EtherSpaceLink C++ Reference

Generated by Doxygen 1.8.5

Wed Aug 5 2020 16:17:49

Contents

1 Module Index **1**

1.1 Modules . . . 1

2 Namespace Index **3**

2.1 Namespace List . . . 3

3 Hierarchical Index **5**

3.1 Class Hierarchy . . . 5

4 Class Index **7**

4.1 Class List . . . 7

5 Module Documentation **9**

5.1 Connection . . . 9

5.2 Physical Link Attributes . . . 10

5.3 Virtual Link Attributes . . . 11

5.3.1 Detailed Description . . . 11

5.4 Handling Spacewire Traffic . . . 12

5.4.1 Detailed Description . . . 12

5.5 Event handling on spacewire links . . . 13

5.5.1 Detailed Description . . . 13

5.6 Sending data on a spaceiwre link . . . 14

5.7 Reading data from a spacewire link . . . 15

5.8 TimeTag . . . 16

5.8.1 Detailed Description . . . 16

5.9 Error Reporting . . . 17

5.9.1 Detailed Description . . . 17

5.10 Error Waveforms . . . 18

5.10.1 Detailed Description . . . 18

5.11 Extension codes . . . 19

5.11.1 Detailed Description . . . 20

5.12 TimeTag mask fields . . . 21

5.12.1 Detailed Description . . . 21

5.13 Error mask fields . . . 22
5.13.1 Detailed Description . . . 22
5.14 Error Waveform Triggers . . . 23
5.14.1 Detailed Description . . . 23
5.15 Error Waveform Sources . . . 24
5.15.1 Detailed Description . . . 24
5.16 Memory Mapped Addresses . . . 25
5.16.1 Detailed Description . . . 25
5.17 Error Codes . . . 26
5.17.1 Detailed Description . . . 27
5.18 functions . . . 28
5.18.1 Detailed Description . . . 28
5.18.2 Function Documentation . . . 28
5.18.2.1 barrier_lifted . . . 28
5.18.2.2 err . . . 29
5.18.2.3 error_event . . . 29
5.18.2.4 esc_eep . . . 29
5.18.2.5 esc_eop . . . 29
5.18.2.6 esc_esc . . . 29
5.18.2.7 link_selected . . . 29
5.18.2.8 link_timeout . . . 30
5.18.2.9 link_tx_speed . . . 30
5.18.2.10 missing_data . . . 30
5.18.2.11 parity_error . . . 30
5.18.2.12 perror1 . . . 30
5.18.2.13 perror2 . . . 30
5.18.2.14 port_status . . . 31
5.18.2.15 received_esc_fct . . . 31
5.18.2.16 received_fct . . . 31
5.18.2.17 reported_version . . . 31
5.18.2.18 startdate . . . 31
5.18.2.19 timecode . . . 31
5.18.2.20 timetag . . . 32
5.18.2.21 timetag_uncertainty . . . 32
5.18.2.22 timezero . . . 32
5.18.2.23 unknown_extn_data . . . 32
5.18.2.24 unknown_ram_data . . . 33
5.18.2.25 unknown_special_data . . . 33
5.18.2.26 waveform_data . . . 33

6 Namespace Documentation **35**

6.1 esl Namespace Reference . . . 35

6.1.1 Detailed Description . . . 35

7 Class Documentation **37**

7.1 esl::connection Class Reference . . . 37

7.1.1 Constructor & Destructor Documentation . . . 48

7.1.1.1 ~connection . . . 48

7.1.1.2 connection . . . 48

7.1.2 Member Function Documentation . . . 49

7.1.2.1 abort . . . 49

7.1.2.2 active_port . . . 49

7.1.2.3 complex_read . . . 49

7.1.2.4 current_time . . . 49

7.1.2.5 dump . . . 49

7.1.2.6 dump_limit . . . 49

7.1.2.7 dump_state . . . 50

7.1.2.8 ei_esc_eep . . . 50

7.1.2.9 ei_esc_eop . . . 50

7.1.2.10 ei_esc_esc . . . 50

7.1.2.11 ei_esc_fct . . . 50

7.1.2.12 EI_flow_control . . . 50

7.1.2.13 EI_ignore_events . . . 50

7.1.2.14 ei_parity_error . . . 50

7.1.2.15 eintr . . . 50

7.1.2.16 ER_reporting . . . 51

7.1.2.17 EW_clear . . . 51

7.1.2.18 EW_reporting . . . 51

7.1.2.19 EW_request . . . 51

7.1.2.20 EW_reset . . . 51

7.1.2.21 EW_source . . . 51

7.1.2.22 flush . . . 52

7.1.2.23 get_link . . . 52

7.1.2.24 get_rx_timeout . . . 52

7.1.2.25 link_connected . . . 52

7.1.2.26 log . . . 52

7.1.2.27 mac . . . 52

7.1.2.28 manufacturer . . . 52

7.1.2.29 mode . . . 53

7.1.2.30 modes . . . 54

7.1.2.31 module . . . 54
7.1.2.32 nolinks . . . 54
7.1.2.33 non_blocking . . . 54
7.1.2.34 options . . . 54
7.1.2.35 percent_read . . . 54
7.1.2.36 product . . . 55
7.1.2.37 raw . . . 55
7.1.2.38 raw_bytes . . . 55
7.1.2.39 read . . . 55
7.1.2.40 read_info . . . 55
7.1.2.41 record . . . 55
7.1.2.42 record_file . . . 56
7.1.2.43 record_flush . . . 56
7.1.2.44 record_size . . . 56
7.1.2.45 request_link_status . . . 56
7.1.2.46 request_link_status_port . . . 56
7.1.2.47 request_tx_speed . . . 56
7.1.2.48 rx_speed . . . 56
7.1.2.49 send_timecode . . . 56
7.1.2.50 sendfile . . . 57
7.1.2.51 serial . . . 57
7.1.2.52 set_link . . . 57
7.1.2.53 set_max_packet_data . . . 57
7.1.2.54 set_rx_timeout . . . 57
7.1.2.55 slot . . . 57
7.1.2.56 sma_56_pulse_width . . . 57
7.1.2.57 socket . . . 58
7.1.2.58 speed . . . 58
7.1.2.59 speed . . . 58
7.1.2.60 TT_reporting . . . 58
7.1.2.61 tx_record . . . 58
7.1.2.62 version . . . 58
7.1.2.63 write . . . 58
7.1.2.64 write_buffer_empty . . . 59
7.2 esl::devstatus Class Reference . . . 59
7.3 esl::error Class Reference . . . 60
7.3.1 Detailed Description . . . 60
7.3.2 Constructor & Destructor Documentation . . . 60
7.3.2.1 error . . . 60
7.3.2.2 error . . . 60

7.3.3 Member Function Documentation 61
7.3.3.1 get 61
7.3.3.2 oserr 61
7.3.3.3 text 61
7.4 esl::utils::lock Class Reference 61
7.5 esl::portstatus Class Reference 61
7.6 esl::utils::scopelock Class Reference 62
7.7 esl::stats Class Reference 63
7.8 esl::timestamp Struct Reference 63
7.8.1 Detailed Description 63

Index **64**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

- Connection . . . 9
- Physical Link Attributes . . . 10
- Virtual Link Attributes . . . 11
  - TimeTag . . . 16
    - TimeTag mask fields . . . 21
  - Error Reporting . . . 17
    - Error mask fields . . . 22
  - Error Waveforms . . . 18
    - Error Waveform Triggers . . . 23
    - Error Waveform Sources . . . 24
- Handling Spacewire Traffic . . . 12
  - Event handling on spacewire links . . . 13
    - functions . . . 28
  - Sending data on a spaceiwre link . . . 14
  - Reading data from a spacewire link . . . 15
  - Extension codes . . . 19
- Memory Mapped Addresses . . . 25
- Error Codes . . . 26

Chapter 2

Namespace Index

2.1 Namespace List

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

esl
The esl namespace is used for EtherSpaceLink hardware 35

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

esl::connection 37
esl::devstatus 59
exception
    esl::error 60
esl::utils::lock 61
esl::portstatus 61
esl::utils::scopelock 62
esl::stats 63
esl::timestamp 63

Chapter 4

Class Index

4.1 Class List

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

esl::connection 37
esl::devstatus 59
esl::error
Exception class, whenever an error occurrs an exception is thrown, this makes error path handling somewhat easier 60
esl::utils::lock 61
esl::portstatus 61
esl::utils::scopelock 62
esl::stats 63
esl::timestamp
Connection this class provides the base class of communicating with a DSI/MSR 63

Chapter 5

Module Documentation

5.1 Connection

This file contains the definitions of constants used to drive ESL functions.

