROR\_CODE on failure.

**5.2.5.20 int swd\_cmd\_queue\_append\_miso\_parity ( swd\_ctx\_t \* swdctx, char \* parity )**

Append command queue with parity bit read.

**Parameters**

*\*swdctx* swd context pointer.  
*\*parity* parity value pointer.

**Returns**

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.2.5.21 int swd\_cmd\_queue\_append\_miso\_trn ( swd\_ctx\_t \* swdctx )**

Append command queue with Turnaround activating MISO mode.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

return number of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.2.5.22 int swd\_cmd\_queue\_append\_mosi\_control ( swd\_ctx\_t \* swdctx, char \* ctlmsg, int len )**

Append command queue with len-octet size control sequence.

This control sequence can be used for instance to send payload of packets switching DAP between JTAG and SWD mode.

**Parameters**

*\*swdctx* swd context pointer.  
*\*ctlmsg* control message array pointer.  
*len* number of elements to send from *\*ctlmsg*.

**Returns**

number of elements appended (len), or SWD\_ERROR\_CODE on failure.

**5.2.5.23 int swd\_cmd\_queue\_append\_mosi\_data ( swd\_ctx\_t \* *swdctx*, int \* *data* )**

Append command queue with data and parity write.

**Parameters**

*\*swdctx* swd context pointer.

*\*data* data value pointer.

**Returns**

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

**5.2.5.24 int swd\_cmd\_queue\_append\_mosi\_data\_ap ( swd\_ctx\_t \* *swdctx*, int \* *data* )**

Append command queue with data and automatic parity write.

**Parameters**

*\*swdctx* swd context pointer.

*\*data* data value pointer.

**Returns**

number of elements appended (2), or SWD\_ERROR\_CODE on failure.

**5.2.5.25 int swd\_cmd\_queue\_append\_mosi\_data\_p ( swd\_ctx\_t \* *swdctx*, int \* *data*, char \* *parity* )**

Append command queue with data and provided parity write.

**Parameters**

*\*swdctx* swd context pointer.

*\*data* data value pointer.

*\*parity* parity value pointer.

**Returns**

number of elements appended (2), or SWD\_ERROR\_CODE on failure.

**5.2.5.26 int swd\_cmd\_queue\_append\_mosi\_nbit ( swd\_ctx\_t \* *swdctx*, char \* *data*, int *count* )**

Append command queue with bus binary write bit-by-bit operation.

This function will append command to the queue for each bit and store one bit into single char array element, so read is not constrained to 8 bits. On error memory is released and appropriate error code is returned. Important: Memory pointed by *\*data* must be allocated prior call!

**Parameters**

*\*swdctx* swd context pointer.

***\*\*data*** allocated data array to write result into.  
***count*** number of bits to read (also the ***\*\*data*** size).

#### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.27 int swd\_cmd\_queue\_append\_mosi\_parity ( swd\_ctx\_t \* swdctx, char \* parity )

Append command queue with parity bit write.

#### Parameters

***\*swdctx*** swd context pointer.  
***\*parity*** parity value pointer.

#### Returns

number of elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.2.5.28 int swd\_cmd\_queue\_append\_mosi\_request ( swd\_ctx\_t \* swdctx, char \* request )

Appends command queue with SWD Request packet header.

Note that contents is not validated, so bad request can be sent as well.

#### Parameters

***\*swdctx*** swd context pointer.  
***\*request*** pointer to the 8-bit request payload.

#### Returns

return number elements appended (1), or SWD\_ERROR\_CODE on failure.

#### 5.2.5.29 int swd\_cmd\_queue\_append\_mosi\_trn ( swd\_ctx\_t \* swdctx )

Append command queue with Turnaround activating MOSI mode.

#### Parameters

***\*swdctx*** swd context pointer.

#### Returns

return number elements appended (1), or SWD\_ERROR\_CODE on failure.



**5.2.5.30 int swd\_cmd\_queue\_append\_swd2jtag ( swd\_ctx\_t \* swdctx )**

Append command queue with SWD-TO-JTAG DAP-switch sequence.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

number of elements appended, or SWD\_ERROR\_CODE on failure.

**5.2.5.31 int swd\_cmd\_queue\_append\_swdpreset ( swd\_ctx\_t \* swdctx )**

Append command queue with SW-DP-RESET sequence.

**Parameters**

*\*swdctx* swd context pointer.

**Returns**

number of elements appended, or SWD\_ERROR\_CODE on failure.

**5.2.5.32 swd\_cmd\_t\* swd\_cmd\_queue\_find\_root ( swd\_cmd\_t \* cmdq )**

Find queue root (first element).

**Parameters**

*\*cmdq* pointer to any queue element

**Returns**

swd\_cmd\_t\* pointer to the first element (root), NULL on failure

**5.2.5.33 swd\_cmd\_t\* swd\_cmd\_queue\_find\_tail ( swd\_cmd\_t \* cmdq )**

Find queue tail (last element).

**Parameters**

*\*cmdq* pointer to any queue element

**Returns**

swd\_cmd\_t\* pointer to the last element (tail), NULL on failure

**5.2.5.34 int swd\_cmd\_queue\_flush ( swd\_ctx\_t \* *swdctx*, swd\_operation\_t *operation* )**

Flush command queue contents into interface driver.

Operation is specified by SWD\_OPERATION and can be used to select how to flush the queue, ie. head-only, tail-only, one, all, etc.

**Parameters**

*\*swdctx* swd context pointer.

*operation* tells how to flush the queue.

**Returns**

number of commands transmitted, or SWD\_ERROR\_CODE on failure.

**5.2.5.35 int swd\_cmd\_queue\_free ( swd\_cmd\_t \* *cmdq* )**

Free queue pointed by \*cmdq element.

**Parameters**

*\*cmdq* pointer to any element on command queue

**Returns**

number of elements destroyed, SWD\_ERROR\_CODE on failure

**5.2.5.36 int swd\_cmd\_queue\_free\_head ( swd\_cmd\_t \* *cmdq* )**

Free queue head up to \*cmdq element.

