EtherSpaceLinks

Generated by Doxygen 1.8.5

Wed Aug 5 2020 16:21:00

Contents

Chapter 1

Hardware

Member EtherSpaceLink_set_speed_double (EtherSpaceLink link, double speed)

dsi

Member EtherSpaceLink_write_packet (EtherSpaceLink link, void *buffer, size_t length, uint32_t flags)

dsi

Chapter 2

Bug List

Member EtherSpaceLink_autonomous_timecode_report (EtherSpaceLink link, void ∗buffer)

no check on buffer length

Member EtherSpaceLink_extract_timetag (EtherSpaceLink link, void ∗buffer)

no checking on buffer length

Member EtherSpaceLink_extract_timetag_ns (EtherSpaceLink link, void ∗buffer)

no checking on buffer length

Member EtherSpaceLink_get_control_packet (EtherSpaceLink link, void ∗control_buffer, size_t buffer_length, int slot)

does not handle error on sending the request

Member EtherSpaceLink_get_HWA (EtherSpaceLink link, unsigned char ∗HWA)

if the buffer is less than 6 bytes in size, errors may occurr

eats traffic, if you call this whilst analyzing traffic data will be lost

can return random mac address if there is network error

Member EtherSpaceLink_open_as_server (SKT sock)

For future transports this function may be obsoleted

Member EtherSpaceLink_read_packet_extension_callback (EtherSpaceLink link, void ∗callback_buffer, size_t callback_buffer_length, ESL_CALLBACK callback)

if buffer goes out of scope program may crash

Member EtherSpaceLink_read_packet_special_callback (EtherSpaceLink link, void ∗callback_buffer, size_t callback_buffer_length, ESL_CALLBACK callback)

if buffer goes out of scope program may crash

Member EtherSpaceLink_set_log_file (EtherSpaceLink link, char ∗file_name)

errno may not be preserved

in the event of an error may pollute FD 1

Member EtherSpaceLink_set_speed_double (EtherSpaceLink link, double speed)

Don't actually know if speed has been set

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 Connection

Functions

- ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open (char *address)

  opens a connection to the specified device
- ESL_FN_EXPORT int EtherSpaceLink_device_type (EtherSpaceLink link)

  returns the device type
- ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_flagged (char *filename, uint32_t flags_, int timeout)

  opens a connection to a device or file
- ESL_FN_EXPORT int EtherSpaceLink_File (EtherSpaceLink link, ESL_STRING file)

  starts a connection reading frrom a file

5.1.1 Detailed Description

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

5.1.2 Function Documentation

5.1.2.1 ESL_FN_EXPORT int EtherSpaceLink_device_type (EtherSpaceLink *link*)

returns the device type

Parameters

link	the handle on opened filed

Returns

code indicating the device type

```
// connect to host
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}
```

```
printf ("Device type = %d\n",  EtherSpaceLink_device_type(esldev));

return 0;
```

5.1.2.2 ESL_FN_EXPORT int EtherSpaceLink_File (EtherSpaceLink *link*, ESL_STRING *file*)

starts a connection reading frrom a file

This function allows the connection to switch over to a different file maintaining state and context

Parameters

link	the handle on opened filed
filename	the name of the file

Returns

0 if successful

```
// connect to host
char * filename = "msrrun.txt";
char * nextfile = "msrrun2.txt";

printf ("Reading file %s ...\n", filename);
EtherSpaceLink esldev = EtherSpaceLink_open_flagged(filename,
      EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
     printf("Unable to open  %s error %d\n", filename, EtherSpaceLink_get_error());
     return 0;
}

printf ("Opened recording file ...");


// sometime later after processing first file
if (EtherSpaceLink_File(esldev, nextfile))
{
     printf("Unable to open %s error %d", nextfile,  EtherSpaceLink_get_error());
     return 1;
}

return 0;
```

5.1.2.3 ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open (char * *address*)

opens a connection to the specified device

Opens a connection the the etherspace link device specified which may be resolvable hostname or an ipv4 address.

A port number can be specified by adding a suffix with :portnumber. For example, 1.2.3.4:9999 will connect to a device at IP 1.2.3.4 with port number 9999

It also reads the table of modules installed in the EtherSpaceLink to an internal buffer, for use by procedures accessing status and module information. When opened, the SpaceWire link will be in the disabled state and its default speed will be 10Mb/s. Module and link parameters can be set immediately but the link must be started (using EtherSpaceLink_set_mode()) before data can be transferred over the SpaceWire link.

IPV6 is currently not supported by our devices

Note on the first call to this function we set the SIG_PIPE handler to SIG_IGN.

Parameters

address	The address / address:port specifier

Returns

EtherSpaceLink NULL if there was an error otherwise a EtherSpaceLink Handle

```
// connect to host
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}

printf ("Connected to %s with handle %p\n", host, esldev);

return 0;
```

5.1.2.4 ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_flagged (char * *filename*, uint32_t *flags_*, int *timeout*)

opens a connection to a device or file

This function establishes an ESL connection to a device or a file. It is different from the open call in that it can be used to open a file, and in the case of an ESL device a connect timeout may be given.

Parameters

filename	the name of the file/hostname
*flags_*	operations for the connection EtherSpaceLink_CONNECT_FILE connecting to a file EtherSpaceLink_CONNECT_TIMEOUT connect with a timeout EtherSpaceLink_CONNECT_RX_TIMEOUT set initial timeout

Returns

EtherSpaceLink NULL if there was an error otherwise a EtherSpaceLink Handle

```
// connect to host
char * filename = "msrrun.txt";
int ec = 0;

printf ("Reading file %s ...\n", filename);
EtherSpaceLink esldev = EtherSpaceLink_open_flagged(filename,
      EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
     printf("Unable to open  %s error %d\n", filename, EtherSpaceLink_get_error());
     return 0;
}

printf ("Opened recording file ...");

return 0;
```

5.2 Physical Link Attributes

Functions

- ESL_FN_EXPORT int EtherSpaceLink_set_speed (EtherSpaceLink link, int speed)

  Sets the transmit speed of the link
  Sets the transmit speed of all of the SpaceWire links on thisEtherSpaceLink unit.
- ESL_FN_EXPORT int EtherSpaceLink_set_mode (EtherSpaceLink link, int mode)

  set mode of current link
- ESL_FN_EXPORT int EtherSpaceLink_set_mode_portmask (EtherSpaceLink link, int mode, uint32_t ports)

  set mode of list of links
- ESL_FN_EXPORT int EtherSpaceLink_link_connected (EtherSpaceLink link)

  returns if the currently active link is connected
- ESL_FN_EXPORT int EtherSpaceLink_set_active_link (EtherSpaceLink link, int n)

  Sets the currently active link.

5.2.1 Detailed Description

Functions and definitions for controlling physical link attributes

5.2.2 Function Documentation

5.2.2.1 ESL_FN_EXPORT int EtherSpaceLink_link_connected (EtherSpaceLink *link*)

returns if the currently active link is connected

Parameters

link	connection to the device

Returns

0 not connected, < 0 if error, 1 connected

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 1;
}

int con;
switch ((con = EtherSpaceLink_link_connected ( esldev )))
{
    case 0:         // Not Connected
        printf ("Not Connected\n");
        break;

    case 1:         // Connected
        printf ("Connected\n");
        break;

    cdefault:
        printf ("Error %d\n", con);
              return 2;
        break;

    default:    // Unknown return 1;
        printf ("Unkown return %d\n", con);
              return 3;
```

```
        break;
}

return 0;
```

5.2.2.2 ESL_FN_EXPORT int EtherSpaceLink_set_active_link (EtherSpaceLink *link,* int *n*)

Sets the currently active link.

Parameters

link	connection to the device
n	the link we want to make active

Returns

< 0 error, 0 success

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 2;
}

printf("Made link 1 active\n");
return 0;
```

5.2.2.3 ESL_FN_EXPORT int EtherSpaceLink_set_mode (EtherSpaceLink *link,* int *mode*)

set mode of current link

Sets the operating mode of the currently-active SpaceWire link.

After opening a connection, the link is disabled; it must then be enabled into one of its operational modes before data can be transferred.

One of the three modes EtherSpaceLink_LINK_mode_disabled, EtherSpaceLink_LINK_mode_normal and EtherSpaceLink_LINK_mode_legacy should be chosen.

The use of EtherSpaceLink_LINK_mode_fixed_speed to set some DSI ports to 10Mb/s, together with the conventional EtherSpaceLink_set_speed() mechanism is the only way to run a DSIs links at two different speeds

Parameters

link	connection to the device
mode	of operation

EtherSpaceLink_LINK_mode_disabled

```
 The link is idle and silent.
```

EtherSpaceLink_LINK_mode_normal

```
Start the link by actively trying to establish contact.
```

EtherSpaceLink_LINK_mode_legacy

```
Dont start until activity on the link is seen. Use with SMCS/TSS901 devices.
```

EtherSpaceLink_LINK_mode_long_timeout

Extends the timeout period in the link state machine to provide a potentialy more reliable link start at very low data rates (i.e. for slow (lowpower) links near to 2Mb/s). It is necessary to set the link speed with an EtherSpaceLink_set_speed() API call before calling EtherSpaceLink_set_mode() EtherSpaceLink_LINK_mode_slow_speed.

EtherSpaceLink_LINK_mode_fixed_speed

The link speed remains at its default startup speed (10Mb/s nominal; actually within the range 9.8 to 10.2Mb/s)

EtherSpaceLink_LINK_mode_slow_speed

This setting combines the long_timeout and fixed_speed modifiers, thereby also setting the initial link speed to the final operating speed.

Returns

0 if the request queued , !0 if not

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_active_link ( esldev , 4 ))
{
    printf ("Unable to set  active link\n");
    return 2;
}

if (EtherSpaceLink_set_speed( esldev, 20 ))
{
    printf ("Unable to set 20MBs\n");
    return 3;
}

if (EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal))
{
    printf ("Failed to set mode\n");
    return 4;
}

return 0;


// For 10Mb/s operation on port 3, and 50Mb/s operation on other ports, use [on a DSI or SRR]:

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
}

EtherSpaceLink_set_active_link ( esldev, 3 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal | EtherSpaceLink_LINK_mode_fixed_speed );
EtherSpaceLink_set_active_link ( esldev, 1 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );
EtherSpaceLink_set_active_link ( esldev, 2 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );
EtherSpaceLink_set_speed_double ( esldev, 50.0 );


// For 2Mb/s operation on port 5, use [on a DSI or SRR]:

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
 }

 EtherSpaceLink_set_active_link ( esldev, 5 );
 EtherSpaceLink_set_speed ( esldev, 2 );
 EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal | EtherSpaceLink_LINK_mode_slow_speed );
```

5.2.2.4 ESL_FN_EXPORT int EtherSpaceLink_set_mode_portmask (EtherSpaceLink *link,* int *mode,* uint32_t *ports*)

set mode of list of links

Sets the operating mode of a given set of links

After opening a connection, the link is disabled; it must then be enabled into one of its operational modes before data can be transferred.

The use of EtherSpaceLink_LINK_mode_fixed_speed to set some DSI ports to 10Mb/s, together with the conventional EtherSpaceLink_set_speed() mechanism is the only way to run a DSIs links at two different speeds. The active port is the highest number listed port in the mask

Parameters

link	connection to the device
mode	of operation

Returns

0 if the request queued , !0 if not

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_mode_portmask ( esldev,
       EtherSpaceLink_LINK_mode_normal , (1<<4) | (1<<3)))
{
    printf ("Failed to set mode\n");
    return 4;
}

return 0;
```

5.2.2.5 ESL_FN_EXPORT int EtherSpaceLink_set_speed (EtherSpaceLink *link,* int *speed*)

Sets the transmit speed of the link

Sets the transmit speed of all of the SpaceWire links on thisEtherSpaceLink unit.

Links set with the additional mode modifier EtherSpaceLink_LINK_mode_fixed_speed, which remain at their start-up speed of 10Mb/s.

Parameters

link	connection to the device
speed	the number of megabits per second

Returns

0 if request has been put on the wire, !0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_speed( esldev, 100))
{
     printf( "Failed to set speed connect\n");
     return 1;
}

return 0;
```

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

Functions and definitions for handling events on spacewire links

5.6 Sending data on a spaceiwre link

Functions

- ESL_FN_EXPORT int EtherSpaceLink_write_packet (EtherSpaceLink link, void *buffer, size_t length, uint32_t flags)

  queue data for transmission
  Queues message for transmission, if there is no room left in the buffer, the buffer is transmitted. Note, that even the queued data is transmitted the data added to it may not be. If you want to guarantee transmission of this data you need to call flush.
- ESL_FN_EXPORT int EtherSpaceLink_flush (EtherSpaceLink link)

  transmit any buffered data
  EtherSpaceLink_write_packet may queue data for transport, this function puts queued data onto the wire

5.6.1 Detailed Description

Functions and definitions for sending data

5.6.2 Function Documentation

5.6.2.1 ESL_FN_EXPORT int EtherSpaceLink_flush (EtherSpaceLink *link*)

transmit any buffered data

EtherSpaceLink_write_packet may queue data for transport, this function puts queued data onto the wire

Parameters

link	connection to ESL device
buffer	the buffer to send
length	the length of the buffer to send
flags	indicating how the data to is be treated EtherSpaceLink_EOP the data is to be terminated with an EOP EtherSpaceLink_EEP the data is to be terminated with an EEP EtherSpaceLink_PART_EOP_EEP the data is not yet terminated EtherSpaceLink_INCOMPLETE the data is not yet terminated (but queued in such a way on termination it will be sent in one block)

If you logically OR the flags value with EtherSpaceLink_FLUSH a network flush is perfomed and the data is transmitted, if this is not performed data will be only transmitted when the network buffer is full or the flush method is called

Returns

0 if sucessfull, !0 if not, errno setup and error code in handle

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[60];

int sz = snprintf(msg, sizeof(msg), "Hello world %ld\n", time(NULL));

// send message with an implicit flush of network data
if (EtherSpaceLink_write_packet(esldev, msg, sz,
      EtherSpaceLink_EOP | EtherSpaceLink_FLUSH))
{
    // Error sending information or queuing ...
```

```
    printf ("Error Sending\n");
      return 0;
}

printf ("Sent\n");
return 0;
```

5.6.2.2 ESL_FN_EXPORT int EtherSpaceLink_write_packet (EtherSpaceLink *link*, void * *buffer*, size_t *length*, uint32_t *flags*)

queue data for transmission

Queues message for transmission, if there is no room left in the buffer, the buffer is transmitted. Note, that even the queued data is transmitted the data added to it may not be. If you want to guarantee transmission of this data you need to call flush.

Parameters

link	connection to ESL device
buffer	the data to send
length	the size of the buffer to size
flags	additional metadata about the frame we are transmitting

EtherSpaceLink_EOP

```
This is the last part, or all, of a data packet; an end-ofpacket  (EOP) is added.
```

EtherSpaceLink_EEP

```
This is the last part, or all, of a data packet; an error endof packet (EEP) is added.
```

EtherSpaceLink_PART_EOP_EEP

```
This is part of a data packet; no end-of-packet is added. This effectively allows one to send
part packet data, do not rely on this working correctly with other devices as it is not part
of the spacewire specification.
```

EtherSpaceLink_EXTN

```
This is a complete extension character sequence. Extension packets have a maximum length of 60 bytes.
```

EherSpaceLink_SPECIAL

```
This is a complete special packet
```

EEP would not normally be used to terminate a packet but is available here to assist with testing where an erroneous packet may usefully be generated. Data is queued in buffers in the API in order to make best use of the TCP/IP stream and may not be sent immediately. EtherSpaceLink_flush() should be used to ensure the immediate transmission of any buffered data. The one-character extension sequences may be sent using EtherSpaceLink_write_EXTN.

Hardware dsi

Returns

0 if sucessfull, <0 if not (-error number), errno setup and error code in handle

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf("Unable to connect\n");
     return 1;
}
```

```
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}

EtherSpaceLink_flush ( esldev );
printf ("Message sent\n");

char rx_buffer[1024];
uint32_t flags;
int loop_index;
int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
printf ( "Received packet of length %i: ", bytes_received );
for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
  printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}
printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
      EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

5.7 Reading data from a spacewire link

Functions

- ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent (EtherSpaceLink link)

  reads a spacewire packet of data
  Receives data from the SpaceWire link via the EtherSpaceLink unit. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

- ESL_FN_EXPORT uint32_t EtherSpaceLink_get_rx_flags (EtherSpaceLink link)

  return message flags of last packet data read
  Sometimes it may be necessary to obtain the message flags outside of the read call

- ESL_FN_EXPORT ssize_t EtherSpaceLink_read_packet_full (EtherSpaceLink link, void ∗buffer, size_t buffer_length, uint32_t ∗rx_flags, int special_data_action)

  reads a packet handling out of band information
  Receives data from the SpaceWire link via the EtherSpaceLink unit with option to include special and extension data frames. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

- ESL_FN_EXPORT ssize_t EtherSpaceLink_get_packet (EtherSpaceLink link, void ∗buffer, ssize_t offset, ssize_t buffer_length, int sda)

  read packet not returning packet type.
  This function is similar to that of read_packet_full, however the rx_flags parameter is not present and as such can be retrieved by calling EtherSpaceLink_get_rx_flags

5.7.1 Detailed Description

Functions and definitions for reading data

5.7.2 Function Documentation

5.7.2.1 ESL_FN_EXPORT ssize_t EtherSpaceLink_get_packet (EtherSpaceLink *link,* void ∗ *buffer,* ssize_t *offset,* ssize_t *buffer_length,* int *sda*)

read packet not returning packet type.

This function is similar to that of read_packet_full, however the rx_flags parameter is not present and as such can be retrieved by calling EtherSpaceLink_get_rx_flags

Parameters

link	connection to ESL device
buffer	where to read data into
offset	offset into the above buffer (i.e. data written to buffer+offset)
buffer_length	the number of bytes to read
sda	how to treat special_actions lower 4 bits enumerate to EtherSpaceLink_DISCARD_SPECIAL_DATA ignores special data EtherSpaceLink_REPORT_SPECIAL_DATA returns special data as -ve return value EtherSpaceLink_RETURN_SPECIAL_DATA returns data as normal message EtherSpaceLink_CALLBACK_SPECIAL_DATA calls callback upper 4 bits enumberate to EtherSpaceLink_DISCARD_EXTENSION_DATA (0) ignores extension data EtherSpaceLink_REPORT_EXTENSION_DATA returns extension data as -ve return value EtherSpaceLink_RETURN_EXTENSION_DATA returns data as normal message EtherSpaceLink_CALLBACK_EXTENSION_DATA calls callback

Returns

< 0 error code

number of bytes the in the message

```
*
 // connect to host on port 1234
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
      return 1;
 }

 EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

 // build message
 char msg[1024];

 int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

 // send message
 if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
 {
     // Error sending information or queuing ...
      printf("Unable to send\n");
      return 1;
 }

 // get the data on the wire
 if (EtherSpaceLink_flush ( esldev ))
 {
      printf("Unable to clear buffers\n");
      return 2;
 }
 EtherSpaceLink_flush ( esldev );

 printf ("Message sent\n");

 char rx_buffer[1024];
 int loop_index;
 uint32_t flags;
 int bytes_received = EtherSpaceLink_get_packet ( esldev, rx_buffer, 0, sizeof (
      rx_buffer), EtherSpaceLink_DISCARD_SPECIAL_DATA| EtherSpaceLink_DISCARD_EXTENSION_DATA);
 printf ( "Received packet of length %i: ", bytes_received );
```

```
for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
    printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}

// Record what kind of packet it was
printf ( " %s\n", EtherSpaceLink_get_rx_flags (esldev) ==
     EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
     EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

5.7.2.2 ESL_FN_EXPORT uint32_t EtherSpaceLink_get_rx_flags (EtherSpaceLink *link*)

return message flags of last packet data read

Sometimes it may be necessary to obtain the message flags outside of the read call

This function returns the value of the flags performed by the last read call.

Parameters

link	connection to ESL device

```
  // connect to host on port 1234
  EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent(
      EtherSpaceLink link);
  if (!esldev)
  {
       printf( "Unable to connect\n");
       return 1;
  }

  EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

  // build message
  char msg[1024];

  int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

  // send message
  if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
  {
      // Error sending information or queuing ...
       printf("Unable to send\n");
       return 1;
  }

  // get the data on the wire
  if (EtherSpaceLink_flush ( esldev ))
  {
       printf("Unable to clear buffers\n");
       return 2;
  }

   char rx_buffer[1024];
   uint32_t flags;

   int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
   printf ( "Received packet of length %i: with flags %08x:%08x\n ", bytes_received, flags,
      EtherSpaceLink_get_rx_flags(esldev));

   EtherSpaceLink_close ( esldev );
   return 0;
```

5.7.2.3 ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent (EtherSpaceLink *link*)

reads a spacewire packet of data

Receives data from the SpaceWire link via the EtherSpaceLink unit. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

Extension sequences and Special packets are never returned by this interface.

Use EtherSpaceLink_read_packet_full if you wish to receive special packets or extension sequences.

Parameters

link	connection to ESL device
buffer	the target destination for the data we want to read
buffer_length	the size of the buffer
rx_flags	pointer to metadata about the frame EtherSpaceLink_EOP This is the last part, or all, of a packet; an end-of-packet (EOP) was received. EtherSpaceLink_EEP This is the last part, or all, of a packet containing an error; an error end-of-packet EtherSpaceLink_PART_EOP_EEP The packet was larger than the available buffer. Another read will retrieve more data f

Returns

<0 error, number of bytes the in the message

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}

printf ("Message sent\n");

 char rx_buffer[1024];
 uint32_t flags;
 int loop_index;

 int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
 printf ( "Received packet of length %i: ", bytes_received );
 for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
 {
     printf ( " #%02x", rx_buffer[loop_index] & 0xff );
 }
 printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
      EtherSpaceLink_EOP ? "EOP" : "EEP");

 EtherSpaceLink_close ( esldev );
 return 0;
```