This file contains the definitions of constants used to drive ESL functions. (c) 4Links Limited 2000-2019

These functions are used to make a connection to an EtherSpaceLink Device/File

5.2 Physical Link Attributes

Functions and definitions for controlling physical link attributes

5.3 Virtual Link Attributes

Modules

- TimeTag
- Error Reporting
- Error Waveforms

5.3.1 Detailed Description

Functions and definitions for controlling virtual link attributes

5.4 Handling Spacewire Traffic

Modules

- Event handling on spacewire links
- Sending data on a spaceiwre link
- Reading data from a spacewire link
- Extension codes

5.4.1 Detailed Description

Functions and definitions for handling spacewire traffic

5.5 Event handling on spacewire links

Modules

- functions

5.5.1 Detailed Description

Functions and definitions for handling events on spacewire links

5.6 Sending data on a spaceiwre link

Functions and definitions for sending data

5.7 Reading data from a spacewire link

Functions and definitions for reading data

5.8 TimeTag

Modules

- TimeTag mask fields

5.8.1 Detailed Description

Functions and definitions for reporting Timetags

5.9 Error Reporting

Modules

- Error mask fields

5.9.1 Detailed Description

Functions and definitions for reporting Errors

5.10 Error Waveforms

Modules

- Error Waveform Triggers
- Error Waveform Sources

5.10.1 Detailed Description

Functions and definitions for capturing waveforms

5.11 Extension codes

Variables

- static const int **esl::connection::FCT** = 0x100
- static const int esl::connection::EEP = 0x101

  Error End of Packet.
- static const int esl::connection::EOP = 0x102

  End of Packet.
- static const int esl::connection::ESC = 0x103

  Escape.
- static const int esl::connection::ESC_FCT = 0x104

  Escape FCT aka a NULL character.
- static const int esl::connection::ESC_EEP = 0x105

  Escape End of Packet.
- static const int esl::connection::ESC_EOP = 0x106

  Escape Error of packet.
- static const int esl::connection::ESC_ESC = 0x107

  Escape Escape.
- static const int esl::connection::Timeout = 0x108

  Timeout message.
- static const int esl::connection::ParityError = 0x109

  Parity Error message.
- static const int esl::connection::PERROR1 = 0x10A

  Error 1 message.
- static const int esl::connection::PERROR2 = 0x10B

  Error 2 message.
- static const int **esl::connection::STORE** = 0x10C
- static const int **esl::connection::FORWARD** = 0x10D
- static const int **esl::connection::ATOM** = 0x10E
- static const int **esl::connection::MOTA** = 0x10F
- static const int **esl::connection::JOIN** = 0x110
- static const int **esl::connection::BARRIER** = 0x111
- static const int **esl::connection::RESIGN** = 0x112
- static const int **esl::connection::EVENT** = 0x113
- static const int esl::connection::Missing_data = 0x114

  Missed data message.
- static const int **esl::connection::HOLD** = 0x12F
- static const int **esl::connection::Delay** = 0x130
- static const int esl::connection::PortSelect = 0x140

  Port select message.
- static const int esl::connection::PortSelect_max = 0x17F

  Max port select message.
- static const int **esl::connection::Multi_byte_extn_start** = 0x180
- static const int esl::connection::TimeTag = 0x188

  Timetag message.
- static const int esl::connection::TimeTag_delta = 0x182

  Timetag delta message.
- static const int esl::connection::TimeTag_uncertainty = 0x181

  Timetag uncertain message.
- static const int esl::connection::TimeCode = 0x191

  Spacewire timecode.

- static const int esl::connection::Module = 0x192

  Module data.
- static const int esl::connection::TimeZero = 0x198

  First timecode on the link.
- static const int **esl::connection::TRUNCATE_1** = 0x1A1
- static const int **esl::connection::TRUNCATE_2** = 0x1A2
- static const int **esl::connection::REPEAT_1** = 0x1B1
- static const int **esl::connection::REPEAT_2** = 0x1B2
- static const int **esl::connection::REPEAT_3** = 0x1B3
- static const int esl::connection::Year = 0x1C8

  Capture start date/time.
- static const int esl::connection::Header = 0x1CE

  Capture Header containing version and time information.

5.11.1 Detailed Description

5.12 TimeTag mask fields

Variables

- static const int **esl::connection::TT** = 7
- static const int **esl::connection::TT_64** = 15
- static const int esl::connection::TT_report_nothing = 0x00

  Report Nothing.
- static const int esl::connection::TT_report_first_byte = (0x01 | report_first_byte)

  Timetag first byte of packet.
- static const int esl::connection::TT_report_intermediate_bytes = (0x02 | report_mid_bytes)

  Timetag middle byte.
- static const int esl::connection::TT_report_EOP_EEP = (0x04 | report_EEP | report_EOP)

  Timetag end of packet markers.
- static const int esl::connection::TT_report_EEP = (0x04 | report_EEP)

  Timetag report Error End of Packet.
- static const int esl::connection::TT_report_EOP = (0x04 | report_EOP)

  Timetag report End of Packet.
- static const int esl::connection::TT_report_time_code = report_time_code

  Timetag report spacewire timecode.
- static const int esl::connection::TT_report_fct = report_FCT

  Timetag report FCT.
- static const int esl::connection::TT_report_null = report_NULL

  Timetag report NULL.
- static const int esl::connection::TT_report_parity_error = report_parity_error

  Timetag report parity error.
- static const int esl::connection::TT_report_ESC_EOP = report_ESC_EOP

  Timetag report ESC End of Packet.
- static const int esl::connection::TT_report_ESC_EEP = report_ESC_EEP

  Timetag report ESC Error End of Packet.
- static const int esl::connection::TT_report_ESC_ESC = report_ESC_ESC

  Timetag report ESC ESC.
- static const int esl::connection::TT_report_timeout = report_timeout

  Timetag report timeout.

5.12.1 Detailed Description

5.13 Error mask fields

Variables

- static const int **esl::connection::ER** = 8
- static const int **esl::connection::ER_64** = 16
- static const int esl::connection::ER_report_nothing = 0x00

  Error reporting report nothing.
- static const int esl::connection::ER_report_first_null = 0x02

  Error report first null.
- static const int esl::connection::ER_report_first_fct = 0x04

  Error report first fct.
- static const int esl::connection::ER_report_running_error = (0x08 | report_parity_error | report_ESC_EOP | report_ESC_EEP | report_ESC_ESC | report_timeout)

  Error report running.
- static const int **esl::connection::ER_report_starting_error** = 0x10
- static const int **esl::connection::ER_report_nchar** = 0x40
- static const int esl::connection::ER_report_time_code = (0x80 | report_time_code)

  report time code
- static const int esl::connection::ER_report_fct = report_FCT

  report FCT
- static const int esl::connection::ER_report_null = report_NULL

  report null
- static const int esl::connection::ER_report_parity_error = report_parity_error

  report parity error
- static const int esl::connection::ER_report_ESC_EOP = report_ESC_EOP

  report Escape End of Packet
- static const int esl::connection::ER_report_ESC_EEP = report_ESC_EEP

  report Escape Error End of Packet
- static const int esl::connection::ER_report_ESC_ESC = report_ESC_ESC

  report Escape Escape
- static const int esl::connection::ER_report_timeout = report_timeout

  report Timeout

5.13.1 Detailed Description

5.14 Error Waveform Triggers

Variables

- static const int **esl::connection::EW** = 9
- static const int **esl::connection::EW_RT** = 13
- static const int **esl::connection::EW_capture_nothing** = 0x00
- static const int esl::connection::EW_capture_first_null = (0x02 | report_first_null)

  trigger on first null
- static const int esl::connection::EW_capture_first_fct = 0x04

  trigger on first fct
- static const int esl::connection::EW_capture_running_error = (0x08 | report_parity_error | report_ESC_EOP | report_ESC_EEP | report_ESC_ESC | report_timeout)

  trigger on run error
- static const int esl::connection::EW_capture_starting_error = 0x10

  trigger on start error
- static const int esl::connection::EW_capture_nchar = (0x40 | report_nchar)

  trigger on n char
- static const int esl::connection::EW_capture_time_code = (0x80 | report_time_code)

  trigger on timecode
- static const int esl::connection::EW_capture_EOP = report_EOP

  trigger on End of Packet
- static const int esl::connection::EW_capture_EEP = report_EEP

  trigger on Error End of Packet
- static const int esl::connection::EW_capture_FCT = report_FCT

  trigger on FCT
- static const int esl::connection::EW_capture_excess_FCT = report_excess_FCT

  trigger on excess fct
- static const int esl::connection::EW_capture_excess_data = report_excess_data

  trigger on excess data
- static const int esl::connection::EW_capture_null = report_NULL

  trigger on NULL
- static const int esl::connection::EW_capture_parity_error = report_parity_error

  trigger on parity error
- static const int esl::connection::EW_capture_ESC_EOP = report_ESC_EOP

  trigger on Escape End of Packet
- static const int esl::connection::EW_capture_ESC_EEP = report_ESC_EEP

  trigger on Escape Error End of Packet
- static const int esl::connection::EW_capture_ESC_ESC = report_ESC_ESC

  trigger on Escape Escape
- static const int esl::connection::EW_capture_timeout = report_timeout

  trigger on timeout

5.14.1 Detailed Description

5.15 Error Waveform Sources

Variables

- static const int esl::connection::EW_Source_barrier = 0x0001

  Barrier.
- static const int esl::connection::EW_Source_port_1 = 0x0002

  Port 1.
- static const int esl::connection::EW_Source_port_2 = 0x0004

  Port 2.
- static const int esl::connection::EW_Source_port_3 = 0x0008

  Port 3.
- static const int esl::connection::EW_Source_port_4 = 0x0010

  Port 4.
- static const int esl::connection::EW_Source_port_5 = 0x0020

  Port 5.
- static const int esl::connection::EW_Source_port_6 = 0x0040

  Port 6.
- static const int esl::connection::EW_Source_port_7 = 0x0080

  Port 7.
- static const int esl::connection::EW_Source_port_8 = 0x0100

  Port 8.
- static const int esl::connection::EW_Source_SMA_12 = 0x0200

  SMA 1/2 changing state.
- static const int esl::connection::EW_Source_SMA_34 = 0x0400

  SMA 3/4 changing state.
- static const int esl::connection::EW_Source_SMA_56 = 0x0800

  SMA 5/6 changing state.
- static const int esl::connection::EW_Source_SMA_78 = 0x1000

  SMA 7/8 changing state.
- static const int esl::connection::EW_Source_local_clock = 0x8000

  Local clock.