**Parameters**

*\*cmdq* pointer to the element that becomes new queue root.

**Returns**

number of elements destroyed, or SWD\_ERROR\_CODE on failure.

**5.2.5.37 int swd\_cmd\_queue\_free\_tail ( swd\_cmd\_t \* *cmdq* )**

Free queue tail starting after \*cmdq element.

**Parameters**

*\*cmdq* pointer to the last element on the new queue.

**Returns**

number of elements destroyed, or SWD\_ERROR\_CODE on failure.

**5.2.5.38 int swd\_cmd\_queue\_init ( swd\_cmd\_t \* cmdq )**

Initialize new queue element in memory that becomes a queue root.

**Parameters**

\**cmdq* pointer to the command queue element of type [swd\\_cmd\\_t](#)

**Returns**

SWD\_OK on success, SWD\_ERROR\_CODE code on failure

**5.2.5.39 int swd\_deinit ( swd\_ctx\_t \* swdctx )**

De-initialize selected swd context and its command queue.

**Parameters**

\**swdctx* swd context pointer.

**Returns**

number of elements freed, or SWD\_ERROR\_CODE on failure.

**5.2.5.40 int swd\_deinit\_cmdq ( swd\_ctx\_t \* swdctx )**

De-initialize command queue and free its memory on selected swd context.

**Parameters**

\**swdctx* swd context pointer.

**Returns**

number of commands freed, or SWD\_ERROR\_CODE on failure.

**5.2.5.41 int swd\_deinit\_ctx ( swd\_ctx\_t \* swdctx )**

De-initialize selected swd context and free its memory.

Note: This function will not free command queue for selected context!

**Parameters**

\**swdctx* swd context pointer.

**Returns**

SWD\_OK on success, SWD\_ERROR\_CODE on failure.

**5.2.5.42** `int swd_idcode ( swd_ctx_t * swdctx, swd_operation_t operation, int * idcode, char * ack, char * parity )`

Read target's IDCODE register value.

#### Parameters

- \*swdctx* swd context pointer.
- operation* type of action to perform (queue or execute).
- \*idcode* resulting register value pointer.
- \*ack* resulting acknowledge response value pointer.
- \*parity* resulting data parity value pointer.

#### Returns

number of elements processed on the queue, or SWD\_ERROR\_CODE on failure.

**5.2.5.43** `swd_ctx_t* swd_init ( void )`

LibSWD initialization routine.

It should be called prior any operation made with libswd. It initializes command queue and basic parameters for context that is returned as pointer.

#### Returns

pointer to the initialized swd context.

**5.2.5.44** `int swd_miso_ack ( swd_ctx_t * swdctx, swd_operation_t operation, char * ack )`

Perform ACK read into \*ack and verify received data.

#### Parameters

- \*swdctx* swd context pointer.
- operation* type of action to perform with generated request.
- \*ack* pointer to the result location.

#### Returns

number of commands processed, or SWD\_ERROR\_CODE on failure.

**5.2.5.45** `int swd_miso_data_p ( swd_ctx_t * swdctx, swd_operation_t operation, int * data, char * parity )`

Perform (MISO) data read.

#### Parameters

- \*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

*\*parity* payload parity value pointer.

### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.46 int swd\_mosi\_data\_ap ( swd\_ctx\_t \* swdctx, swd\_operation\_t operation, int \* data )

Perform (MOSI) data write with automatic parity calculation.

### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.47 int swd\_mosi\_data\_p ( swd\_ctx\_t \* swdctx, swd\_operation\_t operation, int \* data, char \* parity )

Perform (MOSI) data write with provided parity value.

### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform on generated command.

*\*data* payload value pointer.

*\*parity* payload parity value pointer.

### Returns

number of elements processed, or SWD\_ERROR\_CODE on failure.

#### 5.2.5.48 int swd\_mosi\_request ( swd\_ctx\_t \* swdctx, swd\_operation\_t operation, char \* APnDP, char \* RnW, char \* addr )

Perform Request.

### Parameters

*\*swdctx* swd context pointer.

*operation* type of action to perform with generated request.

*\*APnDP* AccessPort (high) or DebugPort (low) access value pointer.

*\*RnW* Read (high) or Write (low) access value pointer.

*\*addr* target register address value pointer.

**Returns**

number of commands processed, or SWD\_ERROR\_CODE on failure.