5.7.2.4 ESL_FN_EXPORT ssize_t EtherSpaceLink_read_packet_full (EtherSpaceLink *link*, void * *buffer*, size_t *buffer_length*, uint32_t * *rx_flags*, int *special_data_action*)

reads a packet handling out of band information

Receives data from the SpaceWire link via the EtherSpaceLink unit with option to include special and extension data frames. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

Parameters

link	connection to ESL device
buffer	where to read data into
buffer_length	the number of bytes to read
rx_flags	message flags EtherSpaceLink_EOP This is the last part, or all, of a packet; an end-of-packet (EOP) was received. EtherSpaceLink_EEP This is the last part, or all, of a packet containing an error; an error end-of-packet EtherSpaceLink_PART_EOP_EEP The packet was larger than the available buffer. Another read will retrieve more data f EtherSpaceLink_EXTN Data frame is an extended frame EtherSpaceLink_PART_EXTN Data frame is an extended frame but is as yet incomplete and the functiion should be ca EtherSpaceLink_SPECIAL Data frame is an special frame EtherSpaceLink_PART_EXTN Data frame is an special frame but is as yet incomplete and the functiion should be cal EtherSpaceLink_SPECIAL_SIZE When called with REPORT_SPECIAL_SIZE indicates how much special data follows EtherSpaceLink_EXTENSION_SIZE When called with REPORT_SPECIAL_SIZE indicates how much extension data follows
special_data_action	how to treat special_actions lower 4 bits enumerate to EtherSpaceLink_DISCARD_SPECIAL_DATA ignores special data EtherSpaceLink_REPORT_SPECIAL_DATA returns special data size (and flags set to EtherSpaceLink_ EtherSpaceLink_RETURN_SPECIAL_DATA returns data as normal message EtherSpaceLink_CALLBACK_SPECIAL_DATA calls callback upper 4 bits enumberate to EtherSpaceLink_DISCARD_EXTENSION_DATA (0) ignores extension data EtherSpaceLink_REPORT_EXTENSION_DATA returns extension data size (and flags set to EtherSpaceLin EtherSpaceLink_RETURN_EXTENSION_DATA returns data as normal message EtherSpaceLink_CALLBACK_EXTENSION_DATA calls callback

Returns

< 0 -ve value of error number

number of bytes the in the message

```
*
 // connect to host on port 1234
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
      return 1;
```

```
}

EtherSpaceLink_set_mode ( esldev,
     EtherSpaceLink_LINK_mode_normal );

// build message
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
     EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}
EtherSpaceLink_flush ( esldev );

printf ("Message sent\n");

char rx_buffer[1024];
int loop_index;
uint32_t flags;
int bytes_received = EtherSpaceLink_read_packet_full ( esldev, rx_buffer,
     sizeof (rx_buffer), &flags, EtherSpaceLink_DISCARD_SPECIAL_DATA| EtherSpaceLink_DISCARD_EXTENSION_DATA);
printf ( "Received packet of length %i: ", bytes_received );

for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
    printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}
printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
     EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

5.8 TimeTag

Modules

- TimeTag mask fields

Functions

- ESL_FN_EXPORT int EtherSpaceLink_TT_enable_reporting (EtherSpaceLink link, int when)

  Enable timetags for currently active link.

5.8.1 Detailed Description

Functions and definitions for reporting Timetags

5.8.2 Function Documentation

5.8.2.1 ESL_FN_EXPORT int EtherSpaceLink_TT_enable_reporting (EtherSpaceLink *link,* int *when*)

Enable timetags for currently active link.

Parameters

link	connection to the device
when	what events generate a timetag

Returns

0 on success or transmitted , $<$ 0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

// Only interested in parity errors
EtherSpaceLink_TT_enable_reporting(esldev,
      EtherSpaceLink_TT_report_parity_error);

return 0;
```

5.9 Error Reporting

Modules

- Error mask fields

Functions

- ESL_FN_EXPORT int EtherSpaceLink_ER_enable_reporting (EtherSpaceLink link, int what)

  Enables, or disables, error reporting.

5.9.1 Detailed Description

Functions and definitions for reporting Errors

5.9.2 Function Documentation

5.9.2.1 ESL_FN_EXPORT int EtherSpaceLink_ER_enable_reporting (EtherSpaceLink *link,* int *what*)

Enables, or disables, error reporting.

Parameters

link	connection to the device
what	error reporting we wish to enable

Returns

0 on success or transmitted , $<$ 0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

// Only interested in parity errors
EtherSpaceLink_ER_enable_reporting(esldev,
      EtherSpaceLink_ER_report_parity_error);

return 0;
```

5.10 Error Waveforms

Modules

- Error Waveform Triggers
- Error Waveform Sources

Functions

- ESL_FN_EXPORT int EtherSpaceLink_EW_enable_reporting (EtherSpaceLink link, int what)

  Enables, or disables, waveform capture triggers.
  Triggering may be on errors or on other significant events. The parameter what should be set to EtherSpaceLink_EW_capture_nothing to disable all reporting, or to a combination of the.
- ESL_FN_EXPORT int EtherSpaceLink_EW_source (EtherSpaceLink link, int sources)

  Selects waveform capture trigger sources.
  Triggering may be on events from ports other than that associated with the capture circuit.

5.10.1 Detailed Description

Functions and definitions for capturing waveforms

5.10.2 Function Documentation

5.10.2.1 ESL_FN_EXPORT int EtherSpaceLink_EW_enable_reporting (EtherSpaceLink *link,* int *what*)

Enables, or disables, waveform capture triggers.

Triggering may be on errors or on other significant events. The parameter what should be set to EtherSpaceLink_EW_capture_nothing to disable all reporting, or to a combination of the.

In addition to the given triggers, a (non-maskable) EVENT in the DSI transmit data stream can also trigger a waveform capture. Each port of a DSI has a waveform capture circuit. Each capture circuit can be triggered by events on its own port, and also on other ports and external events. By default, each capture circuit will respond only to its own port. EtherSpaceLink_EW_source() can be used to expand the recognised source of triggers.

Parameters

link	connection to the device
what	error reporting we wish to enable

Returns

0 on success or transmitted , $<$ 0 if error

```
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }

 // Only interested in parity errors
 EtherSpaceLink_EW_enable_reporting(esldev,
      EtherSpaceLink_EW_capture_parity_error);

 return 0;
```

5.10.2.2 ESL_FN_EXPORT int EtherSpaceLink_EW_source (EtherSpaceLink *link,* int *sources*)

Selects waveform capture trigger sources.

Triggering may be on events from ports other than that associated with the capture circuit.

For example, waveforms may be captured on all ports for an event occurring on only one of them.

Parameters

link	connection to the device
sources	EtherSpaceLink_EW_Source_port_1 Trigger on events from port 1. EtherSpaceLink_EW_Source_port_2 Trigger on events from port 2. EtherSpaceLink_EW_Source_port_3 Trigger on events from port 3. EtherSpaceLink_EW_Source_port_4 Trigger on events from port 4. EtherSpaceLink_EW_Source_port_5 Trigger on events from port 5. EtherSpaceLink_EW_Source_port_6 Trigger on events from port 6. EtherSpaceLink_EW_Source_port_7 Trigger on events from port 7. EtherSpaceLink_EW_Source_port_8 Trigger on events from port 8.

EtherSpaceLink_EW_Source_SMA_12 -LS, -MS∗ platforms Trigger on a rising edge on SMA connectors 1-2. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_SMA_34 -LS, -MS platforms Trigger on a rising edge on SMA connectors 3-4. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_SMA_56 -LS, -MS platforms Trigger on a rising edge on SMA connectors 5-6. The threshold level is 0.5 V. EtherSpaceLink_EW_Source-_SMA_78 -LS, -MS platforms Trigger on a rising edge on SMA connectors 7-8. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_barrier SO Trigger when the synchronisation barrier is lifted

Returns

0 if the request was successful

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

 // Capture waveform on ALL ports whenever an error occurs on
 // any one (or more)

int all_ports = EtherSpaceLink_EW_Source_port_1
    | EtherSpaceLink_EW_Source_port_2
    | EtherSpaceLink_EW_Source_port_3
    | EtherSpaceLink_EW_Source_port_4
    | EtherSpaceLink_EW_Source_port_5
    | EtherSpaceLink_EW_Source_port_6
    | EtherSpaceLink_EW_Source_port_7
    | EtherSpaceLink_EW_Source_port_8;
int port;
for ( port = 1; port <= 8; port ++ )
{
    EtherSpaceLink_set_active_link ( esldev, port );
    EtherSpaceLink_EW_enable_reporting (esldev,
      EtherSpaceLink_ER_report_running_error );
    EtherSpaceLink_EW_source ( esldev, all_ports );
}

return 0;
```

5.11 Extension codes

Macros

- #define **EtherSpaceLink_FCT** 0x100
- #define EtherSpaceLink_EEP 0x101

  Error End of Packet.
- #define EtherSpaceLink_EOP 0x102

  End of Packet.
- #define EtherSpaceLink_ESC 0x103

  Escape.
- #define EtherSpaceLink_ESC_FCT 0x104

  Escape FCT aka a NULL character.
- #define EtherSpaceLink_ESC_EEP 0x105

  Escape End of Packet.
- #define EtherSpaceLink_ESC_EOP 0x106

  Escape Error of packet.
- #define EtherSpaceLink_ESC_ESC 0x107

  Escape Escape.
- #define EtherSpaceLink_Timeout 0x108

  Timeout message.
- #define EtherSpaceLink_ParityError 0x109

  Parity Error message.
- #define EtherSpaceLink_PERROR1 0x10A

  Error 1 message.
- #define EtherSpaceLink_PERROR2 0x10B

  Error 2 message.
- #define **EtherSpaceLink_STORE** 0x10C
- #define **EtherSpaceLink_FORWARD** 0x10D
- #define **EtherSpaceLink_ATOM** 0x10E
- #define **EtherSpaceLink_MOTA** 0x10F
- #define **EtherSpaceLink_JOIN** 0x110
- #define **EtherSpaceLink_BARRIER** 0x111
- #define **EtherSpaceLink_RESIGN** 0x112
- #define **EtherSpaceLink_EVENT** 0x113
- #define EtherSpaceLink_Missing_data 0x114

  Missed data message.
- #define **EtherSpaceLink_HOLD** 0x12F
- #define **EtherSpaceLink_Delay** 0x130
- #define EtherSpaceLink_PortSelect 0x140

  Port select message.
- #define EtherSpaceLink_PortSelect_max 0x17F

  Max port select message.
- #define **EtherSpaceLink_Multi_byte_extn_start** 0x180
- #define EtherSpaceLink_TimeTag 0x188

  Timetag message.
- #define EtherSpaceLink_TimeTag_delta 0x182

  Timetag delta message.
- #define EtherSpaceLink_TimeTag_uncertainty 0x181

  Timetag uncertain message.
- #define EtherSpaceLink_TimeCode 0x191

  Spacewire timecode.

- #define EtherSpaceLink_Module 0x192

  Module data.
- #define EtherSpaceLink_TimeZero 0x198

  First timecode on the link.
- #define **EtherSpaceLink_TRUNCATE_1** 0x1A1
- #define **EtherSpaceLink_TRUNCATE_2** 0x1A2
- #define **EtherSpaceLink_REPEAT_1** 0x1B1
- #define **EtherSpaceLink_REPEAT_2** 0x1B2
- #define **EtherSpaceLink_REPEAT_3** 0x1B3
- #define EtherSpaceLink_Year 0x1C8

  Capture start date/time.
- #define EtherSpaceLink_Header 0x1CE

  Capture Header containing version and time information.

5.11.1 Detailed Description

5.12 TimeTag mask fields

Macros

- #define **EtherSpaceLink_TT** 7
- #define **EtherSpaceLink_TT_64** 15
- #define EtherSpaceLink_TT_report_nothing 0x00

  Report Nothing.
- #define EtherSpaceLink_TT_report_first_byte (0x01 | EtherSpaceLink_report_first_byte)

  Timetag first byte of packet.
- #define EtherSpaceLink_TT_report_intermediate_bytes (0x02 | EtherSpaceLink_report_mid_bytes)

  Timetag middle byte.
- #define EtherSpaceLink_TT_report_EOP_EEP (0x04 | EtherSpaceLink_report_EEP | EtherSpaceLink_report_EOP)

  Timetag end of packet markers.
- #define EtherSpaceLink_TT_report_EEP (0x04 | EtherSpaceLink_report_EEP)

  Timetag report Error End of Packet.
- #define EtherSpaceLink_TT_report_EOP (0x04 | EtherSpaceLink_report_EOP)

  Timetag report End of Packet.
- #define EtherSpaceLink_TT_report_time_code EtherSpaceLink_report_time_code

  Timetag report spacewire timecode.
- #define EtherSpaceLink_TT_report_fct EtherSpaceLink_report_FCT

  Timetag report FCT.
- #define EtherSpaceLink_TT_report_null EtherSpaceLink_report_NULL

  Timetag report NULL.
- #define EtherSpaceLink_TT_report_parity_error EtherSpaceLink_report_parity_error

  Timetag report parity error.
- #define EtherSpaceLink_TT_report_ESC_EOP EtherSpaceLink_report_ESC_EOP

  Timetag report ESC End of Packet.
- #define EtherSpaceLink_TT_report_ESC_EEP EtherSpaceLink_report_ESC_EEP

  Timetag report ESC Error End of Packet.
- #define EtherSpaceLink_TT_report_ESC_ESC EtherSpaceLink_report_ESC_ESC

  Timetag report ESC ESC.
- #define EtherSpaceLink_TT_report_timeout EtherSpaceLink_report_timeout

  Timetag report timeout.

5.12.1 Detailed Description

5.13 Error mask fields

Macros

- #define **EtherSpaceLink_ER** 8
- #define **EtherSpaceLink_ER_64** 16
- #define EtherSpaceLink_ER_report_nothing 0x00

  Error reporting report nothing.
- #define EtherSpaceLink_ER_report_first_null 0x02

  Error report first null.
- #define EtherSpaceLink_ER_report_first_fct 0x04

  Error report first fct.
- #define EtherSpaceLink_ER_report_running_error (0x08 | EtherSpaceLink_report_parity_error | EtherSpaceLink_report_ESC_EOP | EtherSpaceLink_report_ESC_EEP | EtherSpaceLink_report_ESC_ESC | EtherSpaceLink_report_timeout)

  Error report running.
- #define **EtherSpaceLink_ER_report_starting_error** 0x10
- #define **EtherSpaceLink_ER_report_nchar** 0x40
- #define EtherSpaceLink_ER_report_time_code (0x80 | EtherSpaceLink_report_time_code)

  report time code
- #define EtherSpaceLink_ER_report_fct EtherSpaceLink_report_FCT

  report FCT
- #define EtherSpaceLink_ER_report_null EtherSpaceLink_report_NULL

  report null
- #define EtherSpaceLink_ER_report_parity_error EtherSpaceLink_report_parity_error

  report parity error
- #define EtherSpaceLink_ER_report_ESC_EOP EtherSpaceLink_report_ESC_EOP

  report Escape End of Packet
- #define EtherSpaceLink_ER_report_ESC_EEP EtherSpaceLink_report_ESC_EEP

  report Escape Error End of Packet
- #define EtherSpaceLink_ER_report_ESC_ESC EtherSpaceLink_report_ESC_ESC

  report Escape Escape
- #define EtherSpaceLink_ER_report_timeout EtherSpaceLink_report_timeout

  report Timeout

5.13.1 Detailed Description

5.14 Error Waveform Triggers

Macros

- #define **EtherSpaceLink_EW** 9
- #define **EtherSpaceLink_EW_RT** 13
- #define **EtherSpaceLink_EW_capture_nothing** 0x00
- #define EtherSpaceLink_EW_capture_first_null (0x02 | EtherSpaceLink_report_first_null)

  trigger on first null
- #define EtherSpaceLink_EW_capture_first_fct 0x04

  trigger on first fct
- #define EtherSpaceLink_EW_capture_running_error (0x08 | EtherSpaceLink_report_parity_error | EtherSpaceLink_report_ESC_EOP | EtherSpaceLink_report_ESC_EEP | EtherSpaceLink_report_ESC_ESC | EtherSpaceLink_report_timeout)

  trigger on run error
- #define EtherSpaceLink_EW_capture_starting_error 0x10

  trigger on start error
- #define EtherSpaceLink_EW_capture_nchar (0x40 | EtherSpaceLink_report_nchar)

  trigger on n char
- #define EtherSpaceLink_EW_capture_time_code (0x80 | EtherSpaceLink_report_time_code)

  trigger on timecode
- #define EtherSpaceLink_EW_capture_EOP EtherSpaceLink_report_EOP

  trigger on End of Packet
- #define EtherSpaceLink_EW_capture_EEP EtherSpaceLink_report_EEP

  trigger on Error End of Packet
- #define EtherSpaceLink_EW_capture_FCT EtherSpaceLink_report_FCT

  trigger on FCT
- #define EtherSpaceLink_EW_capture_excess_FCT EtherSpaceLink_report_excess_FCT

  trigger on excess fct
- #define EtherSpaceLink_EW_capture_excess_data EtherSpaceLink_report_excess_data

  trigger on excess data
- #define EtherSpaceLink_EW_capture_null EtherSpaceLink_report_NULL

  trigger on NULL
- #define EtherSpaceLink_EW_capture_parity_error EtherSpaceLink_report_parity_error

  trigger on parity error
- #define EtherSpaceLink_EW_capture_ESC_EOP EtherSpaceLink_report_ESC_EOP

  trigger on Escape End of Packet
- #define EtherSpaceLink_EW_capture_ESC_EEP EtherSpaceLink_report_ESC_EEP

  trigger on Escape Error End of Packet
- #define EtherSpaceLink_EW_capture_ESC_ESC EtherSpaceLink_report_ESC_ESC

  trigger on Escape Escape
- #define EtherSpaceLink_EW_capture_timeout EtherSpaceLink_report_timeout

  trigger on timeout

5.14.1 Detailed Description

5.15 Error Waveform Sources

Macros

- #define EtherSpaceLink_EW_Source_barrier 0x0001

  Barrier.
- #define EtherSpaceLink_EW_Source_port_1 0x0002

  Port 1.
- #define EtherSpaceLink_EW_Source_port_2 0x0004

  Port 2.
- #define EtherSpaceLink_EW_Source_port_3 0x0008

  Port 3.
- #define EtherSpaceLink_EW_Source_port_4 0x0010

  Port 4.
- #define EtherSpaceLink_EW_Source_port_5 0x0020

  Port 5.
- #define EtherSpaceLink_EW_Source_port_6 0x0040

  Port 6.
- #define EtherSpaceLink_EW_Source_port_7 0x0080

  Port 7.
- #define EtherSpaceLink_EW_Source_port_8 0x0100

  Port 8.
- #define EtherSpaceLink_EW_Source_SMA_12 0x0200

  SMA 1/2 changing state.
- #define EtherSpaceLink_EW_Source_SMA_34 0x0400

  SMA 3/4 changing state.
- #define EtherSpaceLink_EW_Source_SMA_56 0x0800

  SMA 5/6 changing state.
- #define EtherSpaceLink_EW_Source_SMA_78 0x1000

  SMA 7/8 changing state.
- #define EtherSpaceLink_EW_Source_local_clock 0x8000

  Local clock.

5.15.1 Detailed Description

5.16 Memory Mapped Addresses

Macros

- #define **EtherSpaceLink_LINK_address** 0x0000
- #define **EtherSpaceLink_TX_SPEED_address** 0x87FD
- #define **EtherSpaceLink_RX_SPEED_address** 0x0001
- #define **EtherSpaceLink_HWA_address** 0x8800
- #define **EtherSpaceLink_VERSION_address** 0x880A
- #define **EtherSpaceLink_DESCRIPTION_address** 0x880B
- #define **EtherSpaceLink_OPTIONS_address** 0x8F60
- #define **EtherSpaceLink_NLINKS_address** 0x8FFF
- #define **EtherSpaceLink_EW_address** 0x1000
- #define **EtherSpaceLink_PC_address** 0x2000
- #define **EtherSpaceLink_PG_address** 0x4000
- #define **EtherSpaceLink_ATI_address** 0x0100
- #define **EtherSpaceLink_OBSERVE_address** 0x0020
- #define **EtherSpaceLink_TIMETAG_address** 0x0030
- #define **EtherSpaceLink_IGNORE_address** 0x0040
- #define **EtherSpaceLink_Event_cause_address** 0x0060
- #define **EtherSpaceLink_EW_source_address** 0x0070
- #define **EtherSpaceLink_FLOW_CONTROL_address** 0x0050
- #define **EtherSpaceLink_SMA_56_pulse_width_address** 0x00F0
- #define **EtherSpaceLink_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

Macros

- #define EtherSpaceLink_Error_RecFile_Open -1

  Couldn't open recording file.
- #define EtherSpaceLink_Error_RecFile_Write -2

  record_file write failed
- #define EtherSpaceLink_Error_LogFile_Open -3

  Couldn't open logging file.
- #define EtherSpaceLink_Error_LogFile_Write -4

  log_file write failed
- #define EtherSpaceLink_Error_Receiver_Timeout -10

  we have a network timeout timeout
- #define EtherSpaceLink_Error_Receiver_Shutdown -11

  peer has performed an orderly shutdown
- #define EtherSpaceLink_Error_IO_Error -12

  we have an IO error
- #define EtherSpaceLink_Error_SaveBuf_Overflow_Save -15

  Saving the read_packet_full() save_buffer failed.
- #define EtherSpaceLink_Error_SaveBuf_Overflow_Restore -16

  Restoring the read_packet_full() save_buffer failed.
- #define EtherSpaceLink_Error_Function_Not_Supported -17

  Device does not support the requested function.
- #define EtherSpaceLink_Error_Network -18

  Error reading / writing to/from the device.
- #define EtherSpaceLink_Error_Network_Format_Error -19

  Error understanding recieved packet.
- #define EtherSpaceLink_Error_Request_Too_Large -20

  The I/O request can't be fullfilled by the hardware.
- #define EtherSpaceLink_Error_Sequence_Error -21

  Didn't receive expected notification from the hardware.
- #define EtherSpaceLink_Error_Response_Too_Small -22

  Response from the device didn't contain enough data.
- #define EtherSpaceLink_Error_Response_Mismatch -23

  Response does not match I/O request.
- #define EtherSpaceLink_Error_Module_Not_Present -24

  Module not present.
- #define EtherSpaceLink_Error_Parameter_RangeIncorrect -25

  Parameter not in range.
- #define EtherSpaceLink_Error_File_Not_Present -26

  Requested file is not present.
- #define EtherSpaceLink_Error_EINTR -27

  EINTR occurred.
- #define EtherSpaceLink_Error_Link_Incorrect -28

  Link number is incorrect.
- #define EtherSpaceLink_Error_Incorrect_Device -29

  Connecting to a device which does not support functionality.
- #define EtherSpaceLink_Error_Memory -30

  Unable to allocate memory.
- #define EtherSpaceLink_Error_Host_Unresolvable -31

Unable to resolve host.