5.15.1 Detailed Description

5.16 Memory Mapped Addresses

Variables

- static const int **esl::connection::LINK_address** = 0x0000
- static const int **esl::connection::TX_SPEED_address** = 0x87FD
- static const int **esl::connection::RX_SPEED_address** = 0x0001
- static const int **esl::connection::HWA_address** = 0x8800
- static const int **esl::connection::VERSION_address** = 0x880A
- static const int **esl::connection::DESCRIPTION_address** = 0x880B
- static const int **esl::connection::OPTIONS_address** = 0x8F60
- static const int **esl::connection::NLINKS_address** = 0x8FFF
- static const int **esl::connection::EW_address** = 0x1000
- static const int **esl::connection::PC_address** = 0x2000
- static const int **esl::connection::PG_address** = 0x4000
- static const int **esl::connection::ATI_address** = 0x0100
- static const int **esl::connection::OBSERVE_address** = 0x0020
- static const int **esl::connection::TIMETAG_address** = 0x0030
- static const int **esl::connection::IGNORE_address** = 0x0040
- static const int **esl::connection::Event_cause_address** = 0x0060
- static const int **esl::connection::EW_source_address** = 0x0070
- static const int **esl::connection::FLOW_CONTROL_address** = 0x0050
- static const int **esl::connection::SMA_56_pulse_width_address** = 0x00F0
- static const int **esl::connection::max_packet_data** = 0x0010

5.16.1 Detailed Description

Error codes which API calls may set and be retrieved by the get error call

5.17 Error Codes

Variables

- static const int esl::connection::Error_RecFile_Open = -1

  Couldn't open recording file.
- static const int esl::connection::Error_RecFile_Write = -2

  record_file write failed
- static const int esl::connection::Error_LogFile_Open = -3

  Couldn't open logging file.
- static const int esl::connection::Error_LogFile_Write = -4

  log_file write failed
- static const int esl::connection::Error_Receiver_Timeout = -10

  we have a network timeout timeout
- static const int esl::connection::Error_Receiver_Shutdown = -11

  peer has performed an orderly shutdown
- static const int esl::connection::Error_IO_Error = -12

  we have an IO error
- static const int esl::connection::Error_SaveBuf_Overflow_Save = -15

  Saving the read_packet_full() save_buffer failed.
- static const int esl::connection::Error_SaveBuf_Overflow_Restore = -16

  Restoring the read_packet_full() save_buffer failed.
- static const int esl::connection::Error_Function_Not_Supported = -17

  Device does not support the requested function.
- static const int esl::connection::Error_Network = -18

  Error reading / writing to/from the device.
- static const int esl::connection::Error_Network_Format_Error = -19

  Error understanding recieved packet.
- static const int esl::connection::Error_Request_Too_Large = -20

  The I/O request can't be fullfilled by the hardware.
- static const int esl::connection::Error_Sequence_Error = -21

  Didn't receive expected notification from the hardware.
- static const int esl::connection::Error_Response_Too_Small = -22

  Response from the device didn't contain enough data.
- static const int esl::connection::Error_Response_Mismatch = -23

  Response does not match I/O request.
- static const int esl::connection::Error_Module_Not_Present = -24

  Module not present.
- static const int esl::connection::Error_Parameter_RangeIncorrect = -25

  Parameter not in range.
- static const int esl::connection::Error_File_Not_Present = -26

  Requested file is not present.
- static const int esl::connection::Error_EINTR = -27

  EINTR occurred.
- static const int esl::connection::Error_Link_Incorrect = -28

  Link number is incorrect.
- static const int esl::connection::Error_Incorrect_Device = -29

  Connecting to a device which does not support functionality.
- static const int esl::connection::Error_Memory = -30

  Unable to allocate memory.
- static const int esl::connection::Error_Host_Unresolvable = -31

Unable to resolve host.

- static const int esl::connection::Error_Host_Unresponsive = -32

  Unable to connect to host.
- static const int esl::connection::Error_WaveForm_Dir_Create = -33

  Unable to create waveform directory.
- static const int esl::connection::Error_Zero_Read = -34

  asked to read zero bytes
- static const int esl::connection::Error_Set_Option_File = -35

  Asked to set an option when playing back from file.
- static const int esl::connection::Error_Invalid_Device = -36

  Device is not supported by API.
- static const int esl::connection::Error_File_Move = -37

  Unable to move file into place.
- static const int esl::connection::Error_Invalid_File = -38

  Unable to open file.
- static const int esl::connection::Error_Callback_Return = -39

  Callback has asked for a return.
- static const int esl::connection::Error_FileList_Empty = -40

  List of files given is empty.
- static const int esl::connection::Error_Unknown_System_Type = -41

  Unknown type.
- static const int esl::connection::Error_Not_Known = -42

  API returned 0 as an error should (should not happen)
- static const int esl::connection::Error_EXE_Start_Failed = -43

  Cannot start executable.
- static const int esl::connection::Error_NO_Connection = -44

  Link Not estblished.
- static const int esl::connection::Error_Invalid_Link = -45

  Invalid Link selected.
- static const int esl::connection::Error_Would_Block = -48

  I/O call would block.
- static const int esl::connection::Error_Link_Not_Connected = -49

  Link Not Connected.
- static const int esl::connection::Error_ReadHandler_Running = -50

  There is a read handler running for this connection.
- static const int esl::connection::Error_Buffer_Full = -51

  can't do non blocking write as buffer is full
- static const int esl::connection::Error_CaptureThread_Failed = -52

  Capture thread failed.
- static const int esl::connection::Option_SO = 1

  Option SO module is installed.

5.17.1 Detailed Description

Error codes which API calls may set and be retrieved by the get error call

5.18 functions

Functions

- virtual void esl::connection::reported_version (uint8_t version_, uint8_t type_)
- virtual void esl::connection::startdate (time_t lnow_, uint32_t ns_)
- virtual bool esl::connection::timetag_uncertainty (uint32_t uncertainty_)
- virtual bool esl::connection::missing_data ()
- virtual bool esl::connection::esc_eep ()
- virtual bool esl::connection::received_fct ()
- virtual bool esl::connection::received_esc_fct ()
- virtual bool esl::connection::esc_eop ()
- virtual bool esl::connection::esc_esc ()
- virtual void esl::connection::link_selected (int link_)
- virtual bool esl::connection::link_timeout ()
- virtual bool esl::connection::port_status (int link_, double rxspeed_, bool connected_, int runstatnus_)
- virtual bool esl::connection::link_tx_speed (int link_, double txspeed_)
- virtual bool esl::connection::parity_error ()
- virtual bool esl::connection::error_event ()
- virtual bool esl::connection::perror1 ()
- virtual bool esl::connection::perror2 ()
- virtual bool esl::connection::timecode (uint8_t tc_, uint8_t a_, uint8_t b_, uint8_t t_)
- virtual void esl::connection::timetag (uint64_t time_)

  time code
- virtual void esl::connection::timezero (double time_)

  time tag
- virtual void esl::connection::err (double time_, int state_, int error_bits_)

  when recording started
- virtual bool esl::connection::barrier_lifted (uint64_t t_)
- virtual bool esl::connection::waveform_data (int erridx, int unit, int port, int ew_port, int event, char *cause, unsigned char *data_, size_t sz)
- virtual bool esl::connection::unknown_ram_data (void *data_, int length_, int complete_, int data_buffer_position_)
- virtual bool esl::connection::unknown_special_data (void *data_, int length_, int complete_, int data_buffer_position_)
- virtual bool esl::connection::unknown_extn_data (void *data_, int length_, int complete_, int data_buffer_position_)
- std::string **esl::connection::address** () const

5.18.1 Detailed Description

These functions are called when an event on a spacewire link occurs

5.18.2 Function Documentation

5.18.2.1 virtual bool esl::connection::barrier_lifted (uint64_t *t_*) [inline], [virtual]

callback informing the application a barrier has lifted

Parameters

t	the time the barrier was raised

5.18.2.2 virtual void esl::connection::err (double *time_*, int *state_*, int *error_bits_*) [inline], [virtual]

when recording started

callback informing the application of an error on the currently active rx link

Parameters

*time_*	the current time
*state_*	the state of the port
*error_bits_*	mask of error bits

5.18.2.3 virtual bool esl::connection::error_event ( ) [inline], [virtual]

callback informing the application that the currently active port has encountered an error with error waveform data available

Returns

false to return to calling read

5.18.2.4 virtual bool esl::connection::esc_eep ( ) [inline], [virtual]

callback informing the application the device has seen an escape eep frame

Returns

false to return to calling read

5.18.2.5 virtual bool esl::connection::esc_eop ( ) [inline], [virtual]

callback informing the application the device has seen an escape eop frame

Returns

false to return to calling read

5.18.2.6 virtual bool esl::connection::esc_esc ( ) [inline], [virtual]

callback informing the application the device has seen an escape escape frame

Returns

false to return to calling read

5.18.2.7 virtual void esl::connection::link_selected (int *link_*) [inline], [virtual]

callback when the rx link has changed

Parameters

*link_*	the newly active rx link

5.18.2.8 virtual bool esl::connection::link_timeout ( ) [inline], [virtual]

callback informing the application that the currently rx link has timedout (disconnected?)

Returns

false to return to calling read

5.18.2.9 virtual bool esl::connection::link_tx_speed (int *link_*, double *txspeed_*) [inline], [virtual]

callback informing the application that a port has changed TX speed

Parameters

*link_*	the link to which this pertains
*txspeed_*	the TX speed

Returns

false to return to calling read

5.18.2.10 virtual bool esl::connection::missing_data ( ) [inline], [virtual]

callback informing the application that the device has missed data

Returns

false to return to calling read

5.18.2.11 virtual bool esl::connection::parity_error ( ) [inline], [virtual]

callback informing the application that the currently active link has had a parity error

Returns

false to return to calling read

5.18.2.12 virtual bool esl::connection::perror1 ( ) [inline], [virtual]

callback informing the application that the currently active port has had an error

Returns

false to return to calling read

5.18.2.13 virtual bool esl::connection::perror2 ( ) [inline], [virtual]

callback informing the application that the currently active port has had an error

Returns

false to return to calling read

5.18.2.14 virtual bool esl::connection::port_status (int *link_*, double *rxspeed_*, bool *connected_*, int *runstatnus_*) [inline], [virtual]

callback informing the application that a port has changed status

Parameters

*link_*	the link to which this pertains
*rxspeed_*	the RX speed
*connected_*	is the link connected
runstatus	the raw status

Returns

false to return to calling read

5.18.2.15 virtual bool esl::connection::received_esc_fct ( ) [inline], [virtual]

callback informing the application the device has seen a ESC FCT (aka NULL)

Returns

false to return to calling read

5.18.2.16 virtual bool esl::connection::received_fct ( ) [inline], [virtual]

callback informing the application the device has seen a FCT

Returns

false to return to calling read

5.18.2.17 virtual void esl::connection::reported_version (uint8_t *version_*, uint8_t *type_*) [inline], [virtual]

callback when reading the start date from a capture file

Parameters

*version_*	the version of the recording file
*type_*	the syustem type of the device

5.18.2.18 virtual void esl::connection::startdate (time_t *lnow_*, uint32_t *ns_*) [inline], [virtual]

callback when reading the start date from a capture file

Parameters

*lnow_*	the time when recording the capture file started
*ns_*	the nano second offset

5.18.2.19 virtual bool esl::connection::timecode (uint8_t *tc_*, uint8_t *a_*, uint8_t *b_*, uint8_t *t_*) [inline], [virtual]

callback informing the application of a spacewire timecode

Parameters

tc_,the	full spacewire timecode
a_,a	value for timecode
b_,b	value for timecode
t_,current	timecode

Returns

false to return to calling read

5.18.2.20 virtual void esl::connection::timetag (uint64_t *time_*) [inline], [virtual]

time code

callback informing the application of the currently active rx time

Parameters

*time_*	the current time, the number of 1/10ths of nano seconds since the beginning of the year if synchronised with GPS otherwise the number of 1/10ths since power on.