# Index

- abort
  - swd\_swdp\_t, [11](#)
- ack
  - swd\_cmd\_t, [9](#)
  - swd\_swdp\_t, [11](#)
- AHB\_AP\_BD0
  - libswd.h, [33](#)
- AHB\_AP\_BD1
  - libswd.h, [33](#)
- AHB\_AP\_BD2
  - libswd.h, [33](#)
- AHB\_AP\_BD3
  - libswd.h, [33](#)
- AHB\_AP\_CONTROLSTATUS
  - libswd.h, [33](#)
- AHB\_AP\_DROMT
  - libswd.h, [33](#)
- AHB\_AP\_DRW
  - libswd.h, [33](#)
- AHB\_AP\_IDR
  - libswd.h, [33](#)
- AHB\_AP\_TAR
  - libswd.h, [33](#)
- bd0
  - swd\_ahbap\_t, [8](#)
- bd1
  - swd\_ahbap\_t, [8](#)
- bd2
  - swd\_ahbap\_t, [8](#)
- bd3
  - swd\_ahbap\_t, [8](#)
- bits
  - swd\_cmd\_t, [9](#)
- cmdq
  - swd\_ctx\_t, [10](#)
- cmdtype
  - swd\_cmd\_t, [9](#)
- config
  - swd\_ctx\_t, [10](#)
- control
  - swd\_cmd\_t, [9](#)
- controlstatus
  - swd\_ahbap\_t, [8](#)
- ctrlstat
  - swd\_swdp\_t, [11](#)
- device
  - swd\_driver\_t, [11](#)
- done
  - swd\_cmd\_t, [9](#)
- driver
  - swd\_ctx\_t, [10](#)
- dromt
  - swd\_ahbap\_t, [8](#)
- drw
  - swd\_ahbap\_t, [8](#)
- idcode
  - swd\_swdp\_t, [11](#)
- idr
  - swd\_ahbap\_t, [8](#)
- initialized
  - swd\_context\_config\_t, [10](#)
- libswd.c, [13](#)
  - swd\_bin32\_bitswap, [14](#)
  - swd\_bin32\_parity\_even, [14](#)
  - swd\_bin32\_print, [15](#)
  - swd\_bin32\_string, [15](#)
  - swd\_bin8\_bitswap, [15](#)
  - swd\_bin8\_parity\_even, [15](#)
  - swd\_bin8\_print, [16](#)
  - swd\_bin8\_string, [16](#)
  - swd\_bus\_setdir\_miso, [16](#)
  - swd\_bus\_setdir\_mosi, [16](#)
  - swd\_cmd\_append\_mosi\_n\_data\_ap, [17](#)
  - swd\_cmd\_append\_mosi\_n\_data\_p, [17](#)
  - swd\_cmd\_queue\_append, [17](#)
  - swd\_cmd\_queue\_append\_jtag2swd, [17](#)
  - swd\_cmd\_queue\_append\_miso\_ack, [18](#)
  - swd\_cmd\_queue\_append\_miso\_data, [18](#)
  - swd\_cmd\_queue\_append\_miso\_data\_p, [18](#)
  - swd\_cmd\_queue\_append\_miso\_n\_data\_p, [18](#)
  - swd\_cmd\_queue\_append\_miso\_nbit, [19](#)
  - swd\_cmd\_queue\_append\_miso\_parity, [19](#)
  - swd\_cmd\_queue\_append\_miso\_trn, [19](#)
  - swd\_cmd\_queue\_append\_mosi\_control, [20](#)
  - swd\_cmd\_queue\_append\_mosi\_data, [20](#)

- swd\_cmd\_queue\_append\_mosi\_data\_ap, 20
- swd\_cmd\_queue\_append\_mosi\_data\_p, 20
- swd\_cmd\_queue\_append\_mosi\_nbit, 21
- swd\_cmd\_queue\_append\_mosi\_parity, 21
- swd\_cmd\_queue\_append\_mosi\_request, 21
- swd\_cmd\_queue\_append\_mosi\_trn, 22
- swd\_cmd\_queue\_append\_swd2jtag, 22
- swd\_cmd\_queue\_append\_swdpreset, 22
- swd\_cmd\_queue\_find\_root, 22
- swd\_cmd\_queue\_find\_tail, 22
- swd\_cmd\_queue\_flush, 23
- swd\_cmd\_queue\_free, 23
- swd\_cmd\_queue\_free\_head, 23
- swd\_cmd\_queue\_free\_tail, 23
- swd\_cmd\_queue\_init, 24
- swd\_deinit, 24
- swd\_deinit\_cmdq, 24
- swd\_deinit\_ctx, 24
- swd\_error\_string, 25
- swd\_idcode, 25
- swd\_init, 25
- swd\_log, 25
- swd\_miso\_ack, 25
- swd\_miso\_data\_p, 26
- swd\_mosi\_data\_ap, 26
- swd\_mosi\_data\_p, 26
- swd\_mosi\_request, 26
- libswd.h, 27
  - AHB\_AP\_BD0, 33
  - AHB\_AP\_BD1, 33
  - AHB\_AP\_BD2, 33
  - AHB\_AP\_BD3, 33
  - AHB\_AP\_CONTROLSTATUS, 33
  - AHB\_AP\_DROMT, 33
  - AHB\_AP\_DRW, 33
  - AHB\_AP\_IDR, 33
  - AHB\_AP\_TAR, 33
  - SWD\_CMDTYPE\_MISO, 34
  - SWD\_CMDTYPE\_MISO\_ACK, 34
  - SWD\_CMDTYPE\_MISO\_BITBANG, 34
  - SWD\_CMDTYPE\_MISO\_DATA, 34
  - SWD\_CMDTYPE\_MISO\_PARITY, 34
  - SWD\_CMDTYPE\_MISO\_TRN, 34
  - SWD\_CMDTYPE\_MOSI, 34
  - SWD\_CMDTYPE\_MOSI\_BITBANG, 34
  - SWD\_CMDTYPE\_MOSI\_CONTROL, 34
  - SWD\_CMDTYPE\_MOSI\_DATA, 34
  - SWD\_CMDTYPE\_MOSI\_PARITY, 34
  - SWD\_CMDTYPE\_MOSI\_REQUEST, 34
  - SWD\_CMDTYPE\_MOSI\_TRN, 34
  - SWD\_CMDTYPE\_UNDEFINED, 34
  - SWD\_DEBUG, 34
  - SWD\_DIR\_LSBFIRST, 34
  - SWD\_DIR\_MSBFIRST, 34
  - SWD\_ERROR, 34
  - SWD\_ERROR\_ACK, 35
  - SWD\_ERROR\_ACK\_FAULT, 35
  - SWD\_ERROR\_ACK\_WAIT, 35
  - SWD\_ERROR\_ACKMISMATCH, 35
  - SWD\_ERROR\_ACKMISSING, 35
  - SWD\_ERROR\_ACKNOTDONE, 35
  - SWD\_ERROR\_ACKORDER, 35
  - SWD\_ERROR\_ACKUNKNOWN, 35
  - SWD\_ERROR\_ADDR, 35
  - SWD\_ERROR\_ApNdp, 35
  - SWD\_ERROR\_BADCMDDATA, 35
  - SWD\_ERROR\_BADCMDTYPE, 35
  - SWD\_ERROR\_BADOPCODE, 35
  - SWD\_ERROR\_DATAADDR, 35
  - SWD\_ERROR\_DEFINITION, 35
  - SWD\_ERROR\_DRIVER, 35
  - SWD\_ERROR\_GENERAL, 34
  - SWD\_ERROR\_NODATACMD, 35
  - SWD\_ERROR\_NOPARITYCMD, 35
  - SWD\_ERROR\_NOTDONE, 35
  - SWD\_ERROR\_NULLCONTEXT, 35
  - SWD\_ERROR\_NULLPOINTER, 34
  - SWD\_ERROR\_NULLQUEUE, 34
  - SWD\_ERROR\_NULLTRN, 35
  - SWD\_ERROR\_OUTOFMEM, 35
  - SWD\_ERROR\_PARAM, 35
  - SWD\_ERROR\_PARITY, 35
  - SWD\_ERROR\_PARITYADDR, 35
  - SWD\_ERROR\_QUEUE, 35
  - SWD\_ERROR\_QUEUENOTFREE, 35
  - SWD\_ERROR\_QUEUEUEROOT, 35
  - SWD\_ERROR\_RANGE, 35
  - SWD\_ERROR\_RESULT, 35
  - SWD\_ERROR\_RnW, 35
  - SWD\_ERROR\_TRANSPORT, 35
  - SWD\_ERROR\_TURNAROUND, 35
  - SWD\_FALSE, 33
  - SWD\_LOGLEVEL\_DEBUG, 35
  - SWD\_LOGLEVEL\_ERROR, 35
  - SWD\_LOGLEVEL\_INFO, 35
  - SWD\_LOGLEVEL\_SILENT, 35
  - SWD\_LOGLEVEL\_WARNING, 35
  - SWD\_OK, 34
  - SWD\_OPERATION\_EXECUTE, 36
  - SWD\_OPERATION\_FIRST, 36
  - SWD\_OPERATION\_LAST, 36
  - SWD\_OPERATION\_QUEUE, 36
  - SWD\_OPERATION\_QUEUE\_APPEND, 36
  - SWD\_OPERATION\_TRANSMIT\_ALL, 36
  - SWD\_OPERATION\_TRANSMIT\_HEAD, 36
  - SWD\_OPERATION\_TRANSMIT\_LAST, 36
  - SWD\_OPERATION\_TRANSMIT\_ONE, 36
  - SWD\_OPERATION\_TRANSMIT\_TAIL, 36