- #define EtherSpaceLink_Error_Host_Unresponsive -32

  Unable to connect to host.
- #define EtherSpaceLink_Error_WaveForm_Dir_Create -33

  Unable to create waveform directory.
- #define EtherSpaceLink_Error_Zero_Read -34

  asked to read zero bytes
- #define EtherSpaceLink_Error_Set_Option_File -35

  Asked to set an option when playing back from file.
- #define EtherSpaceLink_Error_Invalid_Device -36

  Device is not supported by API.
- #define EtherSpaceLink_Error_File_Move -37

  Unable to move file into place.
- #define EtherSpaceLink_Error_Invalid_File -38

  Unable to open file.
- #define EtherSpaceLink_Error_Callback_Return -39

  Callback has asked for a return.
- #define EtherSpaceLink_Error_FileList_Empty -40

  List of files given is empty.
- #define EtherSpaceLink_Error_Unknown_System_Type -41

  Unknown type.
- #define EtherSpaceLink_Error_Not_Known -42

  API returned 0 as an error should (should not happen)
- #define EtherSpaceLink_Error_EXE_Start_Failed -43

  Cannot start executable.
- #define EtherSpaceLink_Error_NO_Connection -44

  Link Not estblished.
- #define EtherSpaceLink_Error_Invalid_Link -45

  Invalid Link selected.
- #define EtherSpaceLink_Error_Would_Block -48

  I/O call would block.
- #define EtherSpaceLink_Error_Link_Not_Connected -49

  Link Not Connected.
- #define EtherSpaceLink_Error_ReadHandler_Running -50

  There is a read handler running for this connection.
- #define EtherSpaceLink_Error_Buffer_Full -51

  can't do non blocking write as buffer is full
- #define EtherSpaceLink_Error_CaptureThread_Failed -52

  Capture thread failed.
- #define EtherSpaceLink_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

Chapter 6

File Documentation

6.1 /src/api/C/EtherSpaceLink.h File Reference

This file is the primary include file used for writing applications using ESL products.

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

Macros

- #define EtherSpaceLink_version "ESL_RELID"

  The version of the software.
- #define esl_print_error()

  print error
- #define ESL_TRY(v)

  Since we don't have exceptions in C and checking large code blocks for each return value is combersome , we have an emultation of try and catch.
- #define ESL_TRY_IGNORE(v, c, err)

  try a function and don't "throw" if the error is the given one
- #define ESL_CATCH_BEGIN

  equivalent of a catch block start
- #define ESL_CATCH_END }

  equivalent of a catch block end
- #define ESL_THROW goto esl_error;

  equivalent of a throw

Typedefs

- typedef void * EtherSpaceLink
- typedef int(* ESL_CALLBACK)(EtherSpaceLink, void *, int length, int complete, int data_buffer_position)
- typedef void(* **ESL_HUP**)(int)

Functions

- ESL_FN_EXPORT void EtherSpaceLink_set_debug (int level, int output)

  sets debug level for debug library

- ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open (char *address)

  opens a connection to the specified device
- ESL_FN_EXPORT int EtherSpaceLink_device_type (EtherSpaceLink link)

  returns the device type
- ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_flagged (char *filename, uint32_t flags_, int timeout)

  opens a connection to a device or file
- ESL_FN_EXPORT int EtherSpaceLink_File (EtherSpaceLink link, ESL_STRING file)

  starts a connection reading frrom a file
- ESL_FN_EXPORT double EtherSpaceLink_get_percent_file_read (EtherSpaceLink link)

  returns how much data has been read from the file
- ESL_FN_EXPORT void * EtherSpaceLink_close (EtherSpaceLink link)

  Used to close connection to an ESL device or file.
- ESL_FN_EXPORT void * EtherSpaceLink_shutdown (EtherSpaceLink link)

  Used to terminate a connection to an ESL device.
- ESL_FN_EXPORT void EtherSpaceLink_abort (EtherSpaceLink link)

  Use to abort a connection with a thread on a read call.
- ESL_FN_EXPORT int EtherSpaceLink_get_module_slot (EtherSpaceLink link, int module)

  Returns the slot a given module resides in.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_module_string (EtherSpaceLink link, int module)

  Returns the name of a given module.
- ESL_FN_EXPORT int EtherSpaceLink_get_slot (EtherSpaceLink link, int slot)

  Returns the module in a given slot.
- ESL_FN_EXPORT int EtherSpaceLink_set_log_file (EtherSpaceLink link, char *file_name)

  Sets the current log file.
- ESL_FN_EXPORT int EtherSpaceLink_set_record_file (EtherSpaceLink link, char *file_name)

  Sets the recording file.
- ESL_FN_EXPORT int EtherSpaceLink_set_tx_record_file (EtherSpaceLink link, char *file_name)

  Sets the tx recording file.
- ESL_FN_EXPORT uint64_t EtherSpaceLink_get_record_size (EtherSpaceLink link)

  Return the amount of data written to the current recording file.
- ESL_FN_EXPORT ESL_BIN_FILE_WR EtherSpaceLink_get_record_file (EtherSpaceLink link)

  Retreive current recording file.
- ESL_FN_EXPORT int EtherSpaceLink_flush_record_file (EtherSpaceLink link)

  Flush record file.
- ESL_FN_EXPORT void EtherSpaceLink_check_record_writes (EtherSpaceLink link, int on)

  Sets up what to do in the case of an error writing to record file.
- ESL_FN_EXPORT int EtherSpaceLink_record_writes (EtherSpaceLink link)

  Returns whether errors writing to record file are treated as errors.
- ESL_FN_EXPORT void EtherSpaceLink_set_rx_timeout_action (EtherSpaceLink link, int returns_error)

  sets the behaviour on a networktimeout
- ESL_FN_EXPORT int EtherSpaceLink_get_error ()

  Retreive EtherSpaceLink error information.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_error_text ()

  return textual description of the last error
- ESL_FN_EXPORT SKT EtherSpaceLink_get_socket (EtherSpaceLink link)

  Set EtherSpaceLink error information for application use.
- ESL_FN_EXPORT void EtherSpaceLink_set_context (EtherSpaceLink link, void *context)

  Set user context can be associated with a EtherSpaceLink handle.
- ESL_FN_EXPORT void * EtherSpaceLink_get_context (EtherSpaceLink link)

  Return user context associated with a EtherSpaceLink handle.

- ESL_FN_EXPORT int EtherSpaceLink_get_rx_timeout (EtherSpaceLink link)

  Retreive the current recieve timeout (milliseconds)
- ESL_FN_EXPORT int EtherSpaceLink_set_rx_timeout (EtherSpaceLink link, int to)

  Sets the rx timeout in milliseconds.
- ESL_FN_EXPORT SKT get_socket ()

  gets a transport socket for an ESL device
- ESL_FN_EXPORT ssize_t EtherSpaceLink_send (EtherSpaceLink link, void ∗buffer, size_t size, int flags)

  Low level call to send data on ESL transport.
- ESL_FN_EXPORT int EtherSpaceLink_write_packet (EtherSpaceLink link, void ∗buffer, size_t length, uint32_t flags)

  queue data for transmission
  Queues message for transmission, if there is no room left in the buffer, the buffer is transmitted. Note, that even the queued data is transmitted the data added to it may not be. If you want to guarantee transmission of this data you need to call flush.
- ESL_FN_EXPORT int EtherSpaceLink_flush (EtherSpaceLink link)

  transmit any buffered data
  EtherSpaceLink_write_packet may queue data for transport, this function puts queued data onto the wire
- ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent (EtherSpaceLink link)

  reads a spacewire packet of data
  Receives data from the SpaceWire link via the EtherSpaceLink unit. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.
- ESL_FN_EXPORT ssize_t EtherSpaceLink_read_packet (EtherSpaceLink link, void ∗buffer, size_t buffer_length, uint32_t ∗rx_flags)

  return how many bytes of buffer have been sent
  When non blocking mode is being used it is possible that a send call will return a block error code with partial application data buffer, this call returns the number of bytes of ata have been successfully queued
- ESL_FN_EXPORT uint32_t EtherSpaceLink_get_rx_flags (EtherSpaceLink link)

  return message flags of last packet data read
  Sometimes it may be necessary to obtain the message flags outside of the read call
- ESL_FN_EXPORT ssize_t EtherSpaceLink_read_packet_full (EtherSpaceLink link, void ∗buffer, size_t buffer_length, uint32_t ∗rx_flags, int special_data_action)

  reads a packet handling out of band information
  Receives data from the SpaceWire link via the EtherSpaceLink unit with option to include special and extension data frames. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.
- ESL_FN_EXPORT ssize_t EtherSpaceLink_get_packet (EtherSpaceLink link, void ∗buffer, ssize_t offset, ssize_t buffer_length, int sda)

  read packet not returning packet type.
  This function is similar to that of read_packet_full, however the rx_flags parameter is not present and as such can be retrieved by calling EtherSpaceLink_get_rx_flags
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_version ()

  Used to get the version number of the C API being used.
- ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_as_server (SKT sock)

  cCreate an Etherspace connection using a socket
- ESL_FN_EXPORT uint64_t EtherSpaceLink_get_total_raw_bytes_received (EtherSpaceLink link)

  returns the total number of bytes read.
- ESL_FN_EXPORT int EtherSpaceLink_write_EXTN (EtherSpaceLink link, int EXTN)

  Sends an extension packet.
- ESL_FN_EXPORT void EtherSpaceLink_read_packet_special_callback (EtherSpaceLink link, void ∗callback_buffer, size_t callback_buffer_length, ESL_CALLBACK callback)

  sets up a handler for special out of band data.
- ESL_FN_EXPORT void EtherSpaceLink_read_packet_extension_callback (EtherSpaceLink link, void ∗callback_buffer, size_t callback_buffer_length, ESL_CALLBACK callback)

  sets up a handler for extension out of band data.
- ESL_FN_EXPORT void EtherSpaceLink_dump_max (EtherSpaceLink link, int dump_max)

Sets the number of message dumps on any one run.

- ESL_FN_EXPORT int EtherSpaceLink_dump (EtherSpaceLink link, char ∗file_name, char ∗note)

  dumps information about the current rx buffer
- ESL_FN_EXPORT int EtherSpaceLink_set_speed (EtherSpaceLink link, int speed)

  Sets the transmit speed of the link
  Sets the transmit speed of all of the SpaceWire links on thisEtherSpaceLink unit.
- ESL_FN_EXPORT int EtherSpaceLink_set_speed_double (EtherSpaceLink link, double speed)

  Sets the transmit speed of the link allowing partial Mb speeds.
- ESL_FN_EXPORT int EtherSpaceLink_set_mode (EtherSpaceLink link, int mode)

  set mode of current link
- ESL_FN_EXPORT int EtherSpaceLink_set_mode_portmask (EtherSpaceLink link, int mode, uint32_t ports)

  set mode of list of links
- ESL_FN_EXPORT ssize_t EtherSpaceLink_get_control_packet (EtherSpaceLink link, void ∗control_buffer, size_t buffer_length, int slot)

  request control data for slot
- ESL_FN_EXPORT int EtherSpaceLink_get_module_type (EtherSpaceLink link, void ∗buffer)

  Return the module type from the given network buffer.
- ESL_FN_EXPORT int EtherSpaceLink_get_HWA (EtherSpaceLink link, unsigned char ∗HWA)

  Reads the hardware address (MAC) for the ESL device.
- ESL_FN_EXPORT double EtherSpaceLink_extract_timetag (EtherSpaceLink link, void ∗buffer)

  Extract timetag from special data callback.
- ESL_FN_EXPORT uint64_t EtherSpaceLink_extract_timetag_ns (EtherSpaceLink link, void ∗buffer)

  Extract timetag from special data callback data in tenths of nano seconds.
- ESL_FN_EXPORT int EtherSpaceLink_TT_enable_reporting (EtherSpaceLink link, int when)

  Enable timetags for currently active link.
- ESL_FN_EXPORT int EtherSpaceLink_ER_enable_reporting (EtherSpaceLink link, int what)

  Enables, or disables, error reporting.
- ESL_FN_EXPORT int EtherSpaceLink_EW_enable_reporting (EtherSpaceLink link, int what)

  Enables, or disables, waveform capture triggers.
  Triggering may be on errors or on other significant events. The parameter what should be set to EtherSpaceLink_EW_capture_nothing to disable all reporting, or to a combination of the.
- ESL_FN_EXPORT int EtherSpaceLink_EW_request_data (EtherSpaceLink link, int port)

  When Error Waveform reporting is switched on, it is possible that the device can indicate that it has an error waveform available via an Extension data block of data. If you want to record this data , you must request it. This function queues the request and the device will then send the error waveform data which is sent as a SPECIAL block of data.
- ESL_FN_EXPORT int EtherSpaceLink_EW_clear (EtherSpaceLink link, int port)

  This function is called to clear waveform data from the port.
- ESL_FN_EXPORT int EtherSpaceLink_EW_reset (EtherSpaceLink link, int port)

  Re-arms the capture of error waveforms having previously captured a waveform.
- ESL_FN_EXPORT int EtherSpaceLink_EI_ignore_events (EtherSpaceLink link, int what)

  Ignore events from the EI module, the EI module will disconnect a link in the event of an error, this function allows the link to ignore errors and continue running in the event of a particular error.
- ESL_FN_EXPORT int EtherSpaceLink_EI_flow_control (EtherSpaceLink link, int initial_fcts, int flow_control)

  Sets Error Injection flow control characteristics.
- ESL_FN_EXPORT int EtherSpaceLink_request_link_status (EtherSpaceLink link)

  Requests the link status.
- ESL_FN_EXPORT int EtherSpaceLink_request_link_status_port (EtherSpaceLink link, int port)

  Requests the link status for a given port On the ESL range of devices this can change the active transmission port.
- ESL_FN_EXPORT int EtherSpaceLink_extract_link_state (EtherSpaceLink link, void ∗buffer, size_t buflen, int ∗port)

  extracts the link state from a message buffer
- ESL_FN_EXPORT int EtherSpaceLink_request_rx_speed (EtherSpaceLink link)

Requests the rx speed.

- ESL_FN_EXPORT double EtherSpaceLink_extract_rx_speed (EtherSpaceLink link, void ∗buffer, size_t buflen, int ∗port)

  Returns the rx speed from a response.
- ESL_FN_EXPORT int EtherSpaceLink_request_tx_speed (EtherSpaceLink link)

  requests the tx speed for the device
- ESL_FN_EXPORT double EtherSpaceLink_extract_tx_speed (EtherSpaceLink link, void ∗buffer, size_t buflen, int ∗port)

  Returns the tx speed from a response.
- ESL_FN_EXPORT int EtherSpaceLink_link_connected (EtherSpaceLink link)

  returns if the currently active link is connected
- ESL_FN_EXPORT int EtherSpaceLink_write_buffer_empty (EtherSpaceLink link)

  Returns if the write buffer for the currently active link is empty.
- ESL_FN_EXPORT int EtherSpaceLink_get_receive_speed (EtherSpaceLink link)

  Gets the receive speed of the currently active link.
- ESL_FN_EXPORT int EtherSpaceLink_get_number_of_links (EtherSpaceLink link)

  Returns the number of links a device has.
- ESL_FN_EXPORT int EtherSpaceLink_set_active_link (EtherSpaceLink link, int n)

  Sets the currently active link.
- ESL_FN_EXPORT int EtherSpaceLink_send_timecode (EtherSpaceLink link, uint8_t tc_)

  Sends a timcode packet on the current link.
- ESL_FN_EXPORT int EtherSpaceLink_set_timecode_transmit (EtherSpaceLink link, int bits, int first, int interval, int report)

  Control the timecode generator.
- ESL_FN_EXPORT int EtherSpaceLink_autonomous_timecode_report (EtherSpaceLink link, void ∗buffer)
- ESL_FN_EXPORT int EtherSpaceLink_set_max_packet_data (EtherSpaceLink link, int N)

  Control the data receive compressor - discard data from packet.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_product_string (EtherSpaceLink link)

  Returns an ASCII string indicating the product name.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_options_string (EtherSpaceLink link)

  Returns an ASCII string indicating the list of options installed on the device.
- ESL_FN_EXPORT void EtherSpaceLink_system_type (EtherSpaceLink link, int type_)

  Forces system type such that platform derived timetags can be read.
- ESL_FN_EXPORT void EtherSpaceLink_set_slot (EtherSpaceLink link, int index, int type_)

  Sets a particular slot to a given slot type (used internally)
- ESL_FN_EXPORT int EtherSpaceLink_get_options (EtherSpaceLink link)

  Returns an integer bitmap indicating the list of options installed on the device.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_manufacturer_string (EtherSpaceLink link)

  Returns manufacturer string.
- ESL_FN_EXPORT ESL_STRING EtherSpaceLink_HWA_to_serial_number_string (unsigned char ∗HWA)

  Converts MAC hardware to human readable string.
- ESL_FN_EXPORT int EtherSpaceLink_fastclose (EtherSpaceLink link)

  Sets the SO_LINGER timeout to 0 such that when the connection is closed, it is closed quickly.
- ESL_FN_EXPORT void EtherSpaceLink_read_stats (EtherSpaceLink link, uint64_t ∗calls, uint64_t ∗io)

  Reads and resets the read call statistics.
- ESL_FN_EXPORT void EtherSpaceLink_dump_status (EtherSpaceLink link)

  Dumps the current link status to stderr.
- ESL_FN_EXPORT int EtherSpaceLink_EW_source (EtherSpaceLink link, int sources)

  Selects waveform capture trigger sources.
  Triggering may be on events from ports other than that associated with the capture circuit.
- ESL_FN_EXPORT int EtherSpaceLink_sma_56_pulse_width (EtherSpaceLink link, int width)

- ESL_FN_EXPORT int EtherSpaceLink_ATI_calibrate (EtherSpaceLink link, int v)
- ESL_FN_EXPORT int EtherSpaceLink_Observe (EtherSpaceLink link, int what)

  request info from the device
- ESL_FN_EXPORT void EtherSpaceLink_set_EINTR (EtherSpaceLink link, int enable)

  sets behaviour when network i/o is interrupted
- ESL_FN_EXPORT int EtherSpaceLink_Non_Blocking (EtherSpaceLink link, int enable_)

  sets behaviour when network i/o is interrupted
- ESL_FN_EXPORT double **decode_fp16** (int fp16)
- ESL_FN_EXPORT void **EtherSpaceLink_set_crash_handler** (ESL_HUP handler_)

6.1.1 Detailed Description

This file is the primary include file used for writing applications using ESL products. (c) 4Links Limited 2000-2019

Unless explicity stated the functions are applicable to our DSI/SLR/MSR products

6.1.2 Macro Definition Documentation

6.1.2.1 #define ESL_CATCH_BEGIN

Value:

```
if (0) { \
esl_error:
```

equivalent of a catch block start

6.1.2.2 #define ESL_TRY(*v*)

Value:

```
if ((v) < 0) {\
esl_print_error();\
goto esl_error; \
}
```

Since we don't have exceptions in C and checking large code blocks for each return value is combersome , we have an emultation of try and catch.

6.1.2.3 #define ESL_TRY_IGNORE(*v, c, err*)

Value:

```
if ((v) < 0) {\
if (EtherSpaceLink_get_error() != err) { esl_print_error(); goto
      esl_error; }\
}
```

try a function and don't "throw" if the error is the given one

6.1.3 Typedef Documentation

6.1.3.1 typedef int(∗ ESL_CALLBACK)(EtherSpaceLink, void ∗, int length, int complete, int data_buffer_position)

Calllback function type for when we have to callback

6.1.3.2 typedef void* EtherSpaceLink

Hidden type for communicating with ESL devices

6.1.4 Function Documentation

6.1.4.1 ESL_FN_EXPORT void EtherSpaceLink_abort (EtherSpaceLink *link*)

Use to abort a connection with a thread on a read call.

Reading packets can block on an underlying network read and if you have a running thread on that read you will have to wait for a packet to arrive such that it can see you want to terminate the thread.

EtherSpaceLink_abort knocks the thread off the read call. Once you have used this call you can't read from the device again.