5.18.2.21 virtual bool esl::connection::timetag_uncertainty (uint32_t *uncertainty_*) [inline], [virtual]

callback informing the application of the time tag uncertainty

Parameters

*uncertainty_*	the current uncertainty

Returns

false to return to calling read

5.18.2.22 virtual void esl::connection::timezero (double *time_*) [inline], [virtual]

time tag

callback informing the application of the first timetag

Parameters

*time_*	the current time, the number of 1/10ths of nano seconds since the beginning of the year if synchronised with GPS otherwise the number of 1/10ths since power on.

5.18.2.23 virtual bool esl::connection::unknown_extn_data (void * *data_*, int *length_*, int *complete_*, int *data_buffer_position_*) [inline], [virtual]

callback informing the application of unknown extension data

Parameters

*data_*	the data from ram i/o module

*length_*	the length of the data
*complete_*	is the data complete (if not more to come)
*data_buffer_position_*	the current position in the stream of data

5.18.2.24 virtual bool esl::connection::unknown_ram_data (void * *data_*, int *length_*, int *complete_*, int *data_buffer_position_*) [inline],[virtual]

callback informing the application of data from an unknown module (should never trigger)

Parameters

*data_*	the data from ram i/o module
*length_*	the length of the data
*complete_*	is the data complete (if not more to come)
*data_buffer_position_*	the current position in the stream of data

Returns

true to return to calling read

5.18.2.25 virtual bool esl::connection::unknown_special_data (void * *data_*, int *length_*, int *complete_*, int *data_buffer_position_*) [inline],[virtual]

callback informing the application of unknown special data

Parameters

*data_*	the data from ram i/o module
*length_*	the length of the data
*complete_*	is the data complete (if not more to come)
*data_buffer_position_*	the current position in the stream of data

Returns

true to return to calling read

5.18.2.26 virtual bool esl::connection::waveform_data (int *erridx*, int *unit*, int *port*, int *ew_port*, int *event*, char * *cause*, unsigned char * *data_*, size_t *sz*) [inline],[virtual]

callback informing the application of an error waveform data See the waveform class for decoding the daw data

Parameters

erridx	the error index
unit	the unit
port	the port the waveform is appliocable to
ew_port	the triggering port
cause	text description of error cause

*data_*	raw waveform data
sz	the size of the above data

Chapter 6

Namespace Documentation

6.1 esl Namespace Reference

The esl namespace is used for EtherSpaceLink hardware.

Classes

- class error

  Exception class, whenever an error occurrs an exception is thrown, this makes error path handling somewhat easier.
- class stats
- class portstatus
- class devstatus
- struct timestamp

  connection this class provides the base class of communicating with a DSI/MSR
- class connection

6.1.1 Detailed Description

The esl namespace is used for EtherSpaceLink hardware.

Chapter 7

Class Documentation

7.1 esl::connection Class Reference

Public Member Functions

- virtual void reported_version (uint8_t version_, uint8_t type_)
- virtual void startdate (time_t lnow_, uint32_t ns_)
- virtual bool timetag_uncertainty (uint32_t uncertainty_)
- virtual bool missing_data ()
- virtual bool esc_eep ()
- virtual bool received_fct ()
- virtual bool received_esc_fct ()
- virtual bool esc_eop ()
- virtual bool esc_esc ()
- virtual void link_selected (int link_)
- virtual bool link_timeout ()
- virtual bool port_status (int link_, double rxspeed_, bool connected_, int runstatnus_)
- virtual bool link_tx_speed (int link_, double txspeed_)
- virtual bool parity_error ()
- virtual bool error_event ()
- virtual bool perror1 ()
- virtual bool perror2 ()
- virtual bool timecode (uint8_t tc_, uint8_t a_, uint8_t b_, uint8_t t_)
- virtual void timetag (uint64_t time_)

  time code
- virtual void timezero (double time_)

  time tag
- virtual void err (double time_, int state_, int error_bits_)

  when recording started
- virtual bool barrier_lifted (uint64_t t_)
- virtual bool waveform_data (int erridx, int unit, int port, int ew_port, int event, char ∗cause, unsigned char ∗data_, size_t sz)
- virtual bool unknown_ram_data (void ∗data_, int length_, int complete_, int data_buffer_position_)
- virtual bool unknown_special_data (void ∗data_, int length_, int complete_, int data_buffer_position_)
- virtual bool unknown_extn_data (void ∗data_, int length_, int complete_, int data_buffer_position_)
- std::string **address** () const
- int get_link (void)
- void set_link (int link_)
- void **reported_startdate** (time_t t_, uint32_t ns_)

- void **reported_time** (uint64_t t_, bool tt_)
- std::string version ()
- virtual ~connection ()
- **connection** (const std::queue< std::string > &filelist_) throw (error)
- connection (const std::string &address_, bool file_=false) throw (error)
- void abort ()
- void flush () throw (error)
- ssize_t read (char *data_, size_t sz_) throw (error)
- ssize_t complex_read (char *data_, size_t sz_, uint32_t flags_) throw (error)
- int **empty_frame** (ssize_t sz_)
- int read_info ()
- void write (const char *data_, size_t sz_, uint32_t flags_) throw (error)
- virtual void active_port (int link_) throw (error)
- virtual void mode (int mode_) throw (error)
- void modes (uint32_t port_, int mode_) throw (error)
- void **fastclose** () throw (error)
- void speed (double speed_) throw (error)
- void speed (int speed_) throw (error)
- int rx_speed () throw (error)
- std::string manufacturer () throw (error)
- std::string options () throw (error)
- std::string product () throw (error)
- unsigned char * mac () throw (error)
- std::string serial () throw (error)
- void request_link_status () throw (error)
- void request_link_status_port (int link_) throw (error)
- void request_tx_speed () throw (error)
- bool link_connected () throw (error)
- void **connect_link** (int link_, int to_) throw (error)
- int nolinks () throw (error)
- void tx_record (const char *file_) throw (error)
- void record (const char *file_, bool recordwrites_, bool writeerror_=false) throw (error)
- void record_flush () throw (error)
- uint64_t record_size () throw (error)
- FILE * record_file ()
- void log (const std::string &file_) throw (error)
- ::EtherSpaceLink raw ()
- void eintr (bool return_)
- void non_blocking (bool nonblock_) throw (error)
- void set_rx_timeout (uint32_t timeout_) throw (error)
- uint32_t get_rx_timeout () throw (error)
- double percent_read ()
- int module (int module_)
- int slot (int slot_)
- SKT socket ()
- virtual void TT_reporting (int when_) throw (error)
- virtual void ER_reporting (int when_) throw (error)
- virtual void EW_reporting (int when_) throw (error)
- virtual void EW_source (int source_) throw (error)
- void EW_request (int link_) throw (error)
- void EW_clear (int link_) throw (error)
- void EW_reset (int link_) throw (error)
- void dump_limit (int limit_)
- void dump (const std::string &file_, const std::string &prefix_)
- void dump_state ()

- void EI_ignore_events (int what_) throw (error)
- void EI_flow_control (int initial_fct_, int fct_) throw (error)
- bool write_buffer_empty () throw (error)
- void send_timecode (uint8_t tc_) throw (error)
- void set_max_packet_data (int sz_) throw (error)
- void sma_56_pulse_width (int width_) throw (error)
- void ei_parity_error () throw (error)
- void ei_esc_eop () throw (error)
- void ei_esc_eep () throw (error)
- void ei_esc_esc () throw (error)
- void ei_esc_fct () throw (error)
- void **ATI_calibrate** (int v_) throw (error)
- void **Observe** (int v_) throw (error)
- uint64_t raw_bytes ()
- void **io_stats** (uint64_t &rxtotal_, uint64_t &calls, uint64_t &iocalls)
- void **timeout_errors** (bool errors_=true)
- double current_time () const

  Function to obtain the curent status of the run.
- devstatus * **stats** ()
- void **metadata** (const connection *const con_)
- void sendfile (const char *filename_) throw (error)
- void **packet_time** (time_t &s_, snseconds_t &ns_)
- void **set_system_type** (int st_)

Static Public Attributes

- static const int PART_EOP_EEP = 1000

  Error packet.
- static const int SPECIAL = 1003

  We are sending a special frame.
- static const int EXTN = 1005

  We are sending an extension frame.
- static const int **PART_EXTN** = 1006
- static const int INCOMPLETE = 1008

  Used to build up a single packet for the unit.
- static const int **FLUSH** = 2048
- static const int PART_SPECIAL = 1004

  A part of a special frame.
- static const int TRUNCATED = 1007

  Aritifical construct for unhandled data.
- static const int SPECIAL_SIZE = 1009

  Returning the amount of special data.
- static const int EXTENSION_SIZE = 1010

  Returning the amount of extension data.
- static const int **FCT** = 0x100
- static const int EEP = 0x101

  Error End of Packet.
- static const int EOP = 0x102

  End of Packet.
- static const int ESC = 0x103

  Escape.
- static const int ESC_FCT = 0x104

Escape FCT aka a NULL character.