- SWD\_SILENT, 34
- SWD\_TRUE, 33
- SWD\_WARNING, 34
- SWD\_ABORT\_BITNUM\_DAPABORT, 33
- SWD\_ABORT\_BITNUM\_DORUNERRCLR, 33
- SWD\_ABORT\_BITNUM\_DSTKCMPLR, 33
- SWD\_ABORT\_BITNUM\_DSTKERRCLR, 33
- SWD\_ABORT\_BITNUM\_DWDERRCLR, 33
- SWD\_ACK\_BITLEN, 33
- SWD\_ACK\_FAULT, 33
- SWD\_ACK\_OK, 33
- SWD\_ACK\_WAIT, 33
- SWD\_ADDR\_MAXVAL, 33
- SWD\_ADDR\_MINVAL, 33
- swd\_bin32\_bitswap, 36
- swd\_bin32\_parity\_even, 36
- swd\_bin32\_print, 36
- swd\_bin32\_string, 37
- swd\_bin8\_bitswap, 37
- swd\_bin8\_parity\_even, 37
- swd\_bin8\_print, 37
- swd\_bin8\_string, 38
- swd\_bit8\_gen\_request, 38
- swd\_bool\_t, 33
- swd\_bus\_setdir\_miso, 38
- swd\_bus\_setdir\_mosi, 38
- swd\_cmd\_append\_mosi\_n\_data\_ap, 38
- swd\_cmd\_append\_mosi\_n\_data\_p, 39
- swd\_cmd\_queue\_append, 39
- swd\_cmd\_queue\_append\_jtag2swd, 39
- swd\_cmd\_queue\_append\_miso\_ack, 39
- swd\_cmd\_queue\_append\_miso\_data, 40
- swd\_cmd\_queue\_append\_miso\_data\_p, 40
- swd\_cmd\_queue\_append\_miso\_n\_data\_p, 40
- swd\_cmd\_queue\_append\_miso\_nbit, 41
- swd\_cmd\_queue\_append\_miso\_parity, 41
- swd\_cmd\_queue\_append\_miso\_trn, 41
- swd\_cmd\_queue\_append\_mosi\_control, 41
- swd\_cmd\_queue\_append\_mosi\_data, 42
- swd\_cmd\_queue\_append\_mosi\_data\_ap, 42
- swd\_cmd\_queue\_append\_mosi\_data\_p, 42
- swd\_cmd\_queue\_append\_mosi\_nbit, 43
- swd\_cmd\_queue\_append\_mosi\_parity, 43
- swd\_cmd\_queue\_append\_mosi\_request, 43
- swd\_cmd\_queue\_append\_mosi\_trn, 43
- swd\_cmd\_queue\_append\_swd2jtag, 44
- swd\_cmd\_queue\_append\_swdpreset, 44
- swd\_cmd\_queue\_find\_end, 44
- swd\_cmd\_queue\_find\_root, 44
- swd\_cmd\_queue\_flush, 44
- swd\_cmd\_queue\_free, 45
- swd\_cmd\_queue\_free\_head, 45
- swd\_cmd\_queue\_free\_tail, 45
- swd\_cmd\_queue\_init, 45
- swd\_cmd\_t, 33
- SWD\_CMDQLEN\_DEFAULT, 33
- SWD\_CMDTYPE, 34
- swd\_cmdtype\_t, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCDBGPWRUPACK, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCDBGPWRUPREQ, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCDBGRSTACK, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCDBGRSTREQ, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCSYSPWRUPACK, 33
- SWD\_CTRLSTAT\_BITNUM\_-OCSYSPWRUPREQ, 33
- SWD\_CTRLSTAT\_BITNUM\_-OMASKLANE, 33
- SWD\_CTRLSTAT\_BITNUM\_OREADOK, 33
- SWD\_CTRLSTAT\_BITNUM\_-ORUNDETECT, 33
- SWD\_CTRLSTAT\_BITNUM\_-OSTICKYCMP, 33
- SWD\_CTRLSTAT\_BITNUM\_-OSTICKYERR, 33
- SWD\_CTRLSTAT\_BITNUM\_-OSTICKYORUN, 33
- SWD\_CTRLSTAT\_BITNUM\_OTRNCNT, 33
- SWD\_CTRLSTAT\_BITNUM\_OTRNMODE, 33
- SWD\_CTRLSTAT\_BITNUM\_-OWDATAERR, 33
- SWD\_DATA\_BITLEN, 33
- SWD\_DATA\_BYTESIZE, 33
- SWD\_DATA\_MAXBITCOUNT, 33
- SWD\_DEBUGLEVEL, 34
- swd\_debuglevel\_t, 33
- swd\_deinit, 46
- swd\_deinit\_cmdq, 46
- swd\_deinit\_ctx, 46
- SWD\_DIRECTION, 34
- swd\_direction\_t, 33
- SWD\_DP\_ADDR\_ABORT, 33
- SWD\_DP\_ADDR\_CTRLSTAT, 33
- SWD\_DP\_ADDR\_IDCODE, 33
- SWD\_DP\_ADDR\_RDBUF, 33
- SWD\_DP\_ADDR\_RESEND, 33
- SWD\_DP\_ADDR\_SELECT, 33
- SWD\_DP\_ADDR\_WCR, 33
- swd\_drv\_miso\_32, 46