Parameters

link	connecton to hardware device

```
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}
printf ("Have connection %p\n", esldev);


printf ("Kick off thread doing work ...");
sleep(5);


// want to knock thread off read ....
printf ("Knock off thread doing work ...");
EtherSpaceLink_abort (esldev);

return 0;
```

6.1.4.2 ESL_FN_EXPORT int EtherSpaceLink_ATI_calibrate (EtherSpaceLink *link,* int *v*)

Parameters

link	connection to the device

Returns

< 0 indicating an error

6.1.4.3 ESL_FN_EXPORT int EtherSpaceLink_autonomous_timecode_report (EtherSpaceLink *link,* void * *buffer*)

Bug no check on buffer length

Parameters

link	connection to the device
buffer	containing timecode

Returns

< 0 on error , timecode otherwise

6.1.4.4 ESL_FN_EXPORT void EtherSpaceLink_check_record_writes (EtherSpaceLink *link,* int *on*)

Sets up what to do in the case of an error writing to record file.

This function sets whether the read packet functions return an error when an error occurs writing to the record file.

Generally it is advisable to do this as you may end up with truncated log files

Parameters

link	the handle to the connection
on	non zero means we record errors writing to the log file , the default is 0

```
// connect to host
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

    // can decide to abandon things or try and continue without logging
    esldev = EtherSpaceLink_close(esldev);
    return 1;

}
if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);
    return 1;
}

// if we can't write to the file set an error
EtherSpaceLink_check_record_writes( esldev, 1);
```

6.1.4.5 ESL_FN_EXPORT void* EtherSpaceLink_close (EtherSpaceLink *link*)

Used to close connection to an ESL device or file.

Parameters

link	the connection you want to close down

Returns

NULL to dereference the connection

```
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}
```

```
printf ("Have connection %p\n", esldev);

// lets do work with the device

// finished with device

esldev = EtherSpaceLink_close(esldev);

if (esldev)
{
    printf ("close has not worked\n");
    return 1;
}
return 0;
```

6.1.4.6 ESL_FN_EXPORT int EtherSpaceLink_device_type (EtherSpaceLink *link*)

returns the device type

Parameters

link	the handle on opened filed

Returns

code indicating the device type

```
// connect to host
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}
printf ("Device type = %d\n",  EtherSpaceLink_device_type(esldev));

return 0;
```

6.1.4.7 ESL_FN_EXPORT int EtherSpaceLink_dump (EtherSpaceLink *link*, char * *file_name*, char * *note*)

dumps information about the current rx buffer

Parameters

link	connection to the device
file_name	the name of the file to dump to
note	the prefix associated with the current record

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}


// set number of dumps
EtherSpaceLink_dump_max ( esldev, 5);

EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
```

6.1.4.8 ESL_FN_EXPORT void EtherSpaceLink_dump_max (EtherSpaceLink *link*, int *dump_max*)

Sets the number of message dumps on any one run.

Parameters

link	connection to the device
dump_max	number of dumps to do

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}


// set number of dumps
EtherSpaceLink_dump_max ( esldev, 5);


EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");
EtherSpaceLink_dump ( esldev, "/tmp/log.txt", "dump");

printf ("Done\n");
return 0;
```

6.1.4.9 ESL_FN_EXPORT void EtherSpaceLink_dump_status (EtherSpaceLink *link*)

Dumps the current link status to stderr.

Parameters

calls	number of calls to read_packet_full
io	number of times read packet full does I/O

Returns

char * the product string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

EtherSpaceLink_dump_status(esldev);

return 0;
```

6.1.4.10 ESL_FN_EXPORT int EtherSpaceLink_EI_flow_control (EtherSpaceLink *link,* int *initial_fcts,* int *flow_control*)

Sets Error Injection flow control characteristics.

The ECSS-E-ST-50-12C SpaceWire standard requires at least one flow-control token to be sent to start a link, and the maximum flow-control credit to be 56 N-Chars (as indicated by 7 flow-control tokens). Flow-control tokens are normally issued as space becomes available in the receive buffer. Data tokens (actually N-Chars - data, end-of-packet and error-end-of-packet) may be received up to the number for which credit has been issued. The EtherSpaceLink_EI_flow_control() of the DSI allows the link to be set outside the limits defined by the ECSS SpaceWire standard. An initial flow-control issue of 0 tokens should result in a link not starting. An initial flow-control issue of 8 or more FCTs should cause an error. Automatic issuing of flow-control tokens can be suppressed by the DSI, leaving the user to explicitly issue flow-control tokens (see EtherSpaceLink_write_EXTN). Also, the transmission of data can be allowed by the DSI despite there being no available credit - in order to test a receivers behaviour. The initial_fcts parameter of EtherSpaceLink_EI_flow_control() sets the number of flow-control tokens that are sent when the link starts. This value must be between 0 and 15; otherwise, an error is reported. The flow_control

parameter of EtherSpaceLink_EI_flow_control() may take the value EtherSpaceLink_EI_normal_flow_control, or a combination (logical OR) of one or both of the other values:

Note It will be necessary to enable error reporting for many of these events, using EtherSpaceLink_ER_enable_reporting(), before they will be reported back to the application software.

Parameters

link	connection to the device
initial_fcts	
flow_control	the flow control we want

EtherSpaceLink_EI_normal_flow_control Operate the SpaceWire link using the standard SpaceWire flow-control algorithm.

EtherSpaceLink_EI_transmit_anyway Allow data to be sent regardless of the amount of flow-control credit available.

EtherSpaceLink_EI_no_automatic_FCT Do not automatically send any flowcontrol tokens after the link has started.

Returns

0 on success or transmitted , < 0 if error

```
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }

 EtherSpaceLink_EI_flow_control ( esldev, 0,
      EtherSpaceLink_EI_no_automatic_FCT);

 return 0;
```

6.1.4.11 ESL_FN_EXPORT int EtherSpaceLink_EI_ignore_events (EtherSpaceLink *link,* int *what*)

Ignore events from the EI module, the EI module will disconnect a link in the event of an error, this function allows the link to ignore errors and continue running in the event of a particular error.

Parameters

link	connection to the device
what	events to be ignored from EI module

EtherSpaceLink_EI_ignore_excess_FCT EtherSpaceLink_EI_ignore_excess_data EtherSpaceLink_EI_ignore_parity_error EtherSpaceLink_EI_ignore_ESC_ESC EtherSpaceLink_EI_ignore_ESC_EOP EtherSpaceLink_EI_ignore_ESC_EEP EtherSpaceLink_EI_ignore_timeout

Returns

0 on success or transmitted , < 0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
EtherSpaceLink_EI_ignore_events (esldev, EtherSpaceLink_EI_ignore_ESC_ESC |
       EtherSpaceLink_EI_ignore_ESC_EOP | EtherSpaceLink_EI_ignore_ESC_EEP);

return 0;
```

6.1.4.12 ESL_FN_EXPORT int EtherSpaceLink_ER_enable_reporting (EtherSpaceLink *link,* int *what*)

Enables, or disables, error reporting.

Parameters

link	connection to the device
what	error reporting we wish to enable

Returns

0 on success or transmitted , < 0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

// Only interested in parity errors
EtherSpaceLink_ER_enable_reporting(esldev,
      EtherSpaceLink_ER_report_parity_error);

return 0;
```

6.1.4.13 ESL_FN_EXPORT int EtherSpaceLink_EW_clear (EtherSpaceLink *link*, int *port*)

This function is called to clear waveform data from the port.

Parameters

link	connection to the device
port	the port we want to data from

Returns

!0 on error

```
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }
  unsigned char rxbuf[4096];
  int bytes_received, extension, ii;
  uint32_t flags;
  int active_port;

  bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                     rxbuf,
                                                     sizeof(rxbuf),
                                                     &flags,
                                                     EtherSpaceLink_RETURN_EXTENSION_DATA
                                                     | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                   );
   if (flags == EtherSpaceLink_EXTN)
   {
     printf ("EtherSpaceLink_EXTN %d (%d)\n", flags, bytes_received);
     extension = rxbuf[0] | 256;

     if (extension >= EtherSpaceLink_PortSelect &&
         extension <= EtherSpaceLink_PortSelect_max )
     {
         active_port = extension - EtherSpaceLink_PortSelect;
         printf ("data reception now from port %d:\n", active_port );
     }

     else if (extension == EtherSpaceLink_EVENT)
     {
       printf ("EVENT seen on port %d\n", active_port);
       EtherSpaceLink_EW_request_data ( esldev, active_port );
       EtherSpaceLink_flush ( esldev);
     }
  }
  else
```

```
if (flags == EtherSpaceLink_SPECIAL)
{
     printf ("Event data arruved ...\n");
     if ((rxbuf[3] * 256 + rxbuf [4]) == EtherSpaceLink_EW_address )
     {
           printf( "Got Error Waveform data for port %d - %d bytes\n", rxbuf[2], bytes_received);
           if (EtherSpaceLink_EW_clear (esldev, rxbuf[2]) < 0)
           {
              printf ("failed to clear EW buffer\n");
           }
     }
}

return 0;
```

6.1.4.14 ESL_FN_EXPORT int EtherSpaceLink_EW_enable_reporting (EtherSpaceLink *link*, int *what*)

Enables, or disables, waveform capture triggers.

Triggering may be on errors or on other significant events. The parameter what should be set to EtherSpaceLink_-EW_capture_nothing to disable all reporting, or to a combination of the.

In addition to the given triggers, a (non-maskable) EVENT in the DSI transmit data stream can also trigger a waveform capture. Each port of a DSI has a waveform capture circuit. Each capture circuit can be triggered by events on its own port, and also on other ports and external events. By default, each capture circuit will respond only to its own port. EtherSpaceLink_EW_source() can be used to expand the recognised source of triggers.

Parameters

link	connection to the device
what	error reporting we wish to enable

Returns

0 on success or transmitted , $<$ 0 if error

```
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }

 // Only interested in parity errors
 EtherSpaceLink_EW_enable_reporting(esldev,
      EtherSpaceLink_EW_capture_parity_error);

 return 0;
```

6.1.4.15 ESL_FN_EXPORT int EtherSpaceLink_EW_request_data (EtherSpaceLink *link*, int *port*)

When Error Waveform reporting is switched on, it is possible that the device can indicate that it has an error waveform available via an Extension data block of data. If you want to record this data , you must request it. This function queues the request and the device will then send the error waveform data which is sent as a SPECIAL block of data.

Parameters

link	connection to the device
port	the port we want to data from

Returns

!0 on error

```
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }

  unsigned char rxbuf[4096];
  int bytes_received, extension, ii;
  uint32_t flags;
  int active_port;

  bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                     rxbuf,
                                                     sizeof(rxbuf),
                                                     &flags,
                                                     EtherSpaceLink_RETURN_EXTENSION_DATA
                                                     | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                   );
   if (flags == EtherSpaceLink_EXTN)
   {
     printf ("EtherSpaceLink_EXTN %d (%d)\n", flags, bytes_received);
     extension = rxbuf[0] | 256;

     if (extension >= EtherSpaceLink_PortSelect &&
         extension <= EtherSpaceLink_PortSelect_max )
     {
         active_port = extension - EtherSpaceLink_PortSelect;
         printf ("data reception now from port %d:\n", active_port );
     }

     else if (extension == EtherSpaceLink_EVENT)
     {
       printf ("EVENT seen on port %d\n", active_port);
       EtherSpaceLink_EW_request_data ( esldev, active_port );
       EtherSpaceLink_flush ( esldev);
     }
  }
  else
  if (flags == EtherSpaceLink_SPECIAL)
  {
       printf ("Event data arrived ...\n");

       if ((rxbuf[3] * 256 + rxbuf[4]) == EtherSpaceLink_EW_address )
       {
             printf( "Got Error Waveform data for port %d - %d bytes\n", rxbuf[2], bytes_received);
       }
  }

  return 0;
```

6.1.4.16 ESL_FN_EXPORT int EtherSpaceLink_EW_reset (EtherSpaceLink *link*, int *port*)

Re-arms the capture of error waveforms having previously captured a waveform.

Parameters

link	connection to the device
port	the port number we want to reset

Returns

0 on success or transmitted , $<$ 0 if error

```
*
*
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }
```

```
unsigned char rxbuf[4096];
int bytes_received, extension, ii;
uint32_t flags;
int active_port;

bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                   rxbuf,
                                                   sizeof(rxbuf),
                                                   &flags,
                                                   EtherSpaceLink_RETURN_EXTENSION_DATA
                                                   | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                 );
 if (flags == EtherSpaceLink_EXTN)
 {
   printf ("EtherSpaceLink_EXTN %d (%d)\n", flags, bytes_received);
   extension = rxbuf[0] | 256;

   if (extension >= EtherSpaceLink_PortSelect &&
       extension <= EtherSpaceLink_PortSelect_max )
   {
       active_port = extension - EtherSpaceLink_PortSelect;
       printf ("data reception now from port %d:\n", active_port );
   }

   else if (extension == EtherSpaceLink_EVENT)
   {
     printf ("EVENT seen on port %d\n", active_port);
     EtherSpaceLink_EW_request_data ( esldev, active_port );
     EtherSpaceLink_flush ( esldev);
   }
}
else
if (flags == EtherSpaceLink_SPECIAL)
{
     printf ("Event data arrived ...\n");
     if ((rxbuf[3] * 256 + rxbuf [4]) == EtherSpaceLink_EW_address )
     {
           printf( "Got Error Waveform data for port %d - %d bytes\n", rxbuf[2], bytes_received);
           if (EtherSpaceLink_EW_reset(esldev, rxbuf[2]) < 0)
           {
              printf ("failed to reset EW buffer\n");
           }
     }

}

return 0;
```

6.1.4.17 ESL_FN_EXPORT int EtherSpaceLink_EW_source (EtherSpaceLink *link,* int *sources*)

Selects waveform capture trigger sources.

Triggering may be on events from ports other than that associated with the capture circuit.

For example, waveforms may be captured on all ports for an event occurring on only one of them.

Parameters

link	connection to the device
sources	EtherSpaceLink_EW_Source_port_1 Trigger on events from port 1. EtherSpaceLink_EW_Source_port_2 Trigger on events from port 2. EtherSpaceLink_EW_Source_port_3 Trigger on events from port 3. EtherSpaceLink_EW_Source_port_4 Trigger on events from port 4. EtherSpaceLink_EW_Source_port_5 Trigger on events from port 5. EtherSpaceLink_EW_Source_port_6 Trigger on events from port 6. EtherSpaceLink_EW_Source_port_7 Trigger on events from port 7. EtherSpaceLink_EW_Source_port_8 Trigger on events from port 8.

EtherSpaceLink_EW_Source_SMA_12 -LS, -MS∗ platforms Trigger on a rising edge on SMA connectors 1-2. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_SMA_34 -LS, -MS platforms Trigger on a rising edge on SMA connectors 3-4. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_SMA_56 -LS, -MS platforms Trigger on a rising edge on SMA connectors 5-6. The threshold level is 0.5 V. EtherSpaceLink_EW_Source-_SMA_78 -LS, -MS platforms Trigger on a rising edge on SMA connectors 7-8. The threshold level is 0.5 V. EtherSpaceLink_EW_Source_barrier SO Trigger when the synchronisation barrier is lifted

Returns

0 if the request was successful

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

 // Capture waveform on ALL ports whenever an error occurs on
 // any one (or more)

int all_ports = EtherSpaceLink_EW_Source_port_1
    | EtherSpaceLink_EW_Source_port_2
    | EtherSpaceLink_EW_Source_port_3
    | EtherSpaceLink_EW_Source_port_4
    | EtherSpaceLink_EW_Source_port_5
    | EtherSpaceLink_EW_Source_port_6
    | EtherSpaceLink_EW_Source_port_7
    | EtherSpaceLink_EW_Source_port_8;
int port;
for ( port = 1; port <= 8; port ++ )
{
    EtherSpaceLink_set_active_link ( esldev, port );
    EtherSpaceLink_EW_enable_reporting (esldev,
      EtherSpaceLink_ER_report_running_error );
    EtherSpaceLink_EW_source ( esldev, all_ports );
}

return 0;
```

6.1.4.18 ESL_FN_EXPORT int EtherSpaceLink_extract_link_state (EtherSpaceLink *link*, void ∗ *buffer*, size_t *buflen*, int ∗ *port*)

extracts the link state from a message buffer

Parameters

link	connection to the device
buffer	containing message response
buflen	the message buffer size
port	the port number we have read

Returns

< 0 error, otherwise the port link status

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 1;
}

if ( EtherSpaceLink_request_link_status (esldev) < 0)
{
     printf ( "Unable to request link status\n");
     return 1;
}
unsigned char rxbuf[4096];
int bytes_received, extension, ii;
uint32_t flags;
int active_port;

bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                   rxbuf,
                                                   sizeof(rxbuf),
                                                   &flags,
                                                   EtherSpaceLink_RETURN_EXTENSION_DATA
                                                   | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                 );
if (flags == EtherSpaceLink_EXTN)
{
    int state;
    int port;
    if ((state = EtherSpaceLink_extract_link_state ( esldev, rxbuf,
      bytes_received, & port)) >= 0)
    {
       printf("Link State for port %d is %08x\n", port, state);
       return 0;
    }
}

return 1;
```

6.1.4.19 ESL_FN_EXPORT double EtherSpaceLink_extract_rx_speed (EtherSpaceLink *link*, void * *buffer*, size_t *buflen*, int * *port*)

Returns the rx speed from a response.

Parameters

link	connection to the device
buffer	the buffer containing the message
buflen	the size of the buffer
port	the port number we have received

Returns

0.0 if an error, otherwise speed of link

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
```

```
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_request_rx_speed(esldev))
{
     printf ( "Unable to request rx speed\n");
     return 1;
}

char buffer [1024];
uint32_t flags;

int buflen = EtherSpaceLink_read_packet_full ( esldev, buffer, sizeof (
      buffer), &flags, EtherSpaceLink_RETURN_SPECIAL_DATA| EtherSpaceLink_RETURN_EXTENSION_DATA);

int port;

double speed = EtherSpaceLink_extract_rx_speed( esldev, buffer, buflen, &
      port);

if (speed == 0.0)
{
     printf ( "Unable to extract rx speed\n");
     return 1;
}
printf ("RX speed is %f\n", speed);
return 0;
```

6.1.4.20 ESL_FN_EXPORT double EtherSpaceLink_extract_timetag (EtherSpaceLink *link,* void * *buffer*)

Extract timetag from special data callback.

Returns a timetag from a network buffer, the timetag is returned as tenths of nano seconds since device power on or tenths of nano seconds since the beginning of the year if the device is time synchronized to a GPS source.

Bug no checking on buffer length

Parameters

link	connection to the device
buffer	for the timetag ,the first int field of the buffer may be the module type

Returns

the timetag

```
*

   // Callback start
   int extn_handler(EtherSpaceLink device, void * buffer,  int length, int complete, int
      data_buffer_position )
   {
       double rawtime;
       unsigned char * extn_data_buffer = buffer;
       switch (( 0x100 | extn_data_buffer[0]) )
       {

                 case EtherSpaceLink_TimeTag:
                                          rawtime =
       EtherSpaceLink_extract_timetag(device, extn_data_buffer );
                                          break;
       }
   return 0;
   }
   // Callback end

 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
       printf( "Unable to connect\n");
       return 1;
 }

 char callback_data[256];
```

```
EtherSpaceLink_read_packet_extension_callback ( esldev,
     callback_data, sizeof(callback_data), extn_handler);

// any special packaets will call handler

return 0;
```

6.1.4.21 ESL_FN_EXPORT uint64_t EtherSpaceLink_extract_timetag_ns (EtherSpaceLink *link*, void * *buffer*)

Extract timetag from special data callback data in tenths of nano seconds.

Returns a timetag from a network buffer, the timetag is returned as tenths of nano seconds since device power on or tenths of nano seconds since the beginning of the year if the device is time synchronized to a GPS source.

Bug no checking on buffer length

Parameters

link	connection to the device
buffer	for the timetag ,the first int field of the buffer may be the module type

Returns

the timetag

```
*
   // Callback start
   int extn_handler(EtherSpaceLink device, void * buffer,  int length, int complete, int
      data_buffer_position )
   {
       uint64_t rawtime;
       unsigned char * extn_data_buffer = buffer;
       switch (( 0x100 | extn_data_buffer[0]) )
       {

                 case EtherSpaceLink_TimeTag:
                                          rawtime =
      EtherSpaceLink_extract_timetag_ns(device, extn_data_buffer );
                                          break;
       }
   return 0;
   }
   // Callback end

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char callback_data[256];

EtherSpaceLink_read_packet_extension_callback ( esldev,
     callback_data, sizeof(callback_data), extn_handler);

// any special packaets will call handler

return 0;
```

6.1.4.22 ESL_FN_EXPORT double EtherSpaceLink_extract_tx_speed (EtherSpaceLink *link*, void * *buffer*, size_t *buflen*, int * *port*)

Returns the tx speed from a response.