- static const int ESC_EEP = 0x105

  Escape End of Packet.
- static const int ESC_EOP = 0x106

  Escape Error of packet.
- static const int ESC_ESC = 0x107

  Escape Escape.
- static const int Timeout = 0x108

  Timeout message.
- static const int ParityError = 0x109

  Parity Error message.
- static const int PERROR1 = 0x10A

  Error 1 message.
- static const int PERROR2 = 0x10B

  Error 2 message.
- static const int **STORE** = 0x10C
- static const int **FORWARD** = 0x10D
- static const int **ATOM** = 0x10E
- static const int **MOTA** = 0x10F
- static const int **JOIN** = 0x110
- static const int **BARRIER** = 0x111
- static const int **RESIGN** = 0x112
- static const int **EVENT** = 0x113
- static const int Missing_data = 0x114

  Missed data message.
- static const int **HOLD** = 0x12F
- static const int **Delay** = 0x130
- static const int PortSelect = 0x140

  Port select message.
- static const int PortSelect_max = 0x17F

  Max port select message.
- static const int **Multi_byte_extn_start** = 0x180
- static const int TimeTag = 0x188

  Timetag message.
- static const int TimeTag_delta = 0x182

  Timetag delta message.
- static const int TimeTag_uncertainty = 0x181

  Timetag uncertain message.
- static const int TimeCode = 0x191

  Spacewire timecode.
- static const int Module = 0x192

  Module data.
- static const int TimeZero = 0x198

  First timecode on the link.
- static const int **TRUNCATE_1** = 0x1A1
- static const int **TRUNCATE_2** = 0x1A2
- static const int **REPEAT_1** = 0x1B1
- static const int **REPEAT_2** = 0x1B2
- static const int **REPEAT_3** = 0x1B3
- static const int Year = 0x1C8

  Capture start date/time.
- static const int Header = 0x1CE

Capture Header containing version and time information.

- static const int report_EEP = 0x800000

  EEP error event.
- static const int report_nchar = 0x400000

  character received event
- static const int report_first_null = 0x200000

  First null event.
- static const int report_excess_FCT = 0x100000

  Too many FCTS event.
- static const int report_excess_data = 0x080000

  Too much data sent for # of FCT's.
- static const int report_first_byte = 0x040000

  First byte of packet.
- static const int report_mid_bytes = 0x020000

  Frame mide byte.
- static const int report_EOP = 0x010000

  EOP recieved.
- static const int report_time_code = 0x008000

  Time code received.
- static const int report_FCT = 0x004000

  FCT received.
- static const int report_NULL = 0x002000

  Null received.
- static const int report_parity_error = 0x001000

  Parity Error.
- static const int report_ESC_EOP = 0x000800

  Escape EOP error.
- static const int report_ESC_EEP = 0x000400

  Escape EEP Error.
- static const int report_ESC_ESC = 0x000200

  Escape Escape Error.
- static const int report_timeout = 0x000100

  Link Timeout.
- static const int **report_delta** = 0x400000
- static const int **DISCARD_SPECIAL_DATA** = 0x00
- static const int **REPORT_SPECIAL_DATA** = 0x01
- static const int **RETURN_SPECIAL_DATA** = 0x02
- static const int **CALLBACK_SPECIAL_DATA** = 0x03
- static const int **SPECIAL_DATA_FLAGS** = 0x03
- static const int **DISCARD_EXTENSION_DATA** = 0x00
- static const int **REPORT_EXTENSION_DATA** = 0x10
- static const int **RETURN_EXTENSION_DATA** = 0x20
- static const int **CALLBACK_EXTENSION_DATA** = 0x30
- static const int **EXTENSION_DATA_FLAGS** = 0x30
- static const int **READ_IMMEDIATE** = 0x40
- static const int **SpaceWire_state_ErrorReset** = 0
- static const int **SpaceWire_state_ErrorWait** = 1
- static const int **SpaceWire_state_Ready** = 2
- static const int **SpaceWire_state_Started** = 3
- static const int **SpaceWire_state_Connecting** = 4
- static const int **SpaceWire_state_Run** = 5
- static const int **MSR_state_NC** = 6

- static const int **MSR_state_Connected** = 7
- static const int **CAPABILITIES** = 0
- static const int **HWA** = 3
- static const int **LINK_SPEED** = 4
- static const int **MANUFACTURER** = 1
- static const int **PRODUCT** = 2
- static const int **LINK** = 5
- static const int LINK_mode_disabled = 0x01

  Disables the link.
- static const int LINK_mode_normal = 0x02

  Enables the link.
- static const int LINK_mode_legacy = 0x04

  IEEE 1355 (spacewire precursor)
- static const int LINK_mode_master = 0x06

  IEEE 1355 (precurosr)
- static const int **LINK_tx_buffer_empty** = 0x08
- static const int **LINK_state_offset** = 4
- static const int **LINK_mode_long_timeout** = 0x40
- static const int **LINK_mode_fixed_speed** = 0x80
- static const int **LINK_mode_slow_speed** = 0xC0
- static const int **SF** = 6
- static const int **SF_disabled** = 0x00
- static const int **SF_enabled** = 0x80
- static const int **TT** = 7
- static const int **TT_64** = 15
- static const int TT_report_nothing = 0x00

  Report Nothing.
- static const int TT_report_first_byte = (0x01 | report_first_byte)

  Timetag first byte of packet.
- static const int TT_report_intermediate_bytes = (0x02 | report_mid_bytes)

  Timetag middle byte.
- static const int TT_report_EOP_EEP = (0x04 | report_EEP | report_EOP)

  Timetag end of packet markers.
- static const int TT_report_EEP = (0x04 | report_EEP)

  Timetag report Error End of Packet.
- static const int TT_report_EOP = (0x04 | report_EOP)

  Timetag report End of Packet.
- static const int TT_report_time_code = report_time_code

  Timetag report spacewire timecode.
- static const int TT_report_fct = report_FCT

  Timetag report FCT.
- static const int TT_report_null = report_NULL

  Timetag report NULL.
- static const int TT_report_parity_error = report_parity_error

  Timetag report parity error.
- static const int TT_report_ESC_EOP = report_ESC_EOP

  Timetag report ESC End of Packet.
- static const int TT_report_ESC_EEP = report_ESC_EEP

  Timetag report ESC Error End of Packet.
- static const int TT_report_ESC_ESC = report_ESC_ESC

  Timetag report ESC ESC.
- static const int TT_report_timeout = report_timeout

Timetag report timeout.

- static const int **ER** = 8
- static const int **ER_64** = 16
- static const int ER_report_nothing = 0x00

  Error reporting report nothing.
- static const int ER_report_first_null = 0x02

  Error report first null.
- static const int ER_report_first_fct = 0x04

  Error report first fct.
- static const int ER_report_running_error = (0x08 | report_parity_error | report_ESC_EOP | report_ESC_EEP | report_ESC_ESC | report_timeout)

  Error report running.
- static const int **ER_report_starting_error** = 0x10
- static const int **ER_report_nchar** = 0x40
- static const int ER_report_time_code = (0x80 | report_time_code)

  report time code
- static const int ER_report_fct = report_FCT

  report FCT
- static const int ER_report_null = report_NULL

  report null
- static const int ER_report_parity_error = report_parity_error

  report parity error
- static const int ER_report_ESC_EOP = report_ESC_EOP

  report Escape End of Packet
- static const int ER_report_ESC_EEP = report_ESC_EEP

  report Escape Error End of Packet
- static const int ER_report_ESC_ESC = report_ESC_ESC

  report Escape Escape
- static const int ER_report_timeout = report_timeout

  report Timeout
- static const int **EW** = 9
- static const int **EW_RT** = 13
- static const int **EW_capture_nothing** = 0x00
- static const int EW_capture_first_null = (0x02 | report_first_null)

  trigger on first null
- static const int EW_capture_first_fct = 0x04

  trigger on first fct
- static const int EW_capture_running_error = (0x08 | report_parity_error | report_ESC_EOP | report_ESC_EEP | report_ESC_ESC | report_timeout)

  trigger on run error
- static const int EW_capture_starting_error = 0x10

  trigger on start error
- static const int EW_capture_nchar = (0x40 | report_nchar)

  trigger on n char
- static const int EW_capture_time_code = (0x80 | report_time_code)

  trigger on timecode
- static const int EW_capture_EOP = report_EOP

  trigger on End of Packet
- static const int EW_capture_EEP = report_EEP

  trigger on Error End of Packet
- static const int EW_capture_FCT = report_FCT

trigger on FCT

- static const int EW_capture_excess_FCT = report_excess_FCT

  trigger on excess fct
- static const int EW_capture_excess_data = report_excess_data

  trigger on excess data
- static const int EW_capture_null = report_NULL

  trigger on NULL
- static const int EW_capture_parity_error = report_parity_error

  trigger on parity error
- static const int EW_capture_ESC_EOP = report_ESC_EOP

  trigger on Escape End of Packet
- static const int EW_capture_ESC_EEP = report_ESC_EEP

  trigger on Escape Error End of Packet
- static const int EW_capture_ESC_ESC = report_ESC_ESC

  trigger on Escape Escape
- static const int EW_capture_timeout = report_timeout

  trigger on timeout
- static const int EW_Source_barrier = 0x0001

  Barrier.
- static const int EW_Source_port_1 = 0x0002

  Port 1.
- static const int EW_Source_port_2 = 0x0004

  Port 2.
- static const int EW_Source_port_3 = 0x0008

  Port 3.
- static const int EW_Source_port_4 = 0x0010

  Port 4.
- static const int EW_Source_port_5 = 0x0020

  Port 5.
- static const int EW_Source_port_6 = 0x0040

  Port 6.
- static const int EW_Source_port_7 = 0x0080

  Port 7.
- static const int EW_Source_port_8 = 0x0100

  Port 8.
- static const int EW_Source_SMA_12 = 0x0200

  SMA 1/2 changing state.
- static const int EW_Source_SMA_34 = 0x0400

  SMA 3/4 changing state.
- static const int EW_Source_SMA_56 = 0x0800

  SMA 5/6 changing state.
- static const int EW_Source_SMA_78 = 0x1000

  SMA 7/8 changing state.
- static const int EW_Source_local_clock = 0x8000

  Local clock.
- static const int **TC_rx** = 10
- static const int **TC_rx_64** = 17
- static const int **TC_rx_silent** = 0x00
- static const int **TC_rx_report_enabled** = 0x08
- static const int **TC_rx_time_stamp_enabled** = 0x40
- static const int **TC_tx** = 11