- swd\_drv\_miso\_8, 47
- swd\_drv\_miso\_trn, 47
- swd\_drv\_mosi\_32, 47
- swd\_drv\_mosi\_8, 47
- swd\_drv\_mosi\_trn, 47
- SWD\_ERROR\_CODE, 34
- swd\_error\_code\_t, 33
- swd\_error\_string, 47
- swd\_idcode, 47
- swd\_init, 47
- swd\_log, 47
- SWD\_LOGLEVEL, 35
- swd\_loglevel\_t, 33
- SWD\_MASKLANE\_0, 33
- SWD\_MASKLANE\_1, 33
- SWD\_MASKLANE\_2, 33
- SWD\_MASKLANE\_3, 33
- swd\_miso\_ack, 47
- swd\_miso\_data\_p, 48
- swd\_mosi\_data\_ap, 48
- swd\_mosi\_data\_p, 48
- swd\_mosi\_request, 48
- SWD\_OPERATION, 35
- swd\_operation\_t, 33
- SWD\_REQUEST\_A2\_BITNUM, 33
- SWD\_REQUEST\_A3\_BITNUM, 33
- SWD\_REQUEST\_ADDR\_BITNUM, 33
- SWD\_REQUEST\_APnDP\_BITNUM, 33
- SWD\_REQUEST\_BITLEN, 33
- SWD\_REQUEST\_PARITY\_BITNUM, 33
- SWD\_REQUEST\_PARK\_BITNUM, 33
- SWD\_REQUEST\_PARK\_VAL, 33
- SWD\_REQUEST\_RnW\_BITNUM, 33
- SWD\_REQUEST\_START\_BITNUM, 33
- SWD\_REQUEST\_START\_VAL, 33
- SWD\_REQUEST\_STOP\_BITNUM, 33
- SWD\_REQUEST\_STOP\_VAL, 33
- SWD\_SELECT\_BITNUM\_APBANKSEL, 33
- SWD\_SELECT\_BITNUM\_APSEL, 33
- SWD\_SELECT\_BITNUM\_CTRLSEL, 33
- swd\_transfer\_cmd, 49
- swd\_transmit, 49
- SWD\_TURNROUND\_1, 33
- SWD\_TURNROUND\_2, 33
- SWD\_TURNROUND\_3, 33
- SWD\_TURNROUND\_4, 33
- SWD\_TURNROUND\_DEFAULT, 33
- SWD\_TURNROUND\_MAX, 33
- SWD\_TURNROUND\_MIN, 33
- SWD\_WCR\_BITNUM\_PRESCALER, 33
- SWD\_WCR\_BITNUM\_TURNROUND, 33
- SWD\_WCR\_BITNUM\_WIREMODE, 33
- libswd\_drv\_dummy.c, 49
  - swd\_drv\_miso\_32, 50
  - swd\_drv\_miso\_8, 50
  - swd\_drv\_miso\_trn, 50
  - swd\_drv\_mosi\_32, 50
  - swd\_drv\_mosi\_8, 50
  - swd\_drv\_mosi\_trn, 50
- libswd\_drv\_urjtag.c, 50
  - swd\_drv\_miso\_32, 50
  - swd\_drv\_miso\_8, 50
  - swd\_drv\_miso\_trn, 50
  - swd\_drv\_mosi\_32, 50
  - swd\_drv\_mosi\_8, 50
  - swd\_drv\_mosi\_trn, 50
- libswd\_test.c, 50
  - main, 51
- loglevel
  - swd\_context\_config\_t, 10
- main
  - libswd\_test.c, 51
- maxcmdqlen
  - swd\_context\_config\_t, 10
- misoahbap
  - swd\_ctx\_t, 10
- misobit
  - swd\_cmd\_t, 9
- misodata
  - swd\_cmd\_t, 9
- misoswdp
  - swd\_ctx\_t, 10
- mosiahbap
  - swd\_ctx\_t, 10
- mosibit
  - swd\_cmd\_t, 9
- mosidata
  - swd\_cmd\_t, 9
- mosiswdp
  - swd\_ctx\_t, 10
- next
  - swd\_cmd\_t, 9
- parity
  - swd\_cmd\_t, 9
- prev
  - swd\_cmd\_t, 9
- rdbuf
  - swd\_swdp\_t, 11
- request
  - swd\_cmd\_t, 9
- select
  - swd\_swdp\_t, 11
- SWD\_CMDTYPE\_MISO
  - libswd.h, 34