Parameters

link	connection to the device
buffer	the buffer containing the message
buflen	the size of the buffer
port	the port number we have received

Returns

0.0 if an error, otherwise speed of link

```
// Callback start
int handler(EtherSpaceLink link_, void * buffer, int length_, int complete, int
      data_buffer_position)
{
     printf ("Recieved Special data on %p at offset %d size %d\n", buffer, length_, data_buffer_position);
     unsigned char * special_data_buffer = buffer;
     if (complete)
     {
          int type;
          switch((type = EtherSpaceLink_get_module_type( link_, buffer)))
          {
              case EtherSpaceLink_ram_rw:
                       {
                           if ((length_-1 == 6 ) && (((special_data_buffer[3] << 8) | special_data_buffer[4]
      ) == EtherSpaceLink_TX_SPEED_address))
                           {
                               int link = 0;
                               double tx_speed = EtherSpaceLink_extract_tx_speed
       (link_, special_data_buffer, length_, &link);
                               printf ("TX Speed = %f\n", tx_speed);
                           }
                       }
                       break;

                  default:
                       break;

              }
     }
     return 0;
}
// Callback end
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
char callback_data[256];
EtherSpaceLink_read_packet_special_callback ( esldev,
      callback_data, sizeof(callback_data), handler);

if (EtherSpaceLink_request_tx_speed(esldev))
{
     printf ( "Unable to request rx speed\n");
     return 2;
}


// some time later  doing a read
char buffer [1024];
uint32_t flags;

int buflen = EtherSpaceLink_read_packet_full ( esldev, buffer, sizeof (
      buffer), &flags, EtherSpaceLink_CALLBACK_SPECIAL_DATA| EtherSpaceLink_RETURN_EXTENSION_DATA);

return 0;
```

6.1.4.23 ESL_FN_EXPORT int EtherSpaceLink_fastclose (EtherSpaceLink *link*)

Sets the SO_LINGER timeout to 0 such that when the connection is closed, it is closed quickly.

Parameters

the	link

Returns

0 if successful

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
EtherSpaceLink_fastclose(esldev);

return 0;
```

6.1.4.24 ESL_FN_EXPORT int EtherSpaceLink_File (EtherSpaceLink *link*, ESL_STRING *file*)

starts a connection reading frrom a file

This function allows the connection to switch over to a different file maintaining state and context

Parameters

link	the handle on opened filed
filename	the name of the file

Returns

0 if successful

```
// connect to host
char * filename = "msrrun.txt";
char * nextfile = "msrrun2.txt";

printf ("Reading file %s ...\n", filename);
EtherSpaceLink esldev = EtherSpaceLink_open_flagged(filename,
      EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
     printf("Unable to open  %s error %d\n", filename, EtherSpaceLink_get_error());
     return 0;
}

printf ("Opened recording file ...");


// sometime later after processing first file
if (EtherSpaceLink_File(esldev, nextfile))
{
     printf("Unable to open %s error %d", nextfile,  EtherSpaceLink_get_error());
     return 1;
}

return 0;
```

6.1.4.25 ESL_FN_EXPORT int EtherSpaceLink_flush (EtherSpaceLink *link*)

transmit any buffered data

EtherSpaceLink_write_packet may queue data for transport, this function puts queued data onto the wire

Parameters

link	connection to ESL device
buffer	the buffer to send
length	the length of the buffer to send
flags	indicating how the data to is be treated EtherSpaceLink_EOP the data is to be terminated with an EOP EtherSpaceLink_EEP the data is to be terminated with an EEP EtherSpaceLink_PART_EOP_EEP the data is not yet terminated EtherSpaceLink_INCOMPLETE the data is not yet terminated (but queued in such a way on termination it will be sent in one block)

If you logically OR the flags value with EtherSpaceLink_FLUSH a network flush is perfomed and the data is transmitted, if this is not performed data will be only transmitted when the network buffer is full or the flush method is called

Returns

0 if sucessfull, !0 if not, errno setup and error code in handle

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[60];

int sz = snprintf(msg, sizeof(msg), "Hello world %ld\n", time(NULL));

// send message with an implicit flush of network data
if (EtherSpaceLink_write_packet(esldev, msg, sz,
      EtherSpaceLink_EOP | EtherSpaceLink_FLUSH))
{
    // Error sending information or queuing ...
    printf ("Error Sending\n");
      return 0;
}

printf ("Sent\n");
return 0;
```

6.1.4.26 ESL_FN_EXPORT int EtherSpaceLink_flush_record_file (EtherSpaceLink *link*)

Flush record file.

Flush current recording log file, note that this may cause the application to block whilst data is written

Parameters

link	connection to ESL device

Returns

0 if successful !0 if not

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

  // can decide to abandon things or try and continue without logging
```

```
}
if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);
}


// Do some work


// save the current recording file
EtherSpaceLink_flush_record_file(esldev);
```

6.1.4.27 ESL_FN_EXPORT void* EtherSpaceLink_get_context (EtherSpaceLink *link*)

Return user context associated with a EtherSpaceLink handle.

Parameters

link	connection to ESL device

Returns

the context

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}


typedef struct
{
    char * info;
} MYCTX;

MYCTX * ctx = malloc(sizeof(*ctx));

printf ("Setting context %p on connection %p\n", esldev, ctx);
EtherSpaceLink_set_context ( esldev, ctx);

printf ("Some time later ...\n");

MYCTX * ctx2 = EtherSpaceLink_get_context(esldev);

printf("Context is %p\n", ctx2);

return 0;
```

6.1.4.28 ESL_FN_EXPORT ssize_t EtherSpaceLink_get_control_packet (EtherSpaceLink *link*, void * *control_buffer*, size_t *buffer_length*, int *slot*)

request control data for slot

Sends a request for a special packet for the given slot

Parameters

link	connection to the device
control_buffer	where to save data

buffer_length	the size of the buffer
slot	which slot we are requesting

Returns

< 0 an error occurred, otherwise the size

Bug does not handle error on sending the request

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char buffer[1024];
int sz =  EtherSpaceLink_get_control_packet( esldev, buffer, sizeof(buffer
      ), 1);

return 0;
```

6.1.4.29 ESL_FN_EXPORT int EtherSpaceLink_get_error ( )

Retreive EtherSpaceLink error information.

Parameters

link	the handle to the connection

Returns

int the error number

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

// error situation
int err;
switch ((err = EtherSpaceLink_get_error()))
{
    case 1234:
              printf ("Error 1234 happened\n");
              return 0;
    default:
              printf("Unexpected error %d\n", err);
              return 1;
}
```

6.1.4.30 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_error_text ( )

return textual description of the last error

Parameters

link	connection to ESL device

Returns

ESL_STRING containing the last error text

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[60];

int sz = snprintf(msg, sizeof(msg), "Hello world %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, sz,
      EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
    fprintf(stderr, "Error '%s' (%d) occurred sending data\n",
      EtherSpaceLink_get_error_text(),
      EtherSpaceLink_get_error());
    return 1;
}

if (EtherSpaceLink_flush(esldev))
{
    // Error sending information
    return 1;
}

printf ("Sent\n");
return 0;
```

6.1.4.31 ESL_FN_EXPORT int EtherSpaceLink_get_HWA (EtherSpaceLink *link*, unsigned char * *HWA*)

Reads the hardware address (MAC) for the ESL device.

This function reads the MAC address of the device discarding network traffic.

Parameters

link	connection to the device
HWA	buffer to save the hardware address

Returns

0 if successful, $<$ 0 if not

Bug if the buffer is less than 6 bytes in size, errors may occurr

eats traffic, if you call this whilst analyzing traffic data will be lost

can return random mac address if there is network error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

unsigned char mac[6];
if (EtherSpaceLink_get_HWA ( esldev, mac))
{
     printf ( "Failed to get mac address\n");
     return 1;
}
printf ("MAC %02x%02x%02x%02x%02x%02x\n",
     mac[0],
     mac[1],
```

```
     mac[2],
     mac[3],
     mac[4],
     mac[5]);

return 0;
```

6.1.4.32 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_manufacturer_string (EtherSpaceLink *link*)

Returns manufacturer string.

Parameters

link	connection to the device

Returns

the product string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

printf ("Manufacturer is %s\n", EtherSpaceLink_get_manufacturer_string
      ( esldev));
return 0;
```

6.1.4.33 ESL_FN_EXPORT int EtherSpaceLink_get_module_slot (EtherSpaceLink *link,* int *module*)

Returns the slot a given module resides in.

Return the slot containing the module wth the given id This function is generally considered legacy as all devices post ESL use memory mapped I/O and the concept of modules is generally no longer applicable

Parameters

link	connection to the hardware device
module	the id of the module

Returns

0 if the module is present, otherwise the slot

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

int slot = EtherSpaceLink_get_module_slot(esldev, EtherSpaceLink_CAPABILITIES
      );
if (!slot)
{
    printf ("Device does not contain a capabilities module\n");
    return -1;
}
return 0;
```

6.1.4.34 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_module_string (EtherSpaceLink *link,* int *module*)

Returns the name of a given module.

Parameters

link	connection to the hardware device
module	the id of the module

Returns

0 if the module is not present, otherwise the string of the module

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char * name = EtherSpaceLink_get_module_string(esldev,
      EtherSpaceLink_CAPABILITIES);
if (!name)
{
    printf ("Device does not contain a capabilities module\n");
    return -1;
}
printf ("Capabilities module name is %s\n", name);
return 0;
```

6.1.4.35 ESL_FN_EXPORT int EtherSpaceLink_get_module_type (EtherSpaceLink *link*, void * *buffer*)

Return the module type from the given network buffer.

When data is sent from a module in a special packet, we need to know the type of module originated the message so we can further process the message. This function returns the module type.

Parameters

link	connection to the device
buffer	the buffer to examine

Returns

the module type , $<$ 0 if an error has occurred

```
  // Callback start
  int special_handler(EtherSpaceLink esldev, void * buffer,  int length, int complete, int
      data_buffer_position )
  {
      // check which module is calling us
      switch( EtherSpaceLink_get_module_type( esldev , buffer ) )
      {
          case EtherSpaceLink_ram_rw:     // only supported module
               break;
      }
      return 0;
  }
  // Callback end

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char callback_data[256];

EtherSpaceLink_read_packet_special_callback ( esldev,
      callback_data, sizeof(callback_data), special_handler);

// any special packets will call handler

return 0;
```

6.1.4.36 ESL_FN_EXPORT int EtherSpaceLink_get_number_of_links (EtherSpaceLink *link*)

Returns the number of links a device has.

Note: this function can drop inbound traffic

Parameters

link	connection to the device

Returns

< 0 on error, number of links otherwise

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

int nolinks =  EtherSpaceLink_get_number_of_links (esldev);

if (nolinks < 0)
{
       printf( "Unable to get the number of links\n");
       return 1;
}

printf ("There are %d links attached to the device\n", nolinks);
return 0;
```

6.1.4.37 ESL_FN_EXPORT int EtherSpaceLink_get_options (EtherSpaceLink *link*)

Returns an integer bitmap indicating the list of options installed on the device.

Parameters

link	connection to the device

Returns

int device options bitmap

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

printf ("Product is %08x\n", EtherSpaceLink_get_options( esldev ));

return 0;
```

6.1.4.38 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_options_string (EtherSpaceLink *link*)

Returns an ASCII string indicating the list of options installed on the device.

This string is dynamically allocated and can be returned by the free call

Parameters

link	connection to the device

Returns

char * device options string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

printf ("Product is %s\n", EtherSpaceLink_get_options_string( esldev ));

return 0;
```

6.1.4.39 ESL_FN_EXPORT ssize_t EtherSpaceLink_get_packet (EtherSpaceLink *link*, void * *buffer*, ssize_t *offset*, ssize_t *buffer_length*, int *sda*)

read packet not returning packet type.

This function is similar to that of read_packet_full, however the rx_flags parameter is not present and as such can be retrieved by calling EtherSpaceLink_get_rx_flags

Parameters

link	connection to ESL device
buffer	where to read data into
offset	offset into the above buffer (i.e. data written to buffer+offset)
buffer_length	the number of bytes to read
sda	how to treat special_actions lower 4 bits enumerate to EtherSpaceLink_DISCARD_SPECIAL_DATA ignores special data EtherSpaceLink_REPORT_SPECIAL_DATA returns special data as -ve return value EtherSpaceLink_RETURN_SPECIAL_DATA returns data as normal message EtherSpaceLink_CALLBACK_SPECIAL_DATA calls callback upper 4 bits enumberate to EtherSpaceLink_DISCARD_EXTENSION_DATA (0) ignores extension data EtherSpaceLink_REPORT_EXTENSION_DATA returns extension data as -ve return value EtherSpaceLink_RETURN_EXTENSION_DATA returns data as normal message EtherSpaceLink_CALLBACK_EXTENSION_DATA calls callback

Returns

< 0 error code

number of bytes the in the message

```
*
 // connect to host on port 1234
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
      return 1;
 }

 EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

 // build message
```

```
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
     EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}
EtherSpaceLink_flush ( esldev );

printf ("Message sent\n");

char rx_buffer[1024];
int loop_index;
uint32_t flags;
int bytes_received = EtherSpaceLink_get_packet ( esldev, rx_buffer, 0, sizeof (
     rx_buffer), EtherSpaceLink_DISCARD_SPECIAL_DATA| EtherSpaceLink_DISCARD_EXTENSION_DATA);
printf ( "Received packet of length %i: ", bytes_received );

for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
    printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}

// Record what kind of packet it was
printf ( " %s\n", EtherSpaceLink_get_rx_flags (esldev) ==
     EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
     EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

6.1.4.40 ESL_FN_EXPORT double EtherSpaceLink_get_percent_file_read (EtherSpaceLink *link*)

returns how much data has been read from the file

Parameters

link	the handle on opened filed

Returns

% file opened

```
// read the file
char * filename = "msrrun.txt";

printf ("Reading file %s ...\n", filename);
EtherSpaceLink esldev = EtherSpaceLink_open_flagged(filename,
     EtherSpaceLink_CONNECT_FILE,0);
if (!esldev)
{
     printf("Unable to open  %s error %d:%d\n", filename,
      EtherSpaceLink_get_error (), errno);
     return 0;
}

printf ("Opened recording file ...");
char rx_buffer[1024];
uint32_t flags;
ssize_t sz;
while ((sz = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (rx_buffer),
     &flags)) > 0)
{
   printf ( "Received packet of length %d %f complete",(int) sz,
      EtherSpaceLink_get_percent_file_read ( esldev ));
}

EtherSpaceLink_close(esldev);
return 0;
```

6.1.4.41 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_product_string (EtherSpaceLink *link*)

Returns an ASCII string indicating the product name.

This string is extracted from the Unicode string actually returned by the unit, by extracting the least significant 8-bits of each character.

This string is dynamically allocated and can be returned by the free call

Parameters

link	connection to the device

Returns

char * product string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

printf ("Product is %s\n", EtherSpaceLink_get_product_string( esldev ));

return 0;
```

6.1.4.42 ESL_FN_EXPORT int EtherSpaceLink_get_receive_speed (EtherSpaceLink *link*)

Gets the receive speed of the currently active link.

Note that this function has the ability to cause frames to be dropped and EtherSpaceLink_request_rx_speed should be used instead

Parameters

link	connection to the device

Returns

< 0 if an error other gets the speed

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 2;
}

int rxspeed = EtherSpaceLink_get_receive_speed (esldev);
if (rxspeed < 0)
{
     printf ( "Error getting receive speed\n");
     return 3;
}

printf ("Receive speed is %d\n", rxspeed);
return 0;
```

6.1.4.43 ESL_FN_EXPORT ESL_BIN_FILE_WR EtherSpaceLink_get_record_file (EtherSpaceLink *link*)

Retreive current recording file.

Parameters

link	connection to ESL device

Returns

handle on recording file

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

  // can decide to abandon things or try and continue without logging

}
if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);

}

// get the recording file
FILE * rf =  EtherSpaceLink_get_record_file(esldev);

// essentially a FILE * on /traces/run.esl
```

6.1.4.44 ESL_FN_EXPORT uint64_t EtherSpaceLink_get_record_size (EtherSpaceLink *link*)

Return the amount of data written to the current recording file.

Parameters

link	the connection that is recording to the record file

Returns

uint64_t the number of bytes written to the file

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);

}
// lets do work with the device


printf ("We have recorded %d bytes\n", (int) EtherSpaceLink_get_record_size(
      esldev));


// finished with device
esldev = EtherSpaceLink_close(esldev);

return 0;
```

6.1.4.45 ESL_FN_EXPORT uint32_t EtherSpaceLink_get_rx_flags (EtherSpaceLink *link*)

return message flags of last packet data read

Sometimes it may be necessary to obtain the message flags outside of the read call

This function returns the value of the flags performed by the last read call.

Parameters

link	connection to ESL device

```
  // connect to host on port 1234
  EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent(
      EtherSpaceLink link);
  if (!esldev)
  {
       printf( "Unable to connect\n");
       return 1;
  }

  EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

  // build message
  char msg[1024];

  int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

  // send message
  if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
  {
      // Error sending information or queuing ...
       printf("Unable to send\n");
       return 1;
  }

  // get the data on the wire
  if (EtherSpaceLink_flush ( esldev ))
  {
       printf("Unable to clear buffers\n");
       return 2;
  }

   char rx_buffer[1024];
   uint32_t flags;

   int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
   printf ( "Received packet of length %i: with flags %08x:%08x\n ", bytes_received, flags,
      EtherSpaceLink_get_rx_flags(esldev));

   EtherSpaceLink_close ( esldev );
   return 0;
```

6.1.4.46 ESL_FN_EXPORT int EtherSpaceLink_get_rx_timeout (EtherSpaceLink *link*)

Retreive the current recieve timeout (milliseconds)

Parameters

link	connection to ESL device

Returns

integer, the current receive timeout in milliseconds

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}
```

```
int rxt  = EtherSpaceLink_get_rx_timeout(esldev);

printf("Current receieve timeout is %d\n", rxt);

return 0;
```

6.1.4.47 ESL_FN_EXPORT int EtherSpaceLink_get_slot (EtherSpaceLink *link,* int *slot*)

Returns the module in a given slot.

Parameters

link	connection to the hardware device
slot	number

Returns

id of the module of the slot

```
// connect to host on
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

int slot = EtherSpaceLink_get_slot(esldev, 1);
printf ("Contents of slot 1 is %d\n", slot);
return 0;
```

6.1.4.48 ESL_FN_EXPORT SKT EtherSpaceLink_get_socket (EtherSpaceLink *link*)

Set EtherSpaceLink error information for application use.

Parameters

link	the handle to the connection
error	number to set

Returns

int the error number just set

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (esldev)
{
     printf( "Unable to connect %d\n", EtherSpaceLink_get_error());
     return 1;
 }
 return 0;
```

Retreive the underlying transport to the ESL hardware

Note, the underlying transport may change with hardware and software releases.

Parameters

link	the handle to the connection

Returns

SKT, the connection to the device

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

SKT skt = EtherSpaceLink_get_socket(esldev);
printf ("Socket is %d\n", skt);
return 0;
```

6.1.4.49 ESL_FN_EXPORT uint64_t EtherSpaceLink_get_total_raw_bytes_received (EtherSpaceLink *link*)

returns the total number of bytes read.

Parameters

link	the handle to the connection

Returns

uint64_t, the total number of bytes recieved

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

uint64_t sz = EtherSpaceLink_get_total_raw_bytes_received (
      esldev);
printf ("%zu bytes received\n", (size_t) sz);
return 0;
```

6.1.4.50 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_get_version ( )

Used to get the version number of the C API being used.

Note the first character denotes the release status of the API if it starts with an 'r' then it as an officially supported version 'norelease' then it is an internal not for release

The macro EtherSpaceLink_version returns the version of the interface

Returns

ESL_STRING API version number

```
char * version = EtherSpaceLink_get_version ();
printf ("API version is %s\n", version);
char * hversion = EtherSpaceLink_version;
printf ("Header version is %s\n", version);
return 0;
```

6.1.4.51 ESL_FN_EXPORT ESL_STRING EtherSpaceLink_HWA_to_serial_number_string (unsigned char * *HWA*)

Converts MAC hardware to human readable string.

Parameters

HWA	buffer to hardware buffer

Returns

char * the product string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

unsigned char mac[6];
if (EtherSpaceLink_get_HWA ( esldev, mac))
{
     printf ( "Failed to get mac address\n");
     return 1;
}

printf ("Serial number is %s\n", EtherSpaceLink_HWA_to_serial_number_string
      (mac));

return 0;
```

6.1.4.52 ESL_FN_EXPORT int EtherSpaceLink_link_connected (EtherSpaceLink *link*)

returns if the currently active link is connected

Parameters

link	connection to the device

Returns

0 not connected, < 0 if error, 1 connected

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 1;
}

int con;
switch ((con = EtherSpaceLink_link_connected ( esldev )))
{
    case 0:        // Not Connected
        printf ("Not Connected\n");
        break;

    case 1:        // Connected
        printf ("Connected\n");
        break;

    cdefault:
        printf ("Error %d\n", con);
              return 2;
        break;

    default:    // Unknown return 1;
        printf ("Unkown return %d\n", con);
              return 3;
        break;
}

return 0;
```

6.1.4.53 ESL_FN_EXPORT int EtherSpaceLink_Non_Blocking (EtherSpaceLink *link*, int *enable_*)

sets behaviour when network i/o is interrupted

Some applications require sockets to be non blocking and I/O calls to return if they would block. This call allows the application to run in non blocking mode for read and write calls.

Parameters

link	handle
enable	enable non blocking mode
0	if successful, error code otherwise

```
unsigned char rxbuf[4096];
int bytes_received, extension, ii;
uint32_t flags;
int active_port;

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
      printf( "Unable to connect\n");
      return 1;
}