- static const int **TC_tx_trigger_mask** = 0x03
- static const int **TC_tx_no_trigger** = 0x00
- static const int **TC_tx_one_code** = 0x01
- static const int **TC_tx_external_trigger** = 0x02
- static const int **TC_tx_regular_trigger** = 0x03
- static const int **TC_tx_update_interval** = 0x04
- static const int **TC_tx_update_code** = 0x08
- static const int **TC_tx_format_mask** = 0x30
- static const int **TC_tx_no_increment** = 0x00
- static const int **TC_tx_increment_6_bits** = 0x10
- static const int **TC_tx_increment_7_bits** = 0x20
- static const int **TC_tx_increment_8_bits** = 0x30
- static const int **TC_tx_report_transmission** = 0x40
- static const int **CR** = 14
- static const int **router_cs** = 18
- static const int **router_tables** = 19
- static const int **router_stats** = 20
- static const int **ram_rw** = 21
- static const int **barrier** = 22
- static const int **TT_now** = 23
- static const int **EI_ignore_excess_FCT** = report_excess_FCT
- static const int **EI_ignore_excess_data** = report_excess_data
- static const int **EI_ignore_parity_error** = report_parity_error
- static const int **EI_ignore_ESC_EOP** = report_ESC_EOP
- static const int **EI_ignore_ESC_EEP** = report_ESC_EEP
- static const int **EI_ignore_ESC_ESC** = report_ESC_ESC
- static const int **EI_ignore_timeout** = report_timeout
- static const int **EI_normal_flow_control** = 0x00
- static const int **EI_transmit_anyway** = 0x20
- static const int **EI_no_automatic_FCT** = 0x10
- static const int **LINK_address** = 0x0000
- static const int **TX_SPEED_address** = 0x87FD
- static const int **RX_SPEED_address** = 0x0001
- static const int **HWA_address** = 0x8800
- static const int **VERSION_address** = 0x880A
- static const int **DESCRIPTION_address** = 0x880B
- static const int **OPTIONS_address** = 0x8F60
- static const int **NLINKS_address** = 0x8FFF
- static const int **EW_address** = 0x1000
- static const int **PC_address** = 0x2000
- static const int **PG_address** = 0x4000
- static const int **ATI_address** = 0x0100
- static const int **OBSERVE_address** = 0x0020
- static const int **TIMETAG_address** = 0x0030
- static const int **IGNORE_address** = 0x0040
- static const int **Event_cause_address** = 0x0060
- static const int **EW_source_address** = 0x0070
- static const int **FLOW_CONTROL_address** = 0x0050
- static const int **SMA_56_pulse_width_address** = 0x00F0
- static const int **max_packet_data** = 0x0010
- static const int Error_RecFile_Open = -1

  Couldn't open recording file.
- static const int Error_RecFile_Write = -2

  record_file write failed

- static const int Error_LogFile_Open = -3

  Couldn't open logging file.
- static const int Error_LogFile_Write = -4

  log_file write failed
- static const int Error_Receiver_Timeout = -10

  we have a network timeout timeout
- static const int Error_Receiver_Shutdown = -11

  peer has performed an orderly shutdown
- static const int Error_IO_Error = -12

  we have an IO error
- static const int Error_SaveBuf_Overflow_Save = -15

  Saving the read_packet_full() save_buffer failed.
- static const int Error_SaveBuf_Overflow_Restore = -16

  Restoring the read_packet_full() save_buffer failed.
- static const int Error_Function_Not_Supported = -17

  Device does not support the requested function.
- static const int Error_Network = -18

  Error reading / writing to/from the device.
- static const int Error_Network_Format_Error = -19

  Error understanding recieved packet.
- static const int Error_Request_Too_Large = -20

  The I/O request can't be fullfilled by the hardware.
- static const int Error_Sequence_Error = -21

  Didn't receive expected notification from the hardware.
- static const int Error_Response_Too_Small = -22

  Response from the device didn't contain enough data.
- static const int Error_Response_Mismatch = -23

  Response does not match I/O request.
- static const int Error_Module_Not_Present = -24

  Module not present.
- static const int Error_Parameter_RangeIncorrect = -25

  Parameter not in range.
- static const int Error_File_Not_Present = -26

  Requested file is not present.
- static const int Error_EINTR = -27

  EINTR occurred.
- static const int Error_Link_Incorrect = -28

  Link number is incorrect.
- static const int Error_Incorrect_Device = -29

  Connecting to a device which does not support functionality.
- static const int Error_Memory = -30

  Unable to allocate memory.
- static const int Error_Host_Unresolvable = -31

  Unable to resolve host.
- static const int Error_Host_Unresponsive = -32

  Unable to connect to host.
- static const int Error_WaveForm_Dir_Create = -33

  Unable to create waveform directory.
- static const int Error_Zero_Read = -34

  asked to read zero bytes
- static const int Error_Set_Option_File = -35

Asked to set an option when playing back from file.

- static const int Error_Invalid_Device = -36

  Device is not supported by API.

- static const int Error_File_Move = -37

  Unable to move file into place.

- static const int Error_Invalid_File = -38

  Unable to open file.

- static const int Error_Callback_Return = -39

  Callback has asked for a return.

- static const int Error_FileList_Empty = -40

  List of files given is empty.

- static const int Error_Unknown_System_Type = -41

  Unknown type.

- static const int Error_Not_Known = -42

  API returned 0 as an error should (should not happen)

- static const int Error_EXE_Start_Failed = -43

  Cannot start executable.

- static const int Error_NO_Connection = -44

  Link Not estblished.

- static const int Error_Invalid_Link = -45

  Invalid Link selected.

- static const int Error_Would_Block = -48

  I/O call would block.

- static const int Error_Link_Not_Connected = -49

  Link Not Connected.

- static const int Error_ReadHandler_Running = -50

  There is a read handler running for this connection.

- static const int Error_Buffer_Full = -51

  can't do non blocking write as buffer is full

- static const int Error_CaptureThread_Failed = -52

  Capture thread failed.

- static const int Option_SO = 1

  Option SO module is installed.

- static const int **CONNECT_FILE** = (1)
- static const int **Receiver_Timeout_Returns_Zero_Part_Pkt** = 0
- static const int **Receiver_Timeout_Returns_Error** = 1
- static const int **SYSTEM_TYPE_INVALID** = 0
- static const int **SYSTEM_TYPE_401** = 1
- static const int **SYSTEM_TYPE_408** = 2

Protected Member Functions

- bool **raw_link_data** (unsigned char ∗special_, size_t length_)
- bool **raw_speed_data** (unsigned char ∗special_, size_t length_)
- virtual int **ram_rw_data** (void ∗data_, int length_, int complete_, int data_buffer_position_)
- virtual int **special_data** (void ∗data_, int length_, int complete_, int data_buffer_position_)
- virtual int **extn_data** (void ∗data_, int length_, int complete_, int data_buffer_position_)
- void **initialise_con** ()

Static Protected Member Functions

- static int **special_data_s** (EtherSpaceLink lnk_, void *data_, int length_, int complete_, int data_buffer_position_)
- static int **extn_data_s** (EtherSpaceLink lnk_, void *data_, int length_, int complete_, int data_buffer_position_)
- static void **error_handler_s** (EtherSpaceLink lnk_, void *data_, int err_, const char *desc_, const char *location_)

Protected Attributes

- int tx_link

  Current active tx link.
- int rx_link

  currently active rx link
- uint32_t read_flags

  flags returned by last read
- uint32_t uncertainty

  Current uncertainty.
- time_t yeartime

  Year time.
- time_t starttime

  Start.
- bool t0start

  Do we have a time.
- timestamp timestamps [32]

  Time stamp.
- devstatus status

  Status of what is going on.
- bool running

  Are we running.
- struct eslhires baseclock

  Current base clock.
- bool live

  is the connection live
- std::string addr

  Address.
- EtherSpaceLink device

  spacelink device

7.1.1 Constructor & Destructor Documentation

7.1.1.1 virtual esl::connection::~connection ( ) [inline], [virtual]

class destructor

Returns

the API version

7.1.1.2 esl::connection::connection (const std::string & *address_*, bool *file_* = false) throw error) [inline]

connection constructor

Parameters

*address_*	of the device or the filename to replay
*file_*	are we reading from a file ?