- SWD\_CMDTYPE\_MISO\_ACK
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MISO\_BITBANG
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MISO\_DATA
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MISO\_PARITY
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MISO\_TRN
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_BITBANG
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_CONTROL
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_DATA
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_PARITY
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_REQUEST
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_MOSI\_TRN
  - [libswd.h, 34](#)
- SWD\_CMDTYPE\_UNDEFINED
  - [libswd.h, 34](#)
- SWD\_DEBUG
  - [libswd.h, 34](#)
- SWD\_DIR\_LSBFIRST
  - [libswd.h, 34](#)
- SWD\_DIR\_MSBFIRST
  - [libswd.h, 34](#)
- SWD\_ERROR
  - [libswd.h, 34](#)
- SWD\_ERROR\_ACK
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACK\_FAULT
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACK\_WAIT
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACKMISMATCH
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACKMISSING
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACKNOTDONE
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACKORDER
  - [libswd.h, 35](#)
- SWD\_ERROR\_ACKUNKNOWN
  - [libswd.h, 35](#)
- SWD\_ERROR\_ADDR
  - [libswd.h, 35](#)
- SWD\_ERROR\_APnDP
  - [libswd.h, 35](#)
- SWD\_ERROR\_BADCMDDATA
  - [libswd.h, 35](#)
- SWD\_ERROR\_BADCMDTYPE
  - [libswd.h, 35](#)
- SWD\_ERROR\_BADOPCODE
  - [libswd.h, 35](#)
- SWD\_ERROR\_DATAADDR
  - [libswd.h, 35](#)
- SWD\_ERROR\_DEFINITION
  - [libswd.h, 35](#)
- SWD\_ERROR\_DRIVER
  - [libswd.h, 35](#)
- SWD\_ERROR\_GENERAL
  - [libswd.h, 34](#)
- SWD\_ERROR\_NODATACMD
  - [libswd.h, 35](#)
- SWD\_ERROR\_NOPARITYCMD
  - [libswd.h, 35](#)
- SWD\_ERROR\_NOTDONE
  - [libswd.h, 35](#)
- SWD\_ERROR\_NULLCONTEXT
  - [libswd.h, 35](#)
- SWD\_ERROR\_NULLPOINTER
  - [libswd.h, 34](#)
- SWD\_ERROR\_NULLQUEUE
  - [libswd.h, 34](#)
- SWD\_ERROR\_NULLTRN
  - [libswd.h, 35](#)
- SWD\_ERROR\_OUTOFMEM
  - [libswd.h, 35](#)
- SWD\_ERROR\_PARAM
  - [libswd.h, 35](#)
- SWD\_ERROR\_PARITY
  - [libswd.h, 35](#)
- SWD\_ERROR\_PARITYADDR
  - [libswd.h, 35](#)
- SWD\_ERROR\_QUEUE
  - [libswd.h, 35](#)
- SWD\_ERROR\_QUEUENOTFREE
  - [libswd.h, 35](#)
- SWD\_ERROR\_QUEUEROOT
  - [libswd.h, 35](#)
- SWD\_ERROR\_RANGE
  - [libswd.h, 35](#)
- SWD\_ERROR\_RESULT
  - [libswd.h, 35](#)
- SWD\_ERROR\_RnW
  - [libswd.h, 35](#)
- SWD\_ERROR\_TRANSPORT
  - [libswd.h, 35](#)
- SWD\_ERROR\_TURNAROUND
  - [libswd.h, 35](#)
- SWD\_FALSE
  - [libswd.h, 33](#)

- SWD\_LOGLEVEL\_DEBUG
  - libswd.h, 35
- SWD\_LOGLEVEL\_ERROR
  - libswd.h, 35
- SWD\_LOGLEVEL\_INFO
  - libswd.h, 35
- SWD\_LOGLEVEL\_SILENT
  - libswd.h, 35
- SWD\_LOGLEVEL\_WARNING
  - libswd.h, 35
- SWD\_OK
  - libswd.h, 34
- SWD\_OPERATION\_EXECUTE
  - libswd.h, 36
- SWD\_OPERATION\_FIRST
  - libswd.h, 36
- SWD\_OPERATION\_LAST
  - libswd.h, 36
- SWD\_OPERATION\_QUEUE
  - libswd.h, 36
- SWD\_OPERATION\_QUEUE\_APPEND
  - libswd.h, 36
- SWD\_OPERATION\_TRANSMIT\_ALL
  - libswd.h, 36
- SWD\_OPERATION\_TRANSMIT\_HEAD
  - libswd.h, 36
- SWD\_OPERATION\_TRANSMIT\_LAST
  - libswd.h, 36
- SWD\_OPERATION\_TRANSMIT\_ONE
  - libswd.h, 36
- SWD\_OPERATION\_TRANSMIT\_TAIL
  - libswd.h, 36
- SWD\_SILENT
  - libswd.h, 34
- SWD\_TRUE
  - libswd.h, 33
- SWD\_WARNING
  - libswd.h, 34
- SWD\_ABORT\_BITNUM\_DAPABORT
  - libswd.h, 33
- SWD\_ABORT\_BITNUM\_DORUNERRCLR
  - libswd.h, 33
- SWD\_ABORT\_BITNUM\_DSTKCMPCLR
  - libswd.h, 33
- SWD\_ABORT\_BITNUM\_DSTKERRCLR
  - libswd.h, 33
- SWD\_ABORT\_BITNUM\_DWDERRCLR
  - libswd.h, 33
- SWD\_ACK\_BITLEN
  - libswd.h, 33
- SWD\_ACK\_FAULT
  - libswd.h, 33
- SWD\_ACK\_OK
  - libswd.h, 33
- SWD\_ACK\_WAIT
  - libswd.h, 33
- SWD\_ADDR\_MAXVAL
  - libswd.h, 33
- SWD\_ADDR\_MINVAL
  - libswd.h, 33
- swd\_ahbap\_t, 7
  - bd0, 8
  - bd1, 8
  - bd2, 8
  - bd3, 8
  - controlstatus, 8
  - dromt, 8
  - drw, 8
  - idr, 8
  - tar, 8
- swd\_bin32\_bitswap
  - libswd.c, 14
  - libswd.h, 36
- swd\_bin32\_parity\_even
  - libswd.c, 14
  - libswd.h, 36
- swd\_bin32\_print
  - libswd.c, 15
  - libswd.h, 36
- swd\_bin32\_string
  - libswd.c, 15
  - libswd.h, 37
- swd\_bin8\_bitswap
  - libswd.c, 15
  - libswd.h, 37
- swd\_bin8\_parity\_even
  - libswd.c, 15
  - libswd.h, 37
- swd\_bin8\_print
  - libswd.c, 16
  - libswd.h, 37
- swd\_bin8\_string
  - libswd.c, 16
  - libswd.h, 38
- swd\_bit8\_gen\_request
  - libswd.h, 38
- swd\_bool\_t
  - libswd.h, 33
- swd\_bus\_setdir\_miso
  - libswd.c, 16
  - libswd.h, 38
- swd\_bus\_setdir\_mosi
  - libswd.c, 16
  - libswd.h, 38
- swd\_cmd\_append\_mosi\_n\_data\_ap
  - libswd.c, 17
  - libswd.h, 38
- swd\_cmd\_append\_mosi\_n\_data\_p