// set Non Blocking
if ( EtherSpaceLink_Non_Blocking(esldev , 1) < 0)
{
      printf( "Unable to go non blocking\n");
      return 1;
}

bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                   rxbuf,
                                                   sizeof(rxbuf),
                                                   &flags,
                                                   EtherSpaceLink_RETURN_EXTENSION_DATA
                                                   | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                 );

 if (bytes_received < 0)
 {
     if ( bytes_received == EtherSpaceLink_Error_Would_Block)
     {
         printf ("I/O call would block\n");
         return 1;
     }
 }

 return 0;
```

6.1.4.54 ESL_FN_EXPORT int EtherSpaceLink_Observe (EtherSpaceLink *link*, int *what*)

request info from the device

Parameters

link	connection to the device
address	the address
bytes	the number of bytes to read

Returns

< 0 if an error has been sent

6.1.4.55 ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open (char * *address*)

opens a connection to the specified device

Opens a connection the the etherspace link device specified which may be resolvable hostname or an ipv4 address.

A port number can be specified by adding a suffix with :portnumber. For example, 1.2.3.4:9999 will connect to a device at IP 1.2.3.4 with port number 9999

It also reads the table of modules installed in the EtherSpaceLink to an internal buffer, for use by procedures accessing status and module information. When opened, the SpaceWire link will be in the disabled state and its default speed will be 10Mb/s. Module and link parameters can be set immediately but the link must be started (using EtherSpaceLink_set_mode()) before data can be transferred over the SpaceWire link.

IPV6 is currently not supported by our devices

Note on the first call to this function we set the SIG_PIPE handler to SIG_IGN.

Parameters

address	The address / address:port specifier

Returns

EtherSpaceLink NULL if there was an error otherwise a EtherSpaceLink Handle

```
// connect to host
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}

printf ("Connected to %s with handle %p\n", host, esldev);

return 0;
```

6.1.4.56 ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_as_server (SKT *sock*)

cCreate an Etherspace connection using a socket

Bug For future transports this function may be obsoleted

Parameters

sock	the socket to use as the connection

Returns

EtherSpaceLink handle, null if failed

```
SKT skt = get_socket();

struct hostent    *hp;
struct sockaddr_in server;
int                port    = 0x1355;
char              *address = "1.1.1.1";
char              *c;

hp = gethostbyname( address );
free(address);

if (hp == 0 )
{
   return 0;
}

server.sin_family = AF_INET;
memcpy( (void*)(&server.sin_addr), hp->h_addr, hp->h_length );
server.sin_port = htons( (unsigned short int)port );

if ( connect (skt, (struct sockaddr *)&server, sizeof(server) ) < 0 )
{
```

```
    return 1;
}


EtherSpaceLink esldev = EtherSpaceLink_open_as_server ( skt );

if (esldev)
{
   printf ("Connected\n");
   return 0;
}
```

6.1.4.57 ESL_FN_EXPORT EtherSpaceLink EtherSpaceLink_open_flagged (char * *filename*, uint32_t *flags_*, int *timeout*)

opens a connection to a device or file

This function establishes an ESL connection to a device or a file. It is different from the open call in that it can be used to open a file, and in the case of an ESL device a connect timeout may be given.

Parameters

filename	the name of the file/hostname
*flags_*	operations for the connection EtherSpaceLink_CONNECT_FILE connecting to a file EtherSpaceLink_CONNECT_TIMEOUT connect with a timeout EtherSpaceLink_CONNECT_RX_TIMEOUT set initial timeout

Returns

EtherSpaceLink NULL if there was an error otherwise a EtherSpaceLink Handle

```
// connect to host
char * filename = "msrrun.txt";
int ec = 0;

printf ("Reading file %s ...\n", filename);
EtherSpaceLink esldev = EtherSpaceLink_open_flagged(filename,
      EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
     printf("Unable to open  %s error %d\n", filename, EtherSpaceLink_get_error());
     return 0;
}

printf ("Opened recording file ...");

return 0;
```

6.1.4.58 ESL_FN_EXPORT ssize_t EtherSpaceLink_Query_Sent (EtherSpaceLink *link*)

reads a spacewire packet of data

Receives data from the SpaceWire link via the EtherSpaceLink unit. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

Extension sequences and Special packets are never returned by this interface.

Use EtherSpaceLink_read_packet_full if you wish to receive special packets or extension sequences.

Parameters

link	connection to ESL device
buffer	the target destination for the data we want to read
buffer_length	the size of the buffer

rx_flags	pointer to metadata about the frame EtherSpaceLink_EOP This is the last part, or all, of a packet; an end-of-packet (EOP) was received. EtherSpaceLink_EEP This is the last part, or all, of a packet containing an error; an error end-of-packet EtherSpaceLink_PART_EOP_EEP The packet was larger than the available buffer. Another read will retrieve more data f

Returns

<0 error, number of bytes the in the message

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

// build message
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}

printf ("Message sent\n");

 char rx_buffer[1024];
 uint32_t flags;
 int loop_index;

 int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
 printf ( "Received packet of length %i: ", bytes_received );
 for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
 {
     printf ( " #%02x", rx_buffer[loop_index] & 0xff );
 }
 printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
      EtherSpaceLink_EOP ? "EOP" : "EEP");

 EtherSpaceLink_close ( esldev );
 return 0;
```

6.1.4.59 ESL_FN_EXPORT void EtherSpaceLink_read_packet_extension_callback (EtherSpaceLink *link*, void * *callback_buffer*, size_t *callback_buffer_length*, ESL_CALLBACK *callback*)

sets up a handler for extension out of band data.

Registers a buffer and a callback procedure that may be called, if enabled, when an extension packet is received in the middle of reception of a normal data packet. This permits the asynchronous handling of special packets without the need for a normal data read to accept and process such packets*

Parameters

link	the handle to the connection
callback_buffer	for callback data
callback_buffer_length	length of callback buffer
callback	function to call

Bug if buffer goes out of scope program may crash

```
// Callback start
int extension_handler(EtherSpaceLink link_, void * buffer, int length, int complete, int
      data_buffer_position)
{
     printf ("Recieved Special data on %p at offset %d size %d\n", buffer, length, data_buffer_position);
     if (complete)
     {
         printf ("Complete\n");
     }
     return 0;
}
// Callback end

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char callback_data[256];

EtherSpaceLink_read_packet_extension_callback ( esldev,
      callback_data, sizeof(callback_data), extension_handler);

return 0;
```

6.1.4.60 ESL_FN_EXPORT ssize_t EtherSpaceLink_read_packet_full (EtherSpaceLink *link*, void * *buffer*, size_t *buffer_length*, uint32_t * *rx_flags*, int *special_data_action*)

reads a packet handling out of band information

Receives data from the SpaceWire link via the EtherSpaceLink unit with option to include special and extension data frames. The buffer may contain all, or part, of a packet, as indicated by the final parameter, flags.

Parameters

link	connection to ESL device
buffer	where to read data into
buffer_length	the number of bytes to read
rx_flags	message flags EtherSpaceLink_EOP This is the last part, or all, of a packet; an end-of-packet (EOP) was received. EtherSpaceLink_EEP This is the last part, or all, of a packet containing an error; an error end-of-packet EtherSpaceLink_PART_EOP_EEP The packet was larger than the available buffer. Another read will retrieve more data f EtherSpaceLink_EXTN Data frame is an extended frame EtherSpaceLink_PART_EXTN Data frame is an extended frame but is as yet incomplete and the functiion should be ca EtherSpaceLink_SPECIAL Data frame is an special frame EtherSpaceLink_PART_EXTN Data frame is an special frame but is as yet incomplete and the functiion should be cal EtherSpaceLink_SPECIAL_SIZE When called with REPORT_SPECIAL_SIZE indicates how much special data follows EtherSpaceLink_EXTENSION_SIZE When called with REPORT_SPECIAL_SIZE indicates how much extension data follows
special_data_action	how to treat special_actions lower 4 bits enumerate to EtherSpaceLink_DISCARD_SPECIAL_DATA ignores special data EtherSpaceLink_REPORT_SPECIAL_DATA returns special data size (and flags set to EtherSpaceLink_ EtherSpaceLink_RETURN_SPECIAL_DATA returns data as normal message EtherSpaceLink_CALLBACK_SPECIAL_DATA calls callback upper 4 bits enumberate to EtherSpaceLink_DISCARD_EXTENSION_DATA (0) ignores extension data EtherSpaceLink_REPORT_EXTENSION_DATA returns extension data size (and flags set to EtherSpaceLin EtherSpaceLink_RETURN_EXTENSION_DATA returns data as normal message EtherSpaceLink_CALLBACK_EXTENSION_DATA calls callback

Returns

< 0 -ve value of error number

number of bytes the in the message

```
*
 // connect to host on port 1234
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
      return 1;
 }

 EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );

 // build message
 char msg[1024];

 int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

 // send message
```

```
if (EtherSpaceLink_write_packet(esldev, msg, size,
     EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}
EtherSpaceLink_flush ( esldev );

printf ("Message sent\n");

char rx_buffer[1024];
int loop_index;
uint32_t flags;
int bytes_received = EtherSpaceLink_read_packet_full ( esldev, rx_buffer,
     sizeof (rx_buffer), &flags, EtherSpaceLink_DISCARD_SPECIAL_DATA| EtherSpaceLink_DISCARD_EXTENSION_DATA);
printf ( "Received packet of length %i: ", bytes_received );

for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
    printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}
printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
     EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

6.1.4.61 ESL_FN_EXPORT void EtherSpaceLink_read_packet_special_callback (EtherSpaceLink *link*, void * *callback_buffer*, size_t *callback_buffer_length*, ESL_CALLBACK *callback*)

sets up a handler for special out of band data.

Registers a buffer and a callback procedure that may be called, if enabled, when a special packet is received in the middle of reception of a normal data packet. This permits the asynchronous handling of special packets without the need for a normal data read to accept and process such packets*

Parameters

link	the handle to the connection
callback_buffer	for callback data
callback_buffer_length	length of callback buffer
callback	function to call

Bug if buffer goes out of scope program may crash

```
// Callback start
int handler(EtherSpaceLink link_, void * buffer, int length, int complete, int
     data_buffer_position)
{
     printf ("Recieved Special data on %p at offset %d size %d\n", buffer, length, data_buffer_position);
     if (complete)
     {
         printf ("Complete\n");
     }
     return 0;
}
// Callback end

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char callback_data[256];

EtherSpaceLink_read_packet_special_callback ( esldev,
```

```
      callback_data, sizeof(callback_data), handler);

// any special packaets will call habdler

return 0;
```

6.1.4.62 ESL_FN_EXPORT void EtherSpaceLink_read_stats (EtherSpaceLink *link*, uint64_t * *calls*, uint64_t * *io*)

Reads and resets the read call statistics.

Parameters

link	connection to the device
calls	number of calls to read_packet_full
io	number of times read packet full does I/O

Returns

char * the product string

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

uint64_t calls;
uint64_t io;

EtherSpaceLink_read_stats(esldev, &calls, &io);

return 0;
```

6.1.4.63 ESL_FN_EXPORT int EtherSpaceLink_record_writes (EtherSpaceLink *link*)

Returns whether errors writing to record file are treated as errors.

Parameters

link	the handle to the connection

Returns

integer, 0 errors writing to file are ignored, !0 if they are acted on

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

  // can decide to abandon things or try and continue without logging

}
if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);

}

// if we can't write to the file set an error
EtherSpaceLink_check_record_writes( esldev, 1);
```

```
int re = EtherSpaceLink_record_writes(esldev);

printf ("Errors are recorded : %s\n", re?"true":"false");
```

6.1.4.64 ESL_FN_EXPORT int EtherSpaceLink_request_link_status (EtherSpaceLink *link*)

Requests the link status.

Parameters

link	connection to the device

Returns

< 0 if an error, otherwise queued for transmission

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 1;
}

if ( EtherSpaceLink_request_link_status (esldev) < 0)
{
     printf ( "Unable to request link status\n");
     return 1;
}

return 0;
```

6.1.4.65 ESL_FN_EXPORT int EtherSpaceLink_request_link_status_port (EtherSpaceLink *link*, int *port*)

Requests the link status for a given port On the ESL range of devices this can change the active transmission port.

Parameters

link	connection to the device
port	which port you want status on

Returns

< 0 if an error, otherwise queued for transmission

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}
if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 1;
}

if ( EtherSpaceLink_request_link_status_port (esldev, 1) < 0)
{
     printf ( "Unable to request link status\n");
     return 1;
}

return 0;
```

6.1.4.66 ESL_FN_EXPORT int EtherSpaceLink_request_rx_speed (EtherSpaceLink *link*)

Requests the rx speed.

Parameters

link	connection to the device

Returns

< 0 if an error, otherwise queued for transmission

```
// Callback start
int handler(EtherSpaceLink link_, void * buffer, int length_, int complete, int
      data_buffer_position)
{
     printf ("Recieved Special data on %p at offset %d size %d\n", buffer, length_, data_buffer_position);
     unsigned char * special_data_buffer = buffer;
     if (complete)
     {
          int type;
          switch((type = EtherSpaceLink_get_module_type( link_ , buffer)))
          {
              case EtherSpaceLink_ram_rw:
                       {
                           if ((length_-1 == 8) && (((special_data_buffer[3] << 8) | special_data_buffer[4])
       == EtherSpaceLink_LINK_address))
                           {
                               int link = 0;
                               double rx_speed = EtherSpaceLink_extract_rx_speed
      (link_, special_data_buffer, length_, &link);
                               printf ("RX SPEED = %f\n", rx_speed);
                           }
                       }
                       break;
              }
     }
     exit(0);
     return 0;
}
// Callback end

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

char callback_data[256];
EtherSpaceLink_read_packet_special_callback ( esldev,
      callback_data, sizeof(callback_data), handler);

if ((EtherSpaceLink_request_rx_speed(esldev)) < 0)
{
     printf ( "Unable to request rx speed\n");
     return 1;
}

char buffer [1024];
uint32_t flags;

int buflen = EtherSpaceLink_read_packet_full ( esldev, buffer, sizeof (
      buffer), &flags, EtherSpaceLink_CALLBACK_SPECIAL_DATA| EtherSpaceLink_RETURN_EXTENSION_DATA);

return 0;
```

6.1.4.67 ESL_FN_EXPORT int EtherSpaceLink_request_tx_speed (EtherSpaceLink *link*)

requests the tx speed for the device

Parameters

link	connection to the device

Returns

< 0 if an error, otherwise queued for transmission

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
```

```
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_request_tx_speed(esldev))
{
     printf ( "Unable to request tx speed");
     return 2;
}
return 0;
```

6.1.4.68 ESL_FN_EXPORT ssize_t EtherSpaceLink_send (EtherSpaceLink *link,* void * *buffer,* size_t *size,* int *flags*)

Low level call to send data on ESL transport.

This is a low level call which sends some data on an ESL transport, handling EINTR and EAGAIN. It is really applicable if you are sending raw data to the socket

Parameters

link	connection to ESL device
buffer	the data to send
size	the number of bytes to send
flags	transport flags

Returns

number of bytes sent, < 0 if error (-ve error code)

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_send ( esldev, "abcd", 4, 0) != 4)
{
   printf ("Error sending data %d\n", errno);
   return 2;
}

printf ("Sent data\n");
return 0;
```

6.1.4.69 ESL_FN_EXPORT int EtherSpaceLink_send_timecode (EtherSpaceLink *link,* uint8_t *tc_*)

Sends a timcode packet on the current link.

Parameters

link	connection to the device
n	the link we want to make active

Returns

< 0 error, 0 success

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
```

```
{
       printf( "Unable to set active link\n");
       return 2;
}
if (EtherSpaceLink_send_timecode( esldev, 123))
{
       printf( "Unable to send timecode\n");
       return 2;
}

printf("Made link 1 active\n");
return 0;
```

6.1.4.70 ESL_FN_EXPORT int EtherSpaceLink_set_active_link (EtherSpaceLink *link*, int *n*)

Sets the currently active link.

Parameters

link	connection to the device
n	the link we want to make active

Returns

< 0 error, 0 success

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

if (EtherSpaceLink_set_active_link( esldev, 1))
{
       printf( "Unable to set active link\n");
       return 2;
}

printf("Made link 1 active\n");
return 0;
```

6.1.4.71 ESL_FN_EXPORT void EtherSpaceLink_set_context (EtherSpaceLink *link*, void * *context*)

Set user context can be associated with a EtherSpaceLink handle.

Parameters

link	connection to ESL device
context	arbitrary user defined context / structure

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

typedef struct
{
    char * info;
} MYCTX;

MYCTX * ctx = malloc(sizeof(*ctx));

printf ("Setting context %p on connection %p\n", esldev, ctx);
EtherSpaceLink_set_context ( esldev, ctx);

return 0;
```

6.1.4.72 ESL_FN_EXPORT void EtherSpaceLink_set_debug (int *level,* int *output*)

sets debug level for debug library

We supply a tracing library for debug purposes, in order to enable debug call this function when linking against the trace library. This function sets the debug level and writes output to the given file descriptor. If the file descriptor is opened with O_APPEND records written will be atomic.

Note you don't have to call this function explicitly , if you set the following environment variables it will have the same effect.

```
ESL_DEBUG_FILE the name of the debug file
ESL_DEBUG_LEVEL the debug level
```

On the production library this call will do nothing

Parameters

level	the debug level of the library 1 debug function calls 2 debug internal flow within functions 4 debug error states 8 debug data
output	the file descriptor

```
int debug = open("/tmp/debug.txt", O_APPEND|O_CREAT|O_WRONLY|O_TRUNC, 0666);
EtherSpaceLink_set_debug(1|2|4, debug);

char * host = "1.1.1.1:1234";
EtherSpaceLink esldev = EtherSpaceLink_open(host);

return 0;
```

6.1.4.73 ESL_FN_EXPORT void EtherSpaceLink_set_EINTR (EtherSpaceLink *link,* int *enable*)

sets behaviour when network i/o is interrupted

When the API is reading data, it may be using a blocking read call. Normally it soaks up EINTR and EAGAIN calls when reading data, thus if a signal is received the application may not be able to handle any action set up by the signal handler.

Setting the EINTR flag means that an application on a blocking read will return and the error code will be EtherSpaceLink_Error_EINTR

Parameters

link	handle
enable	returning EINTR on read

```
unsigned char rxbuf[4096];
int bytes_received, extension, ii;
uint32_t flags;
int active_port;

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
      printf( "Unable to connect\n");
      return 1;
}

// set EINTR
EtherSpaceLink_set_EINTR(esldev , 1);

bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                                   rxbuf,
                                                   sizeof(rxbuf),
                                                   &flags,
                                                   EtherSpaceLink_RETURN_EXTENSION_DATA
                                                   | EtherSpaceLink_RETURN_SPECIAL_DATA
                                                 );

 if (bytes_received < 0)
```

```
{
    if ( bytes_received == EtherSpaceLink_Error_EINTR)
    {
        printf ("something caused an EINTR ... check signal handlers\n");
    }
}

return 0;
```

6.1.4.74 ESL_FN_EXPORT int EtherSpaceLink_set_log_file (EtherSpaceLink *link*, char * *file_name*)

Sets the current log file.

Creates a log file which will contain a record of data transmitted and received. Data from this file can be later analysed and/or replayed.

If there is an extant log file in then it closed.

if the file name is not null then it is used for logging

Entries in the log file will begin with the text: Rx@p for data received on port p Tx@p for data transmitted on port p

Alternatively, if a SpaceWire link id has been set for a particular EtherSpaceLink unit:

Rx@id@p for data received on port p Tx@id@p for data transmitted on port p

Parameters

link	the connection you want to log
file_name	the name of the logging file

Returns

int, 0 if successful , non zero if not

in the event of failure sets the error to EtherSpaceLink_Error_LogFile_Open passing a null pointer as the file name is treated as an error

Bug errno may not be preserved

Bug in the event of an error may pollute FD 1

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

  // can decide to abandon things or try and continue without logging

}

// lets do work with the device

int some_really_bad_error = 0;

//
if (some_really_bad_error)
{
     esldev = EtherSpaceLink_shutdown(esldev);
     return 2;
}

// finished with device
esldev = EtherSpaceLink_close(esldev);
```

6.1.4.75 ESL_FN_EXPORT int EtherSpaceLink_set_max_packet_data (EtherSpaceLink *link,* int *N*)

Control the data receive compressor - discard data from packet.

Parameters

link	the connection to the hardware device
N	the number of bytes to be forwarded N = 0 to 254 allow N bytes to be forwarded N = 255 allow all bytes to be forwarded return 0 if sucessful and !0 in error situation

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       fprintf(stderr, "Unable to connect\n");
       return 1;
}

// Only interested in 128 bytes of frames
EtherSpaceLink_set_max_packet_data (esldev, 128);

return 0;
```

6.1.4.76 ESL_FN_EXPORT int EtherSpaceLink_set_mode (EtherSpaceLink *link*, int *mode*)

set mode of current link

Sets the operating mode of the currently-active SpaceWire link.

After opening a connection, the link is disabled; it must then be enabled into one of its operational modes before data can be transferred.

One of the three modes EtherSpaceLink_LINK_mode_disabled, EtherSpaceLink_LINK_mode_normal and EtherSpaceLink_LINK_mode_legacy should be chosen.