7.1.2 Member Function Documentation

7.1.2.1 void esl::connection::abort ( ) [inline]

abort connection, generally used to knock off rx thread on a read

7.1.2.2 virtual void esl::connection::active_port (int *link_*) throw error) [inline], [virtual]

sets the active port

Parameters

*link_*	the active link

7.1.2.3 ssize_t esl::connection::complex_read (char * *data_*, size_t *sz_*, uint32_t *flags_*) throw error) [inline]

read data from network

Parameters

*data_*	the data buffer to read into
*sz_*	the sizeof of the above buffer
*flags_*	read with complex flags

Returns

< 0 if the link has finished otherwise the number of bytes receveid

7.1.2.4 double esl::connection::current_time ( ) const [inline]

Function to obtain the curent status of the run.

the current time

Returns

the last time stamp received

7.1.2.5 void esl::connection::dump (const std::string & *file_*, const std::string & *prefix_*) [inline]

set the dump information

Parameters

*file_*	dump file
*prefix_*	the dump prefix

7.1.2.6 void esl::connection::dump_limit (int *limit_*) [inline]

set the maximum number of dump logs

Parameters

*limit_*	the limit of the number of dump logs

7.1.2.7 void esl::connection::dump_state ( ) [inline]

Dumps the current rx state to stderr

7.1.2.8 void esl::connection::ei_esc_eep ( ) throw error) [inline]

Inject an ESC EEP

7.1.2.9 void esl::connection::ei_esc_eop ( ) throw error) [inline]

Inject an ESC EOP

7.1.2.10 void esl::connection::ei_esc_esc ( ) throw error) [inline]

Inject an ESC ESC

7.1.2.11 void esl::connection::ei_esc_fct ( ) throw error) [inline]

Inject an ESC FCT

7.1.2.12 void esl::connection::EI_flow_control (int *initial_fct_*, int *fct_*) throw error) [inline]

Error injection FCT control

Parameters

*initial_fct_*	initial fcts
*fct_*	fcts

7.1.2.13 void esl::connection::EI_ignore_events (int *what_*) throw error) [inline]

Error injection ignore events

Parameters

what	to ignore

7.1.2.14 void esl::connection::ei_parity_error ( ) throw error) [inline]

Inject a parity error

7.1.2.15 void esl::connection::eintr (bool *return_*) [inline]

set the EINTR handling

Parameters

*return_*	is the EINTR passed to calling application

7.1.2.16 virtual void esl::connection::ER_reporting (int *when_*) throw error) [inline], [virtual]

set the error reporting for the current link

Parameters

*when_*	the reporting options

7.1.2.17 void esl::connection::EW_clear (int *link_*) throw error) [inline]

clear the error waveform data for the given link

Parameters

*link_*	the link to clear waveform data

7.1.2.18 virtual void esl::connection::EW_reporting (int *when_*) throw error) [inline], [virtual]

set the error waveform for the current link

Parameters

*when_*	the waveform trigger options

7.1.2.19 void esl::connection::EW_request (int *link_*) throw error) [inline]

request the error waveform data for the given link

Parameters

*link_*	the link to request waveform data

7.1.2.20 void esl::connection::EW_reset (int *link_*) throw error) [inline]

reset error waveform capture settings

Parameters

*link_*	the link to clear waveform setgtings

7.1.2.21 virtual void esl::connection::EW_source (int *source_*) throw error) [inline], [virtual]

set the error waveform source for the current link

Parameters

*source_*	the waveform source

7.1.2.22 void esl::connection::flush ( ) throw error) [inline]

flush network buffer

7.1.2.23 int esl::connection::get_link (void) [inline]

returns the currently active link

Returns

the current rx link

7.1.2.24 uint32_t esl::connection::get_rx_timeout ( ) throw error) [inline]

return the rx timeout

Returns

timeout in milliseconds

7.1.2.25 bool esl::connection::link_connected ( ) throw error) [inline]

is the currently active tx link connected ? This function is blocking and requires the ability to read from the link You can't use this call if you have a read posted. If you have reads posted use request_link_status and handle the port_status virtual method

Returns

true if connected

7.1.2.26 void esl::connection::log (const std::string & *file_*) throw error) [inline]

set the log file

Parameters

*file_*	the name of the log file

7.1.2.27 unsigned char* esl::connection::mac ( ) throw error) [inline]

returns the device MAC address (blocking)

Returns

the mac address

7.1.2.28 std::string esl::connection::manufacturer ( ) throw error) [inline]

returns the device manufacturer (blocking)

Returns

the manufacturer

7.1.2.29 virtual void esl::connection::mode (int *mode_*) throw error) [inline], [virtual]

sets the mode of the currently active port

Parameters

*mode_*	the mode

7.1.2.30 void esl::connection::modes (uint32_t *port_*, int *mode_*) throw error) [inline]

sets the mode of the given port

Parameters

*ports_*	bit mask of the active ports we wish to set
*mode_*	the mode

7.1.2.31 int esl::connection::module (int *module_*) [inline]

return slot of module

Returns

the slot the module is in

7.1.2.32 int esl::connection::nolinks ( ) throw error) [inline]

get the number of links on the device (blocking)

Returns

number of links

7.1.2.33 void esl::connection::non_blocking (bool *nonblock_*) throw error) [inline]

set non blocking behaviour

Parameters

nonblocking	set whether we go non blocking

7.1.2.34 std::string esl::connection::options ( ) throw error) [inline]

returns the device options (blocking)

Returns

the device options

7.1.2.35 double esl::connection::percent_read ( ) [inline]

return % read of file (when reading from a file)

Returns

% of file read

7.1.2.36 std::string esl::connection::product ( ) throw error) [inline]

returns the device product name (blocking)

Returns

the prduct name

7.1.2.37 ::EtherSpaceLink esl::connection::raw ( ) [inline]

return the raw EtherSpaceLink connection

Returns

the pointer to EtherSpaceLink data

7.1.2.38 uint64_t esl::connection::raw_bytes ( ) [inline]

the number of bytes read

Returns

the nbumber of bytes read

7.1.2.39 ssize_t esl::connection::read (char * *data_*, size_t *sz_*) throw error) [inline]

read data from network

Parameters

*data_*	the data buffer to read into
*sz_*	the sizeof of the above buffer

Returns

$<$ 0 if the link has finished otherwise the number of bytes receveid

7.1.2.40 int esl::connection::read_info ( ) [inline]

return data type of previous read

Returns

PART_EOP_EEP, EOP, EEP

7.1.2.41 void esl::connection::record (const char * *file_*, bool *recordwrites_*, bool *writeerror_* = false) throw error) [inline]

set the recording file

Parameters

*file_*	the name of the file to record to
*recordwrites_*	do we record writes ?
writeerror	do we treat a write error as an error

7.1.2.42 FILE* esl::connection::record_file ( ) [inline]

obtain FILE pointer to record file

Returns

the FILE pointer to current recording file

7.1.2.43 void esl::connection::record_flush ( ) throw error) [inline]

flush the recording file

7.1.2.44 uint64_t esl::connection::record_size ( ) throw error) [inline]

obtain size of recorded data

Returns

the size of the current record file

7.1.2.45 void esl::connection::request_link_status ( ) throw error) [inline]

requests the link status of the currently active tx link

7.1.2.46 void esl::connection::request_link_status_port (int *link_*) throw error) [inline]

requests the link status of the given link (on ESL401 may change currently active link)

Parameters

link	the link to request the status of

7.1.2.47 void esl::connection::request_tx_speed ( ) throw error) [inline]

requests the link tx speed of the currently link

7.1.2.48 int esl::connection::rx_speed ( ) throw error) [inline]

returns the speed of the currently active link (blocking)

Returns

speed_ the speed (int)

7.1.2.49 void esl::connection::send_timecode (uint8_t *tc_*) throw error) [inline]

send timecode

Parameters

*tc_*	the timecode

7.1.2.50 void esl::connection::sendfile (const char * *filename_*) throw error) [inline]

send a file

Parameters

*filename_*	the file to send

7.1.2.51 std::string esl::connection::serial ( ) throw error) [inline]

returns the device serial number (blocking)

Returns

the serial number

7.1.2.52 void esl::connection::set_link (int *link_*) [inline]

sets the currently active rx link for internal use only @param link_ the link to set for transmission

7.1.2.53 void esl::connection::set_max_packet_data (int *sz_*) throw error) [inline]

set maximum packet data

Parameters

*sz_*	the size of packet data

7.1.2.54 void esl::connection::set_rx_timeout (uint32_t *timeout_*) throw error) [inline]

set the rx timeout

Parameters

*timeout_*	timeout in milliseconds

7.1.2.55 int esl::connection::slot (int *slot_*) [inline]

return module in slot

Returns

the module in slot

7.1.2.56 void esl::connection::sma_56_pulse_width (int *width_*) throw error) [inline]

set SMA pulse width

Parameters

width	the width of the port

7.1.2.57 SKT esl::connection::socket ( ) [inline]

return the network socket

Returns

connection to the device or file

7.1.2.58 void esl::connection::speed (double *speed_*) throw error) [inline]

sets the speed of the currently active port

Parameters

*speed_*	the speed (double)

7.1.2.59 void esl::connection::speed (int *speed_*) throw error) [inline]

sets the speed of the currently active port

Parameters

*speed_*	the speed (int)

7.1.2.60 virtual void esl::connection::TT_reporting (int *when_*) throw error) [inline], [virtual]

set the time tag reporting for the current link

Parameters

*when_*	the timetag options

7.1.2.61 void esl::connection::tx_record (const char * *file_*) throw error) [inline]

set the tx recording file

Parameters

*file_*	the name of the file to record to

7.1.2.62 std::string esl::connection::version ( ) [inline]

returns the API version

Returns

the API version

7.1.2.63 void esl::connection::write (const char * *data_*, size_t *sz_*, uint32_t *flags_*) throw error) [inline]

write data to network

Parameters

*data_*	the data buffer to write
*sz_*	the sizeof of the above buffer
flags_,the	type of data to write PART_EOP_EEP, EOP, EEP

7.1.2.64 bool esl::connection::write_buffer_empty ( ) throw error) [inline]

write_buffer_empty (blocking)

Returns

is the write buffer empty

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

- /autogen/cpp/EtherSpaceLink.hpp

7.2 esl::devstatus Class Reference

Public Member Functions

- void **copy** (const devstatus &orig_)
- void **copyreset** (devstatus &orig_)
- **devstatus** (const devstatus &orig_)

Public Attributes

- bool linkstatus

  Do we have link status.
- uint32_t runtime

  Capture Runtime.
- uint32_t uncertainty

  Timetag uncertaintiy.
- stats rawbytes

  Raw bytes received.
- stats bytes

  number of "real" bytes of data on all ports
- stats eop

  number of "eop" packets on all ports
- stats eep

  number of eep
- stats eecs

  number of eecs
- stats missed

  number of missed packets
- stats timeout

  number of timeouts
- stats parity

  number of parity errors
- stats error1

number of Errors

- stats error2

  number of Errors

- portstatus links [ESL_MAX_PORTS+1]

  Port status Information.