- libswd.c, 17
- libswd.h, 39
- swd\_cmd\_queue\_append
  - libswd.c, 17
  - libswd.h, 39
- swd\_cmd\_queue\_append\_jtag2swd
  - libswd.c, 17
  - libswd.h, 39
- swd\_cmd\_queue\_append\_miso\_ack
  - libswd.c, 18
  - libswd.h, 39
- swd\_cmd\_queue\_append\_miso\_data
  - libswd.c, 18
  - libswd.h, 40
- swd\_cmd\_queue\_append\_miso\_data\_p
  - libswd.c, 18
  - libswd.h, 40
- swd\_cmd\_queue\_append\_miso\_n\_data\_p
  - libswd.c, 18
  - libswd.h, 40
- swd\_cmd\_queue\_append\_miso\_nbit
  - libswd.c, 19
  - libswd.h, 41
- swd\_cmd\_queue\_append\_miso\_parity
  - libswd.c, 19
  - libswd.h, 41
- swd\_cmd\_queue\_append\_miso\_trn
  - libswd.c, 19
  - libswd.h, 41
- swd\_cmd\_queue\_append\_mosi\_control
  - libswd.c, 20
  - libswd.h, 41
- swd\_cmd\_queue\_append\_mosi\_data
  - libswd.c, 20
  - libswd.h, 42
- swd\_cmd\_queue\_append\_mosi\_data\_ap
  - libswd.c, 20
  - libswd.h, 42
- swd\_cmd\_queue\_append\_mosi\_data\_p
  - libswd.c, 20
  - libswd.h, 42
- swd\_cmd\_queue\_append\_mosi\_nbit
  - libswd.c, 21
  - libswd.h, 43
- swd\_cmd\_queue\_append\_mosi\_parity
  - libswd.c, 21
  - libswd.h, 43
- swd\_cmd\_queue\_append\_mosi\_request
  - libswd.c, 21
  - libswd.h, 43
- swd\_cmd\_queue\_append\_mosi\_trn
  - libswd.c, 22
  - libswd.h, 43
- swd\_cmd\_queue\_append\_sw2jtag
  - libswd.c, 22
  - libswd.h, 44
- swd\_cmd\_queue\_append\_sw2preset
  - libswd.c, 22
  - libswd.h, 44
- swd\_cmd\_queue\_find\_end
  - libswd.h, 44
- swd\_cmd\_queue\_find\_root
  - libswd.c, 22
  - libswd.h, 44
- swd\_cmd\_queue\_find\_tail
  - libswd.c, 22
- swd\_cmd\_queue\_flush
  - libswd.c, 23
  - libswd.h, 44
- swd\_cmd\_queue\_free
  - libswd.c, 23
  - libswd.h, 45
- swd\_cmd\_queue\_free\_head
  - libswd.c, 23
  - libswd.h, 45
- swd\_cmd\_queue\_free\_tail
  - libswd.c, 23
  - libswd.h, 45
- swd\_cmd\_queue\_init
  - libswd.c, 24
  - libswd.h, 45
- swd\_cmd\_t, 8
  - ack, 9
  - bits, 9
  - cmdtype, 9
  - control, 9
  - done, 9
  - libswd.h, 33
  - misobit, 9
  - misodata, 9
  - mosibit, 9
  - mosidata, 9
  - next, 9
  - parity, 9
  - prev, 9
  - request, 9
  - TRNnMOSI, 9
- SWD\_CMDQLEN\_DEFAULT
  - libswd.h, 33
- SWD\_CMDTYPE
  - libswd.h, 34
- swd\_cmdtype\_t
  - libswd.h, 33
- swd\_context\_config\_t, 9
  - initialized, 10
  - loglevel, 10
  - maxcmdqlen, 10
  - trnlen, 10