The use of EtherSpaceLink_LINK_mode_fixed_speed to set some DSI ports to 10Mb/s, together with the conventional EtherSpaceLink_set_speed() mechanism is the only way to run a DSIs links at two different speeds

Parameters

link	connection to the device
mode	of operation

EtherSpaceLink_LINK_mode_disabled

```
 The link is idle and silent.
```

EtherSpaceLink_LINK_mode_normal

```
Start the link by actively trying to establish contact.
```

EtherSpaceLink_LINK_mode_legacy

```
Dont start until activity on the link is seen. Use with SMCS/TSS901 devices.
```

EtherSpaceLink_LINK_mode_long_timeout

```
       Extends the timeout period in the link state  machine to provide a potentialy more reliable link
       start at very low data rates (i.e. for slow (lowpower) links near to 2Mb/s). It is necessary to set
       the link speed with an   EtherSpaceLink_set_speed() API call before calling EtherSpaceLink_set_mode()
       EtherSpaceLink_LINK_mode_slow_speed.


   EtherSpaceLink_LINK_mode_fixed_speed

        The link speed remains at its default startup speed
        (10Mb/s nominal; actually within the range 9.8 to 10.2Mb/s)

   EtherSpaceLink_LINK_mode_slow_speed

        This setting combines the long_timeout and fixed_speed modifiers, thereby also setting the
        initial link speed to the final operating speed.
```

Returns

0 if the request queued , !0 if not

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_active_link ( esldev , 4 ))
{
    printf ("Unable to set  active link\n");
    return 2;
}

if (EtherSpaceLink_set_speed( esldev, 20 ))
{
    printf ("Unable to set 20MBs\n");
    return 3;
}

if (EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal))
{
    printf ("Failed to set mode\n");
    return 4;
}

return 0;


// For 10Mb/s operation on port 3, and 50Mb/s operation on other ports, use [on a DSI or SRR]:

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
}

EtherSpaceLink_set_active_link ( esldev, 3 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal | EtherSpaceLink_LINK_mode_fixed_speed );
EtherSpaceLink_set_active_link ( esldev, 1 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );
EtherSpaceLink_set_active_link ( esldev, 2 );
EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );
EtherSpaceLink_set_speed_double ( esldev, 50.0 );


// For 2Mb/s operation on port 5, use [on a DSI or SRR]:

EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
      printf( "Unable to connect\n");
 }

 EtherSpaceLink_set_active_link ( esldev, 5 );
 EtherSpaceLink_set_speed ( esldev, 2 );
 EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal | EtherSpaceLink_LINK_mode_slow_speed );
```

6.1.4.77 ESL_FN_EXPORT int EtherSpaceLink_set_mode_portmask (EtherSpaceLink *link,* int *mode,* uint32_t *ports*)

set mode of list of links

Sets the operating mode of a given set of links

After opening a connection, the link is disabled; it must then be enabled into one of its operational modes before data can be transferred.

The use of EtherSpaceLink_LINK_mode_fixed_speed to set some DSI ports to 10Mb/s, together with the conventional EtherSpaceLink_set_speed() mechanism is the only way to run a DSIs links at two different speeds. The active port is the highest number listed port in the mask

Parameters

link	connection to the device
mode	of operation

Returns

0 if the request queued , !0 if not

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_mode_portmask ( esldev,
      EtherSpaceLink_LINK_mode_normal , (1<<4) | (1<<3)))
{
    printf ("Failed to set mode\n");
    return 4;
}

return 0;
```

6.1.4.78 ESL_FN_EXPORT int EtherSpaceLink_set_record_file (EtherSpaceLink *link*, char * *file_name*)

Sets the recording file.

Used to set the recording file if there is a recording file in operation it will be closed

Parameters

link	the connection you want to set the record file
file_name	the name of the logging file

Returns

int, 0 if successful , non zero if not

in the event of failure sets the error to EtherSpaceLink_Error_RecFile_Open if the file could not be opened or EtherSpaceLink_Error_RecFile_Write if the file could not be written to.

passing a null pointer as the file name is treated as an error

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_log_file( esldev, "/tmp/debug.txt"))
{
    // setting  log failed ..

  // can decide to abandon things or try and continue without logging

}
if (EtherSpaceLink_set_record_file( esldev, "/traces/run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);

}
int some_really_bad_error = 0;

// lets do work with the device


// something gone wrong;
```

```
if (some_really_bad_error)
{
     esldev = EtherSpaceLink_shutdown(esldev);
     return 2;
}

// finished with device
esldev = EtherSpaceLink_close(esldev);
```

6.1.4.79 ESL_FN_EXPORT int EtherSpaceLink_set_rx_timeout (EtherSpaceLink *link,* int *to*)

Sets the rx timeout in milliseconds.

Sets the maximum period that an EtherSpaceLink_read_packet command will wait for data before returning. When the timeout expires, the EtherSpaceLink_read_packet command will return with as much data as it has received, if any. This is the maximum period of waiting after the last received data, not a delay from issuing the EtherSpaceLink_read_packet command.

Set the current recieve timeout (milliseconds)

Parameters

link	connection to ESL device
to	the new timeout

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

 // expect message back in 10 milliseconds
 EtherSpaceLink_set_rx_timeout( esldev, 10);

 printf ("RX timeout %d\n",  EtherSpaceLink_get_rx_timeout(esldev));

 return 0;
```

6.1.4.80 ESL_FN_EXPORT void EtherSpaceLink_set_rx_timeout_action (EtherSpaceLink *link,* int *returns_error*)

sets the behaviour on a networktimeout

Network errors are treated as a generic errors, but there are circumstances where one wants to know that an explicit timeout has occurred.

Parameters

link	the handle to the connection
returns_error	0 timeouts are returned as generic error !0 timeout errors are identified

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

// want to identify timeouts
EtherSpaceLink_set_rx_timeout_action ( esldev, 1);
```

6.1.4.81 ESL_FN_EXPORT void EtherSpaceLink_set_slot (EtherSpaceLink *link,* int *index,* int *type_*)

Sets a particular slot to a given slot type (used internally)

Parameters

link	connection to the device
slot	no
system	type

```
EtherSpaceLink esldev = EtherSpaceLink_open_flagged("filename.cap"
      , EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
       printf( "Unable to connect error %d\n", EtherSpaceLink_get_error());
       return 1;
}

EtherSpaceLink_set_slot( esldev, 0, 1);

return 0;
```

6.1.4.82 ESL_FN_EXPORT int EtherSpaceLink_set_speed (EtherSpaceLink *link,* int *speed*)

Sets the transmit speed of the link

Sets the transmit speed of all of the SpaceWire links on thisEtherSpaceLink unit.

Links set with the additional mode modifier EtherSpaceLink_LINK_mode_fixed_speed, which remain at their start-up speed of 10Mb/s.

Parameters

link	connection to the device
speed	the number of megabits per second

Returns

0 if request has been put on the wire, !0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_speed( esldev, 100))
{
     printf( "Failed to set speed connect\n");
     return 1;
}

return 0;
```

6.1.4.83 ESL_FN_EXPORT int EtherSpaceLink_set_speed_double (EtherSpaceLink *link,* double *speed*)

Sets the transmit speed of the link allowing partial Mb speeds.

Sets the transmit speed of all of the SpaceWire links on this EtherSpaceLink unit.

The links on an EtherSpaceLink unit can be set to a range of speeds; see the platform description for details.

Notice that the link speed in EtherSpaceLink_set_speed_double() is treated in units of bits/s, if the value is greater than 1 000 000, and in units of Mb/s otherwise. This is unlike EtherSpaceLink_set_speed, which uses Mb/s only.

All links on each EtherSpaceLink unit that are set using EtherSpaceLink_set_mode() to EtherSpaceLink_LINK_-mode_normal run at the same speed. The only exception to this are links set with the additional mode modifier EtherSpaceLink_LINK_mode_fixed_speed, which remain at their start-up speed of 10Mb/s.

Parameters

link	connection to the device
speed	the number of megabits per second

Returns

0 if request has been put on the wire, !0 if error

Hardware dsi

Bug Don't actually know if speed has been set

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_active_link ( esldev , 4 ))
{
    printf ("Unable to set  active link\n");
    return 2;
}

if (EtherSpaceLink_set_speed_double( esldev, 0.50))
{
     printf( "Failed to set speed connect\n");
     return 1;
}
return 0;
```

6.1.4.84 ESL_FN_EXPORT int EtherSpaceLink_set_timecode_transmit (EtherSpaceLink *link,* int *bits,* int *first,* int *interval,* int *report*)

Control the timecode generator.

Parameters

link	connection to the device
bits	' = 0 timecode does not change =-6 or = 6 lowest 6-bits of timecode increment (ECSS satandard) =-7 or = 7 lowest 7-bits of timecode increment =-8 or = 8 all 8-bits of timecode increment A negative value enables generation of reports when a code is sent
first	< 0 timecodes follow previous values >= 0 is the value of the first timecode sent after this command
interval	< 0 send only one timecode (with the value as set by 'first') = 0 send nothing (stop the timecode generator) = 1 send timecode on rising edge of external trigger = N send timecodes at intervals of N micro-seconds 100 <= N <= 999000000 (100us to 999s)
report	

Returns

0 if request sent, < 0 if not

6.1.4.85 ESL_FN_EXPORT int EtherSpaceLink_set_tx_record_file (EtherSpaceLink *link*, char * *file_name*)

Sets the tx recording file.

Used to set the recording file for recording transmitted space wire data

Parameters

link	the connection you want to set the record file
file_name	the name of the logging file

Returns

int, 0 if successful , non zero if not

in the event of failure sets the error to EtherSpaceLink_Error_RecFile_Open if the file could not be opened or EtherSpaceLink_Error_RecFile_Write if the file could not be written to.

passing a null pointer as the file name is treated as an error

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf( "Unable to connect\n");
     return 1;
}

if (EtherSpaceLink_set_tx_record_file( esldev, "/traces/tx_run.esl"))
{
    // unable to set record file .. abandon the run
    printf ("Error on record file ... giving up\n");
    esldev = EtherSpaceLink_close(esldev);

}
int some_really_bad_error = 0;

// lets do work with the device


// something gone wrong;


if (some_really_bad_error)
{
     esldev = EtherSpaceLink_shutdown(esldev);
     return 2;
}

// finished with device
esldev = EtherSpaceLink_close(esldev);
```

6.1.4.86 ESL_FN_EXPORT void* EtherSpaceLink_shutdown (EtherSpaceLink *link*)

Used to terminate a connection to an ESL device.

Generally when things have gone seriously wrong try to reset the TCP.IP networking to the device.

Parameters

link	connecton to hardware device

Returns

NULL to dereference the connection

```
char * host = "1.1.1.1:1234";

printf ("Connecting to %s ...\n", host);
EtherSpaceLink esldev = EtherSpaceLink_open(host);
if (!esldev)
{
```

```
     printf("Unable to connect to %s error %d\n", host, errno);
     return 1;
}
printf ("Have connection %p\n", esldev);


// check for error
int some_really_bad_error = 1;
if (some_really_bad_error)
{
     esldev = EtherSpaceLink_shutdown(esldev);
     if (esldev)
     {
        printf ("close has not worked\n");
        return 1;
    }
    return 0;
}
printf ("Not called shutdown\n");

return 1;
```

6.1.4.87 ESL_FN_EXPORT int EtherSpaceLink_sma_56_pulse_width (EtherSpaceLink *link,* int *width*)

Parameters

link	connection to the device
width	the pulse width for the SMA connector

It is possible to connect an SMA connector This sets the pulse width of the device

Returns

$<$ 0 indicating an error

6.1.4.88 ESL_FN_EXPORT void EtherSpaceLink_system_type (EtherSpaceLink *link,* int *type_*)

Forces system type such that platform derived timetags can be read.

Parameters

link	connection to the device
system	type

```
EtherSpaceLink esldev = EtherSpaceLink_open_flagged("filename.cap"
      , EtherSpaceLink_CONNECT_FILE, 0);
if (!esldev)
{
       printf( "Unable to connect error %d\n", EtherSpaceLink_get_error());
       return 1;
}

EtherSpaceLink_system_type( esldev, EtherSpaceLink_SYSTEM_TYPE_401);

return 0;
```

6.1.4.89 ESL_FN_EXPORT int EtherSpaceLink_TT_enable_reporting (EtherSpaceLink *link,* int *when*)

Enable timetags for currently active link.

Parameters

link	connection to the device
when	what events generate a timetag

Returns

0 on success or transmitted , < 0 if error

```
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
       printf( "Unable to connect\n");
       return 1;
}

// Only interested in parity errors
EtherSpaceLink_TT_enable_reporting(esldev,
      EtherSpaceLink_TT_report_parity_error);

return 0;
```

6.1.4.90 ESL_FN_EXPORT int EtherSpaceLink_write_buffer_empty (EtherSpaceLink *link*)

Returns if the write buffer for the currently active link is empty.

Parameters

link	connection to the device

Returns

1 if empty, 0 if not and < 0 in an error situation

Note: this function can drop inbound traffic

```
 EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
 if (!esldev)
 {
        printf( "Unable to connect\n");
        return 1;
 }

 if (EtherSpaceLink_set_active_link( esldev, 1))
 {
        printf( "Unable to set active link\n");
        return 1;
 }


if (EtherSpaceLink_write_packet( esldev, "ABCD", 4,
      EtherSpaceLink_EOP))
{
    printf ( "Unable to write data\n");
    return 1;
}


if (EtherSpaceLink_flush(link))
{
    printf ("Unable to write data\n");
    return 1;
}

 int con;
 switch ((con = EtherSpaceLink_write_buffer_empty( esldev )))
 {
     case 0:        // Not Empty
         printf ("Not Empty\n");
         break;

     case 1:        // Empty
         printf ("Empty\n");
         break;

     default:    // Unknown return 1;
         printf ("Unexpected error %d\n", con);
               return 1;
         break;
 }

 return 0;
```

6.1.4.91 ESL_FN_EXPORT int EtherSpaceLink_write_EXTN (EtherSpaceLink *link,* int *EXTN*)

Sends an extension packet.

```
Sends a single extension data character via the EtherSpaceLink unit to the currently-selected SpaceWire link

@param link the handle to the connection

@param EXTN id of the extension to write

    EtherSpaceLink_EOP
         End of Packet
    EtherSpaceLink_EEP
         Error End of Packet
    EtherSpaceLink_STORE
         Store flag - part of a store-and-forward sequence
    EtherSpaceLink_FORWARD
         Forward flag - part of a store-and-forward sequence
    EtherSpaceLink_port_select  to  EtherSpaceLink_port_select  + 15
         Select this SpaceWire port for data transmission;
          subsequent data transmitted will be sent to port n,
          where EtherSpaceLink_port_select = n =
          EtherSpaceLink_port_select + 15

          EtherSpaceLink_ESC
               EI Escape - follow with another character (e.g. ESC + data is a time-code)
          EtherSpaceLink_FCT
                EI Flow-control token
          EtherSpaceLink_ESC_EOP
                EI Escape and end-of-packet (ECSS error).
          EtherSpaceLink_ESC_EEP
                EI Escape and error-end-of-packet (ECSS error).
          EtherSpaceLink_ESC_ESC
                EI Escape and escape (ECSS error).
          EtherSpaceLink_ESC_FCT
                EI Escape and flow-control token (NULL).
          EtherSpaceLink_ParityError
                EI Invert the parity bit (and cause an ECSS error).
          EtherSpaceLink_EVENT
                EW Insert a flag - generate an event.
          EtherSpaceLink_JOIN
                SO Join a synchronisation
          EtherSpaceLink_RESIGN
                SO Leave a synchronisation
          EtherSpaceLink_BARRIER
                SO Synchronisation point
          EtherSpaceLink_DELAY  to    EtherSpaceLink_DELAY + 15
                EI,SO Insert delays: values 0 to 15 represent 1 to 16 * 4-bits additional time at the selected
                [(1 to 16) * 4 + 1 bits total] EtherSpaceLink_HOLD CO Hold the following characters until a lo
          EtherSpaceLink_TimeCode
                 EI To send an ECSS time-code, a second byte is  required for the time-code value and   EtherS


@return 0 if the packet has been queued, !0 if an error occurred queuing the extension
```

```
  //connect to host on port 1234
  EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
  if (!esldev)
  {
       printf( "Unable to connect\n");
       return 1;
  }

 printf ("Selects SpaceWire port 2 for subsequent transmissions\n");
 int result;
 result = EtherSpaceLink_write_EXTN ( esldev,
      EtherSpaceLink_PortSelect + 2);


  printf ("Sends ECSS time-code 34\n");
char tcode[] = {EtherSpaceLink_TimeCode & 255, 34};
result = EtherSpaceLink_write_packet ( esldev, tcode, sizeof(tcode),
EtherSpaceLink_EXTN );
EtherSpaceLink_flush ( esldev );
```

```
printf ("Messages sent\n");
return 0;
```

6.1.4.92 ESL_FN_EXPORT int EtherSpaceLink_write_packet (EtherSpaceLink *link*, void * *buffer*, size_t *length*, uint32_t *flags*)

queue data for transmission

Queues message for transmission, if there is no room left in the buffer, the buffer is transmitted. Note, that even the queued data is transmitted the data added to it may not be. If you want to guarantee transmission of this data you need to call flush.

Parameters

link	connection to ESL device
buffer	the data to send
length	the size of the buffer to size
flags	additional metadata about the frame we are transmitting

EtherSpaceLink_EOP

```
This is the last part, or all, of a data packet; an end-ofpacket  (EOP) is added.
```

EtherSpaceLink_EEP

```
This is the last part, or all, of a data packet; an error endof packet (EEP) is added.
```

EtherSpaceLink_PART_EOP_EEP

```
This is part of a data packet; no end-of-packet is added. This effectively allows one to send
part packet data, do not rely on this working correctly with other devices as it is not part
of the spacewire specification.
```

EtherSpaceLink_EXTN

```
This is a complete extension character sequence. Extension packets have a maximum length of 60 bytes.
```

EherSpaceLink_SPECIAL

```
This is a complete special packet
```

EEP would not normally be used to terminate a packet but is available here to assist with testing where an erroneous packet may usefully be generated. Data is queued in buffers in the API in order to make best use of the TC-P/IP stream and may not be sent immediately. EtherSpaceLink_flush() should be used to ensure the immediate transmission of any buffered data. The one-character extension sequences may be sent using EtherSpaceLink_-write_EXTN.

Hardware dsi

Returns

0 if sucessfull, <0 if not (-error number), errno setup and error code in handle

```
// connect to host on port 1234
EtherSpaceLink esldev = EtherSpaceLink_open("1.1.1.1:1234");
if (!esldev)
{
     printf("Unable to connect\n");
     return 1;
}

EtherSpaceLink_set_mode ( esldev,
      EtherSpaceLink_LINK_mode_normal );
```

```
// build message
char msg[1024];

int size = snprintf(msg, sizeof(msg), "Hello World %ld\n", time(NULL));

// send message
if (EtherSpaceLink_write_packet(esldev, msg, size,
      EtherSpaceLink_EOP ))
{
    // Error sending information or queuing ...
     printf("Unable to send\n");
     return 1;
}

// get the data on the wire
if (EtherSpaceLink_flush ( esldev ))
{
     printf("Unable to clear buffers\n");
     return 2;
}

EtherSpaceLink_flush ( esldev );
printf ("Message sent\n");

char rx_buffer[1024];
uint32_t flags;
int loop_index;
int bytes_received = EtherSpaceLink_read_packet ( esldev, rx_buffer, sizeof (
      rx_buffer), &flags);
printf ( "Received packet of length %i: ", bytes_received );
for ( loop_index = 0; loop_index < bytes_received; loop_index++ )
{
  printf ( " #%02x", rx_buffer[loop_index] & 0xff );
}
printf ( " %s\n", flags == EtherSpaceLink_PART_EOP_EEP ? "..." : flags ==
      EtherSpaceLink_EOP ? "EOP" : "EEP");
EtherSpaceLink_close ( esldev );
return 0;
```

6.1.4.93 ESL_FN_EXPORT SKT get_socket ( )

gets a transport socket for an ESL device

This may change in future releases

Returns

SKT transport for an ESL device

```
SKT skt = get_socket();

if (!invalid(skt))
{
   printf ("Socket is %d\n", skt);
   return 0;
}
printf ("Invalid socket\n");
return 1;
```

6.2 /src/api/C/EtherSpaceLink_Constants.h File Reference

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

Macros

- #define EtherSpaceLink_PART_EOP_EEP 1000

  Error packet.
- #define EtherSpaceLink_SPECIAL 1003

  We are sending a special frame.
- #define EtherSpaceLink_EXTN 1005

We are sending an extension frame.

- #define **EtherSpaceLink_PART_EXTN** 1006
- #define EtherSpaceLink_INCOMPLETE 1008

  Used to build up a single packet for the unit.
- #define **EtherSpaceLink_FLUSH** 2048
- #define EtherSpaceLink_PART_SPECIAL 1004

  A part of a special frame.
- #define EtherSpaceLink_TRUNCATED 1007

  Aritifical construct for unhandled data.
- #define EtherSpaceLink_SPECIAL_SIZE 1009

  Returning the amount of special data.
- #define EtherSpaceLink_EXTENSION_SIZE 1010

  Returning the amount of extension data.
- #define **EtherSpaceLink_FCT** 0x100
- #define EtherSpaceLink_EEP 0x101

  Error End of Packet.
- #define EtherSpaceLink_EOP 0x102

  End of Packet.
- #define EtherSpaceLink_ESC 0x103

  Escape.
- #define EtherSpaceLink_ESC_FCT 0x104

  Escape FCT aka a NULL character.
- #define EtherSpaceLink_ESC_EEP 0x105

  Escape End of Packet.
- #define EtherSpaceLink_ESC_EOP 0x106

  Escape Error of packet.
- #define EtherSpaceLink_ESC_ESC 0x107

  Escape Escape.
- #define EtherSpaceLink_Timeout 0x108

  Timeout message.
- #define EtherSpaceLink_ParityError 0x109

  Parity Error message.
- #define EtherSpaceLink_PERROR1 0x10A

  Error 1 message.
- #define EtherSpaceLink_PERROR2 0x10B

  Error 2 message.
- #define **EtherSpaceLink_STORE** 0x10C
- #define **EtherSpaceLink_FORWARD** 0x10D
- #define **EtherSpaceLink_ATOM** 0x10E
- #define **EtherSpaceLink_MOTA** 0x10F
- #define **EtherSpaceLink_JOIN** 0x110
- #define **EtherSpaceLink_BARRIER** 0x111
- #define **EtherSpaceLink_RESIGN** 0x112
- #define **EtherSpaceLink_EVENT** 0x113
- #define EtherSpaceLink_Missing_data 0x114

  Missed data message.
- #define **EtherSpaceLink_HOLD** 0x12F
- #define **EtherSpaceLink_Delay** 0x130
- #define EtherSpaceLink_PortSelect 0x140

  Port select message.
- #define EtherSpaceLink_PortSelect_max 0x17F

  Max port select message.