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

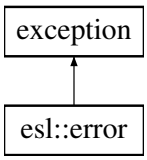
- /autogen/cpp/EtherSpaceLink.hpp

7.3 esl::error Class Reference

Exception class, whenever an error occurrs an exception is thrown, this makes error path handling somewhat easier.

`#include <EtherSpaceLink.hpp>`

Inheritance diagram for esl::error:



Public Member Functions

- error (const std::string &addr_, int e_, const char *text_)
- error (const std::string &addr_, int e_, const char *text_, int oe_)
- int get () const
- const char * text () const
- int oserr () const
- const std::string & **name** () const

7.3.1 Detailed Description

Exception class, whenever an error occurrs an exception is thrown, this makes error path handling somewhat easier.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 esl::error::error (const std::string & *addr_*, int *e_*, const char * *text_*) [inline]

constructor using current errno

Parameters

*addr_*	Name of the device
*e_*	ESL error code
*text_*	ESL error text

7.3.2.2 esl::error::error (const std::string & *addr_*, int *e_*, const char * *text_*, int *oe_*) [inline]

constructor using specified error

Parameters

Name	of the device
*e_*	ESL error code
*text_*	ESL error text
*oe_*	Operating system errno

7.3.3 Member Function Documentation

7.3.3.1 int esl::error::get ( ) const [inline]

get return the ESL error

Returns

the ESL error code

7.3.3.2 int esl::error::oserr ( ) const [inline]

get return the OS error

Returns

the OS error code

7.3.3.3 const char* esl::error::text ( ) const [inline]

text return the ESL error text, the error text should be saved before the next call

Returns

the ESL error text

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

- /autogen/cpp/EtherSpaceLink.hpp

7.4 esl::utils::lock Class Reference

Public Member Functions

- void **acquire** ()
- void **release** ()

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

- /autogen/cpp/EtherSpaceLink.hpp

7.5 esl::portstatus Class Reference

Public Member Functions

- void **copy** (const portstatus &orig_)
- void **copyreset** (portstatus &orig_)

Public Attributes

- int portno

  Which Port we are referring to.
- int runstatus

  The Port Status.
- float rxspeed

  Current Speed.
- bool active

  Is the link active ?
- bool connected

  Is the link connected.
- stats bytes

  Number of bytes received on the port.
- stats eop

  Number of EOP packets.
- stats eep

  Number of EEP packets.
- stats eecs

  Number of EECS packets.
- stats timeout

  Number of Timeouts.
- stats parity

  Number of Parity Errors.
- stats missed

  Number of Missed packets.
- stats error1

  Number of Errors.
- stats error2

  Number of Errors.
- uint8_t timecode

  Time code.

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

- /autogen/cpp/EtherSpaceLink.hpp

7.6 esl::utils::scopelock Class Reference

Public Member Functions

- **scopelock** (lock &l_)

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

- /autogen/cpp/EtherSpaceLink.hpp

7.7 esl::stats Class Reference

Public Member Functions

- void **copy** (const stats &orig_)
- void **copyreset** (stats &orig_)

Public Attributes

- uint64_t total
  total
- uint64_t last
  Last total.

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

- /autogen/cpp/EtherSpaceLink.hpp

7.8 esl::timestamp Struct Reference

connection this class provides the base class of communicating with a DSI/MSR

`#include <EtherSpaceLink.hpp>`

Public Member Functions

- timestamp & **operator=** (const timestamp &other)

Public Attributes

- uint64_t raw
  Current timetag.
- time_t time
  Packet time.
- long ns
  Nano seconds.
- bool active
  has a timestamp event occurred
- bool present
  is there a valid timestamp present

7.8.1 Detailed Description

connection this class provides the base class of communicating with a DSI/MSR

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

- /autogen/cpp/EtherSpaceLink.hpp

Index

~connection
    esl::connection, 48

abort
    esl::connection, 49
active_port
    esl::connection, 49

barrier_lifted
    functions, 28

complex_read
    esl::connection, 49
Connection, 9
connection
    esl::connection, 48
current_time
    esl::connection, 49

dump
    esl::connection, 49
dump_limit
    esl::connection, 49
dump_state
    esl::connection, 50

EI_flow_control
    esl::connection, 50
EI_ignore_events
    esl::connection, 50
ER_reporting
    esl::connection, 51
EW_clear
    esl::connection, 51
EW_reporting
    esl::connection, 51
EW_request
    esl::connection, 51
EW_reset
    esl::connection, 51
EW_source
    esl::connection, 51
ei_esc_eep
    esl::connection, 50
ei_esc_eop
    esl::connection, 50
ei_esc_esc
    esl::connection, 50
ei_esc_fct
    esl::connection, 50
ei_parity_error
    esl::connection, 50
eintr
    esl::connection, 50
err
    functions, 29
error
    esl::error, 60
Error Codes, 26
Error mask fields, 22
Error Reporting, 17
Error Waveform Sources, 24
Error Waveform Triggers, 23
Error Waveforms, 18
error_event
    functions, 29
esc_eep
    functions, 29
esc_eop
    functions, 29
esc_esc
    functions, 29
esl, 35
esl::connection, 37
    ~connection, 48
    abort, 49
    active_port, 49
    complex_read, 49
    connection, 48
    current_time, 49
    dump, 49
    dump_limit, 49
    dump_state, 50
    EI_flow_control, 50
    EI_ignore_events, 50
    ER_reporting, 51
    EW_clear, 51
    EW_reporting, 51
    EW_request, 51
    EW_reset, 51
    EW_source, 51
    ei_esc_eep, 50
    ei_esc_eop, 50
    ei_esc_esc, 50
    ei_esc_fct, 50
    ei_parity_error, 50
    eintr, 50
    flush, 51
    get_link, 52
    get_rx_timeout, 52

link_connected, 52
log, 52
mac, 52
manufacturer, 52
mode, 52
modes, 54
module, 54
nolinks, 54
non_blocking, 54
options, 54
percent_read, 54
product, 54
raw, 55
raw_bytes, 55
read, 55
read_info, 55
record, 55
record_file, 56
record_flush, 56
record_size, 56
request_link_status, 56
request_link_status_port, 56
request_tx_speed, 56
rx_speed, 56
send_timecode, 56
sendfile, 57
serial, 57
set_link, 57
set_max_packet_data, 57
set_rx_timeout, 57
slot, 57
sma_56_pulse_width, 57
socket, 58
speed, 58
TT_reporting, 58
tx_record, 58
version, 58
write, 58
write_buffer_empty, 59
esl::devstatus, 59
esl::error, 60
error, 60
get, 61
oserr, 61
text, 61
esl::portstatus, 61
esl::stats, 63
esl::timestamp, 63
esl::utils::lock, 61
esl::utils::scopelock, 62
Event handling on spacewire links, 13
Extension codes, 19

flush
esl::connection, 51
functions, 28
barrier_lifted, 28
err, 29
error_event, 29
esc_eep, 29
esc_eop, 29
esc_esc, 29
link_selected, 29
link_timeout, 30
link_tx_speed, 30
missing_data, 30
parity_error, 30
perror1, 30
perror2, 30
port_status, 30
received_esc_fct, 31
received_fct, 31
reported_version, 31
startdate, 31
timecode, 31
timetag, 32
timetag_uncertainty, 32
timezero, 32
unknown_extn_data, 32
unknown_ram_data, 33
unknown_special_data, 33
waveform_data, 33

get
esl::error, 61
get_link
esl::connection, 52
get_rx_timeout
esl::connection, 52

Handling Spacewire Traffic, 12

link_connected
esl::connection, 52
link_selected
functions, 29
link_timeout
functions, 30
link_tx_speed
functions, 30
log
esl::connection, 52

mac
esl::connection, 52
manufacturer
esl::connection, 52
Memory Mapped Addresses, 25
missing_data
functions, 30
mode
esl::connection, 52
modes
esl::connection, 54
module
esl::connection, 54

nolinks

esl::connection, 54
non_blocking
esl::connection, 54

options
esl::connection, 54
oserr
esl::error, 61

parity_error
functions, 30
percent_read
esl::connection, 54
perror1
functions, 30
perror2
functions, 30
Physical Link Attributes, 10
port_status
functions, 30
product
esl::connection, 54

raw
esl::connection, 55
raw_bytes
esl::connection, 55
read
esl::connection, 55
read_info
esl::connection, 55
Reading data from a spacewire link, 15
received_esc_fct
functions, 31
received_fct
functions, 31
record
esl::connection, 55
record_file
esl::connection, 56
record_flush
esl::connection, 56
record_size
esl::connection, 56
reported_version
functions, 31
request_link_status
esl::connection, 56
request_link_status_port
esl::connection, 56
request_tx_speed
esl::connection, 56
rx_speed
esl::connection, 56

send_timecode
esl::connection, 56
sendfile
esl::connection, 57
Sending data on a spaceiwre link, 14
serial
esl::connection, 57
set_link
esl::connection, 57
set_max_packet_data
esl::connection, 57
set_rx_timeout
esl::connection, 57
slot
esl::connection, 57
sma_56_pulse_width
esl::connection, 57
socket
esl::connection, 58
speed
esl::connection, 58
startdate
functions, 31

TT_reporting
esl::connection, 58
text
esl::error, 61
TimeTag, 16
TimeTag mask fields, 21
timecode
functions, 31
timetag
functions, 32
timetag_uncertainty
functions, 32
timezero
functions, 32
tx_record
esl::connection, 58

unknown_extn_data
functions, 32
unknown_ram_data
functions, 33
unknown_special_data
functions, 33

version
esl::connection, 58
Virtual Link Attributes, 11

waveform_data
functions, 33
write
esl::connection, 58
write_buffer_empty
esl::connection, 59