- SWD\_CTRLSTAT\_BITNUM\_-  
  OCDBGPWRUPACK  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_-  
  OCDBGPWRUPREQ  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OCDBGRSTACK  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OCDBGRSTREQ  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_-  
  OCSYSPWRUPACK  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_-  
  OCSYSPWRUPREQ  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OMASKLANE  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OREADOK  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_ORUNDETECT  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OSTICKYCMP  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OSTICKYERR  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OSTICKYORUN  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OTRNCNT  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OTRNMODE  
    libswd.h, 33
- SWD\_CTRLSTAT\_BITNUM\_OWDATAERR  
    libswd.h, 33
- swd\_ctx\_t, 10
  - cmdq, 10
  - config, 10
  - driver, 10
  - misoahbap, 10
  - misoswdp, 10
  - mosiahbap, 10
  - mosiswdp, 10
- SWD\_DATA\_BITLEN  
    libswd.h, 33
- SWD\_DATA\_BYTESIZE  
    libswd.h, 33
- SWD\_DATA\_MAXBITCOUNT  
    libswd.h, 33
- SWD\_DEBUGLEVEL  
    libswd.h, 34
- swd\_debuglevel\_t  
    libswd.h, 33
- swd\_deinit  
    libswd.c, 24
- libswd.h, 46
- swd\_deinit\_cmdq  
    libswd.c, 24
- libswd.h, 46
- swd\_deinit\_ctx  
    libswd.c, 24
- libswd.h, 46
- SWD\_DIRECTION  
    libswd.h, 34
- swd\_direction\_t  
    libswd.h, 33
- SWD\_DP\_ADDR\_ABORT  
    libswd.h, 33
- SWD\_DP\_ADDR\_CTRLSTAT  
    libswd.h, 33
- SWD\_DP\_ADDR\_IDCODE  
    libswd.h, 33
- SWD\_DP\_ADDR\_RDBUF  
    libswd.h, 33
- SWD\_DP\_ADDR\_RESEND  
    libswd.h, 33
- SWD\_DP\_ADDR\_SELECT  
    libswd.h, 33
- SWD\_DP\_ADDR\_WCR  
    libswd.h, 33
- swd\_driver\_t, 11
  - device, 11
- swd\_drv\_miso\_32  
    libswd.h, 46
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- swd\_drv\_miso\_8  
    libswd.h, 47
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- swd\_drv\_miso\_trn  
    libswd.h, 47
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- swd\_drv\_mosi\_32  
    libswd.h, 47
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- swd\_drv\_mosi\_8  
    libswd.h, 47
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- swd\_drv\_mosi\_trn  
    libswd.h, 47
- libswd\_drv\_dummy.c, 50
- libswd\_drv\_urjtag.c, 50
- SWD\_ERROR\_CODE  
    libswd.h, 34
- swd\_error\_code\_t

- libswd.h, 33
- swd\_error\_string
  - libswd.c, 25
  - libswd.h, 47
- swd\_idcode
  - libswd.c, 25
  - libswd.h, 47
- swd\_init
  - libswd.c, 25
  - libswd.h, 47
- swd\_log
  - libswd.c, 25
  - libswd.h, 47
- SWD\_LOGLEVEL
  - libswd.h, 35
- swd\_loglevel\_t
  - libswd.h, 33
- SWD\_MASKLANE\_0
  - libswd.h, 33
- SWD\_MASKLANE\_1
  - libswd.h, 33
- SWD\_MASKLANE\_2
  - libswd.h, 33
- SWD\_MASKLANE\_3
  - libswd.h, 33
- swd\_miso\_ack
  - libswd.c, 25
  - libswd.h, 47
- swd\_miso\_data\_p
  - libswd.c, 26
  - libswd.h, 48
- swd\_mosi\_data\_ap
  - libswd.c, 26
  - libswd.h, 48
- swd\_mosi\_data\_p
  - libswd.c, 26
  - libswd.h, 48
- swd\_mosi\_request
  - libswd.c, 26
  - libswd.h, 48
- SWD\_OPERATION
  - libswd.h, 35
- swd\_operation\_t
  - libswd.h, 33
- SWD\_REQUEST\_A2\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_A3\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_ADDR\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_APnDP\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_BITLEN
  - libswd.h, 33
- SWD\_REQUEST\_PARITY\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_PARK\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_PARK\_VAL
  - libswd.h, 33
- SWD\_REQUEST\_RnW\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_START\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_START\_VAL
  - libswd.h, 33
- SWD\_REQUEST\_STOP\_BITNUM
  - libswd.h, 33
- SWD\_REQUEST\_STOP\_VAL
  - libswd.h, 33
- SWD\_SELECT\_BITNUM\_APBANKSEL
  - libswd.h, 33
- SWD\_SELECT\_BITNUM\_APSEL
  - libswd.h, 33
- SWD\_SELECT\_BITNUM\_CTRLSEL
  - libswd.h, 33
- swd\_swdp\_t, 11
  - abort, 11
  - ack, 11
  - ctrlstat, 11
  - idcode, 11
  - rdbuf, 11
  - select, 11
  - wcr, 11
- swd\_transfer\_cmd
  - libswd.h, 49
- swd\_transmit
  - libswd.h, 49
- SWD\_TURNROUND\_1
  - libswd.h, 33
- SWD\_TURNROUND\_2
  - libswd.h, 33
- SWD\_TURNROUND\_3
  - libswd.h, 33
- SWD\_TURNROUND\_4
  - libswd.h, 33
- SWD\_TURNROUND\_DEFAULT
  - libswd.h, 33
- SWD\_TURNROUND\_MAX
  - libswd.h, 33
- SWD\_TURNROUND\_MIN
  - libswd.h, 33
- SWD\_WCR\_BITNUM\_PRESCALER
  - libswd.h, 33
- SWD\_WCR\_BITNUM\_TURNROUND
  - libswd.h, 33
- SWD\_WCR\_BITNUM\_WIREMODE
  - libswd.h, 33

tar

swd\_ahbap\_t, [8](#)

trnlen

swd\_context\_config\_t, [10](#)

TRNnMOSI

swd\_cmd\_t, [9](#)

wcr

swd\_swdp\_t, [11](#)