- #define **EtherSpaceLink_Multi_byte_extn_start** 0x180
- #define EtherSpaceLink_TimeTag 0x188

  Timetag message.
- #define EtherSpaceLink_TimeTag_delta 0x182

  Timetag delta message.
- #define EtherSpaceLink_TimeTag_uncertainty 0x181

  Timetag uncertain message.
- #define EtherSpaceLink_TimeCode 0x191

  Spacewire timecode.
- #define EtherSpaceLink_Module 0x192

  Module data.
- #define EtherSpaceLink_TimeZero 0x198

  First timecode on the link.
- #define **EtherSpaceLink_TRUNCATE_1** 0x1A1
- #define **EtherSpaceLink_TRUNCATE_2** 0x1A2
- #define **EtherSpaceLink_REPEAT_1** 0x1B1
- #define **EtherSpaceLink_REPEAT_2** 0x1B2
- #define **EtherSpaceLink_REPEAT_3** 0x1B3
- #define EtherSpaceLink_Year 0x1C8

  Capture start date/time.
- #define EtherSpaceLink_Header 0x1CE

  Capture Header containing version and time information.
- #define EtherSpaceLink_report_EEP 0x800000

  EEP error event.
- #define EtherSpaceLink_report_nchar 0x400000

  character received event
- #define EtherSpaceLink_report_first_null 0x200000

  First null event.
- #define EtherSpaceLink_report_excess_FCT 0x100000

  Too many FCTS event.
- #define EtherSpaceLink_report_excess_data 0x080000

  Too much data sent for # of FCT's.
- #define EtherSpaceLink_report_first_byte 0x040000

  First byte of packet.
- #define EtherSpaceLink_report_mid_bytes 0x020000

  Frame mide byte.
- #define EtherSpaceLink_report_EOP 0x010000

  EOP recieved.
- #define EtherSpaceLink_report_time_code 0x008000

  Time code received.
- #define EtherSpaceLink_report_FCT 0x004000

  FCT received.
- #define EtherSpaceLink_report_NULL 0x002000

  Null received.
- #define EtherSpaceLink_report_parity_error 0x001000

  Parity Error.
- #define EtherSpaceLink_report_ESC_EOP 0x000800

  Escape EOP error.
- #define EtherSpaceLink_report_ESC_EEP 0x000400

  Escape EEP Error.
- #define EtherSpaceLink_report_ESC_ESC 0x000200

Escape Escape Error.

- #define EtherSpaceLink_report_timeout 0x000100

  Link Timeout.
- #define **EtherSpaceLink_report_delta** 0x400000
- #define **EtherSpaceLink_DISCARD_SPECIAL_DATA** 0x00
- #define **EtherSpaceLink_REPORT_SPECIAL_DATA** 0x01
- #define **EtherSpaceLink_RETURN_SPECIAL_DATA** 0x02
- #define **EtherSpaceLink_CALLBACK_SPECIAL_DATA** 0x03
- #define **EtherSpaceLink_SPECIAL_DATA_FLAGS** 0x03
- #define **EtherSpaceLink_DISCARD_EXTENSION_DATA** 0x00
- #define **EtherSpaceLink_REPORT_EXTENSION_DATA** 0x10
- #define **EtherSpaceLink_RETURN_EXTENSION_DATA** 0x20
- #define **EtherSpaceLink_CALLBACK_EXTENSION_DATA** 0x30
- #define **EtherSpaceLink_EXTENSION_DATA_FLAGS** 0x30
- #define **EtherSpaceLink_READ_IMMEDIATE** 0x40
- #define **EtherSpaceLink_SpaceWire_state_ErrorReset** 0
- #define **EtherSpaceLink_SpaceWire_state_ErrorWait** 1
- #define **EtherSpaceLink_SpaceWire_state_Ready** 2
- #define **EtherSpaceLink_SpaceWire_state_Started** 3
- #define **EtherSpaceLink_SpaceWire_state_Connecting** 4
- #define **EtherSpaceLink_SpaceWire_state_Run** 5
- #define **EtherSpaceLink_MSR_state_NC** 6
- #define **EtherSpaceLink_MSR_state_Connected** 7
- #define **EtherSpaceLink_CAPABILITIES** 0
- #define **EtherSpaceLink_HWA** 3
- #define **EtherSpaceLink_LINK_SPEED** 4
- #define **EtherSpaceLink_MANUFACTURER** 1
- #define **EtherSpaceLink_PRODUCT** 2
- #define **EtherSpaceLink_LINK** 5
- #define EtherSpaceLink_LINK_mode_disabled 0x01

  Disables the link.
- #define EtherSpaceLink_LINK_mode_normal 0x02

  Enables the link.
- #define EtherSpaceLink_LINK_mode_legacy 0x04

  IEEE 1355 (spacewire precursor)
- #define EtherSpaceLink_LINK_mode_master 0x06

  IEEE 1355 (precurosr)
- #define **EtherSpaceLink_LINK_tx_buffer_empty** 0x08
- #define **EtherSpaceLink_LINK_state_offset** 4
- #define **EtherSpaceLink_LINK_mode_long_timeout** 0x40
- #define **EtherSpaceLink_LINK_mode_fixed_speed** 0x80
- #define **EtherSpaceLink_LINK_mode_slow_speed** 0xC0
- #define **EtherSpaceLink_SF** 6
- #define **EtherSpaceLink_SF_disabled** 0x00
- #define **EtherSpaceLink_SF_enabled** 0x80
- #define **EtherSpaceLink_TT** 7
- #define **EtherSpaceLink_TT_64** 15
- #define EtherSpaceLink_TT_report_nothing 0x00

  Report Nothing.
- #define EtherSpaceLink_TT_report_first_byte (0x01 | EtherSpaceLink_report_first_byte)

  Timetag first byte of packet.
- #define EtherSpaceLink_TT_report_intermediate_bytes (0x02 | EtherSpaceLink_report_mid_bytes)

  Timetag middle byte.

- #define EtherSpaceLink_TT_report_EOP_EEP (0x04 | EtherSpaceLink_report_EEP | EtherSpaceLink_report_EOP)

  Timetag end of packet markers.
- #define EtherSpaceLink_TT_report_EEP (0x04 | EtherSpaceLink_report_EEP)

  Timetag report Error End of Packet.
- #define EtherSpaceLink_TT_report_EOP (0x04 | EtherSpaceLink_report_EOP)

  Timetag report End of Packet.
- #define EtherSpaceLink_TT_report_time_code EtherSpaceLink_report_time_code

  Timetag report spacewire timecode.
- #define EtherSpaceLink_TT_report_fct EtherSpaceLink_report_FCT

  Timetag report FCT.
- #define EtherSpaceLink_TT_report_null EtherSpaceLink_report_NULL

  Timetag report NULL.
- #define EtherSpaceLink_TT_report_parity_error EtherSpaceLink_report_parity_error

  Timetag report parity error.
- #define EtherSpaceLink_TT_report_ESC_EOP EtherSpaceLink_report_ESC_EOP

  Timetag report ESC End of Packet.
- #define EtherSpaceLink_TT_report_ESC_EEP EtherSpaceLink_report_ESC_EEP

  Timetag report ESC Error End of Packet.
- #define EtherSpaceLink_TT_report_ESC_ESC EtherSpaceLink_report_ESC_ESC

  Timetag report ESC ESC.
- #define EtherSpaceLink_TT_report_timeout EtherSpaceLink_report_timeout

  Timetag report timeout.
- #define **EtherSpaceLink_ER** 8
- #define **EtherSpaceLink_ER_64** 16
- #define EtherSpaceLink_ER_report_nothing 0x00

  Error reporting report nothing.
- #define EtherSpaceLink_ER_report_first_null 0x02

  Error report first null.
- #define EtherSpaceLink_ER_report_first_fct 0x04

  Error report first fct.
- #define EtherSpaceLink_ER_report_running_error (0x08 | EtherSpaceLink_report_parity_error | EtherSpaceLink_report_ESC_EOP | EtherSpaceLink_report_ESC_EEP | EtherSpaceLink_report_ESC_ESC | EtherSpaceLink_report_timeout)

  Error report running.
- #define **EtherSpaceLink_ER_report_starting_error** 0x10
- #define **EtherSpaceLink_ER_report_nchar** 0x40
- #define EtherSpaceLink_ER_report_time_code (0x80 | EtherSpaceLink_report_time_code)

  report time code
- #define EtherSpaceLink_ER_report_fct EtherSpaceLink_report_FCT

  report FCT
- #define EtherSpaceLink_ER_report_null EtherSpaceLink_report_NULL

  report null
- #define EtherSpaceLink_ER_report_parity_error EtherSpaceLink_report_parity_error

  report parity error
- #define EtherSpaceLink_ER_report_ESC_EOP EtherSpaceLink_report_ESC_EOP

  report Escape End of Packet
- #define EtherSpaceLink_ER_report_ESC_EEP EtherSpaceLink_report_ESC_EEP

  report Escape Error End of Packet
- #define EtherSpaceLink_ER_report_ESC_ESC EtherSpaceLink_report_ESC_ESC

  report Escape Escape

- #define EtherSpaceLink_ER_report_timeout EtherSpaceLink_report_timeout

  report Timeout
- #define **EtherSpaceLink_EW** 9
- #define **EtherSpaceLink_EW_RT** 13
- #define **EtherSpaceLink_EW_capture_nothing** 0x00
- #define EtherSpaceLink_EW_capture_first_null (0x02 | EtherSpaceLink_report_first_null)

  trigger on first null
- #define EtherSpaceLink_EW_capture_first_fct 0x04

  trigger on first fct
- #define EtherSpaceLink_EW_capture_running_error (0x08 | EtherSpaceLink_report_parity_error | EtherSpaceLink_report_ESC_EOP | EtherSpaceLink_report_ESC_EEP | EtherSpaceLink_report_ESC_ESC | EtherSpaceLink_report_timeout)

  trigger on run error
- #define EtherSpaceLink_EW_capture_starting_error 0x10

  trigger on start error
- #define EtherSpaceLink_EW_capture_nchar (0x40 | EtherSpaceLink_report_nchar)

  trigger on n char
- #define EtherSpaceLink_EW_capture_time_code (0x80 | EtherSpaceLink_report_time_code)

  trigger on timecode
- #define EtherSpaceLink_EW_capture_EOP EtherSpaceLink_report_EOP

  trigger on End of Packet
- #define EtherSpaceLink_EW_capture_EEP EtherSpaceLink_report_EEP

  trigger on Error End of Packet
- #define EtherSpaceLink_EW_capture_FCT EtherSpaceLink_report_FCT

  trigger on FCT
- #define EtherSpaceLink_EW_capture_excess_FCT EtherSpaceLink_report_excess_FCT

  trigger on excess fct
- #define EtherSpaceLink_EW_capture_excess_data EtherSpaceLink_report_excess_data

  trigger on excess data
- #define EtherSpaceLink_EW_capture_null EtherSpaceLink_report_NULL

  trigger on NULL
- #define EtherSpaceLink_EW_capture_parity_error EtherSpaceLink_report_parity_error

  trigger on parity error
- #define EtherSpaceLink_EW_capture_ESC_EOP EtherSpaceLink_report_ESC_EOP

  trigger on Escape End of Packet
- #define EtherSpaceLink_EW_capture_ESC_EEP EtherSpaceLink_report_ESC_EEP

  trigger on Escape Error End of Packet
- #define EtherSpaceLink_EW_capture_ESC_ESC EtherSpaceLink_report_ESC_ESC

  trigger on Escape Escape
- #define EtherSpaceLink_EW_capture_timeout EtherSpaceLink_report_timeout

  trigger on timeout
- #define EtherSpaceLink_EW_Source_barrier 0x0001

  Barrier.
- #define EtherSpaceLink_EW_Source_port_1 0x0002

  Port 1.
- #define EtherSpaceLink_EW_Source_port_2 0x0004

  Port 2.
- #define EtherSpaceLink_EW_Source_port_3 0x0008

  Port 3.
- #define EtherSpaceLink_EW_Source_port_4 0x0010

  Port 4.

- #define EtherSpaceLink_EW_Source_port_5 0x0020

  Port 5.
- #define EtherSpaceLink_EW_Source_port_6 0x0040

  Port 6.
- #define EtherSpaceLink_EW_Source_port_7 0x0080

  Port 7.
- #define EtherSpaceLink_EW_Source_port_8 0x0100

  Port 8.
- #define EtherSpaceLink_EW_Source_SMA_12 0x0200

  SMA 1/2 changing state.
- #define EtherSpaceLink_EW_Source_SMA_34 0x0400

  SMA 3/4 changing state.
- #define EtherSpaceLink_EW_Source_SMA_56 0x0800

  SMA 5/6 changing state.
- #define EtherSpaceLink_EW_Source_SMA_78 0x1000

  SMA 7/8 changing state.
- #define EtherSpaceLink_EW_Source_local_clock 0x8000

  Local clock.
- #define **EtherSpaceLink_TC_rx** 10
- #define **EtherSpaceLink_TC_rx_64** 17
- #define **EtherSpaceLink_TC_rx_silent** 0x00
- #define **EtherSpaceLink_TC_rx_report_enabled** 0x08
- #define **EtherSpaceLink_TC_rx_time_stamp_enabled** 0x40
- #define **EtherSpaceLink_TC_tx** 11
- #define **EtherSpaceLink_TC_tx_trigger_mask** 0x03
- #define **EtherSpaceLink_TC_tx_no_trigger** 0x00
- #define **EtherSpaceLink_TC_tx_one_code** 0x01
- #define **EtherSpaceLink_TC_tx_external_trigger** 0x02
- #define **EtherSpaceLink_TC_tx_regular_trigger** 0x03
- #define **EtherSpaceLink_TC_tx_update_interval** 0x04
- #define **EtherSpaceLink_TC_tx_update_code** 0x08
- #define **EtherSpaceLink_TC_tx_format_mask** 0x30
- #define **EtherSpaceLink_TC_tx_no_increment** 0x00
- #define **EtherSpaceLink_TC_tx_increment_6_bits** 0x10
- #define **EtherSpaceLink_TC_tx_increment_7_bits** 0x20
- #define **EtherSpaceLink_TC_tx_increment_8_bits** 0x30
- #define **EtherSpaceLink_TC_tx_report_transmission** 0x40
- #define **EtherSpaceLink_CR** 14
- #define **EtherSpaceLink_router_cs** 18
- #define **EtherSpaceLink_router_tables** 19
- #define **EtherSpaceLink_router_stats** 20
- #define **EtherSpaceLink_ram_rw** 21
- #define **EtherSpaceLink_barrier** 22
- #define **EtherSpaceLink_TT_now** 23
- #define **EtherSpaceLink_EI_ignore_excess_FCT** EtherSpaceLink_report_excess_FCT
- #define **EtherSpaceLink_EI_ignore_excess_data** EtherSpaceLink_report_excess_data
- #define **EtherSpaceLink_EI_ignore_parity_error** EtherSpaceLink_report_parity_error
- #define **EtherSpaceLink_EI_ignore_ESC_EOP** EtherSpaceLink_report_ESC_EOP
- #define **EtherSpaceLink_EI_ignore_ESC_EEP** EtherSpaceLink_report_ESC_EEP
- #define **EtherSpaceLink_EI_ignore_ESC_ESC** EtherSpaceLink_report_ESC_ESC
- #define **EtherSpaceLink_EI_ignore_timeout** EtherSpaceLink_report_timeout
- #define **EtherSpaceLink_EI_normal_flow_control** 0x00
- #define **EtherSpaceLink_EI_transmit_anyway** 0x20

- #define **EtherSpaceLink_EI_no_automatic_FCT** 0x10
- #define **EtherSpaceLink_LINK_address** 0x0000
- #define **EtherSpaceLink_TX_SPEED_address** 0x87FD
- #define **EtherSpaceLink_RX_SPEED_address** 0x0001
- #define **EtherSpaceLink_HWA_address** 0x8800
- #define **EtherSpaceLink_VERSION_address** 0x880A
- #define **EtherSpaceLink_DESCRIPTION_address** 0x880B
- #define **EtherSpaceLink_OPTIONS_address** 0x8F60
- #define **EtherSpaceLink_NLINKS_address** 0x8FFF
- #define **EtherSpaceLink_EW_address** 0x1000
- #define **EtherSpaceLink_PC_address** 0x2000
- #define **EtherSpaceLink_PG_address** 0x4000
- #define **EtherSpaceLink_ATI_address** 0x0100
- #define **EtherSpaceLink_OBSERVE_address** 0x0020
- #define **EtherSpaceLink_TIMETAG_address** 0x0030
- #define **EtherSpaceLink_IGNORE_address** 0x0040
- #define **EtherSpaceLink_Event_cause_address** 0x0060
- #define **EtherSpaceLink_EW_source_address** 0x0070
- #define **EtherSpaceLink_FLOW_CONTROL_address** 0x0050
- #define **EtherSpaceLink_SMA_56_pulse_width_address** 0x00F0
- #define **EtherSpaceLink_max_packet_data** 0x0010
- #define EtherSpaceLink_Error_RecFile_Open -1

  Couldn't open recording file.
- #define EtherSpaceLink_Error_RecFile_Write -2

  record_file write failed
- #define EtherSpaceLink_Error_LogFile_Open -3

  Couldn't open logging file.
- #define EtherSpaceLink_Error_LogFile_Write -4

  log_file write failed
- #define EtherSpaceLink_Error_Receiver_Timeout -10

  we have a network timeout timeout
- #define EtherSpaceLink_Error_Receiver_Shutdown -11

  peer has performed an orderly shutdown
- #define EtherSpaceLink_Error_IO_Error -12

  we have an IO error
- #define EtherSpaceLink_Error_SaveBuf_Overflow_Save -15

  Saving the read_packet_full() save_buffer failed.
- #define EtherSpaceLink_Error_SaveBuf_Overflow_Restore -16

  Restoring the read_packet_full() save_buffer failed.
- #define EtherSpaceLink_Error_Function_Not_Supported -17

  Device does not support the requested function.
- #define EtherSpaceLink_Error_Network -18

  Error reading / writing to/from the device.
- #define EtherSpaceLink_Error_Network_Format_Error -19

  Error understanding recieved packet.
- #define EtherSpaceLink_Error_Request_Too_Large -20

  The I/O request can't be fullfilled by the hardware.
- #define EtherSpaceLink_Error_Sequence_Error -21

  Didn't receive expected notification from the hardware.
- #define EtherSpaceLink_Error_Response_Too_Small -22

  Response from the device didn't contain enough data.
- #define EtherSpaceLink_Error_Response_Mismatch -23

Response does not match I/O request.

- #define EtherSpaceLink_Error_Module_Not_Present -24

  Module not present.
- #define EtherSpaceLink_Error_Parameter_RangeIncorrect -25

  Parameter not in range.
- #define EtherSpaceLink_Error_File_Not_Present -26

  Requested file is not present.
- #define EtherSpaceLink_Error_EINTR -27

  EINTR occurred.
- #define EtherSpaceLink_Error_Link_Incorrect -28

  Link number is incorrect.
- #define EtherSpaceLink_Error_Incorrect_Device -29

  Connecting to a device which does not support functionality.
- #define EtherSpaceLink_Error_Memory -30

  Unable to allocate memory.
- #define EtherSpaceLink_Error_Host_Unresolvable -31

  Unable to resolve host.
- #define EtherSpaceLink_Error_Host_Unresponsive -32

  Unable to connect to host.
- #define EtherSpaceLink_Error_WaveForm_Dir_Create -33

  Unable to create waveform directory.
- #define EtherSpaceLink_Error_Zero_Read -34

  asked to read zero bytes
- #define EtherSpaceLink_Error_Set_Option_File -35

  Asked to set an option when playing back from file.
- #define EtherSpaceLink_Error_Invalid_Device -36

  Device is not supported by API.
- #define EtherSpaceLink_Error_File_Move -37

  Unable to move file into place.
- #define EtherSpaceLink_Error_Invalid_File -38

  Unable to open file.
- #define EtherSpaceLink_Error_Callback_Return -39

  Callback has asked for a return.
- #define EtherSpaceLink_Error_FileList_Empty -40

  List of files given is empty.
- #define EtherSpaceLink_Error_Unknown_System_Type -41

  Unknown type.
- #define EtherSpaceLink_Error_Not_Known -42

  API returned 0 as an error should (should not happen)
- #define EtherSpaceLink_Error_EXE_Start_Failed -43

  Cannot start executable.
- #define EtherSpaceLink_Error_NO_Connection -44

  Link Not estblished.
- #define EtherSpaceLink_Error_Invalid_Link -45

  Invalid Link selected.
- #define EtherSpaceLink_Error_Would_Block -48

  I/O call would block.
- #define EtherSpaceLink_Error_Link_Not_Connected -49

  Link Not Connected.
- #define EtherSpaceLink_Error_ReadHandler_Running -50

  There is a read handler running for this connection.

- #define EtherSpaceLink_Error_Buffer_Full -51

  can't do non blocking write as buffer is full
- #define EtherSpaceLink_Error_CaptureThread_Failed -52

  Capture thread failed.
- #define EtherSpaceLink_Option_SO 1

  Option SO module is installed.
- #define **EtherSpaceLink_CONNECT_FILE** (1)
- #define **EtherSpaceLink_Receiver_Timeout_Returns_Zero_Part_Pkt** 0
- #define **EtherSpaceLink_Receiver_Timeout_Returns_Error** 1
- #define **EtherSpaceLink_SYSTEM_TYPE_INVALID** 0
- #define **EtherSpaceLink_SYSTEM_TYPE_401** 1
- #define **EtherSpaceLink_SYSTEM_TYPE_408** 2

6.2.1 Detailed Description

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

Index