Software 34.81

Release Notes

Version Number change

We have now added patch release numbers to the version identification.

The version numbering is now of the format

<ReleaseNumber>.<VersionNumber><PatchLevel>

The first patch to this version will be known as 34.82.

C++ RMAP

The C++ API for RMAP has been corrected with respect to the endianness of the command byte, previously the bit positions of command flags were reversed causing corrupt RMAP messages.

READ program

The java read program did not load the RMAP plugin correctly.

RMAP plugin

Support for Read Modify Write verb has been added

MSR capture program

The msr capture program places waveforms in the current directory if the environment variable ESL_WF_DIR is not set. ESL_WF_DIR is an environment variable that controls where waveform files are saved.

Waveform files

When a waveform file is saved by the msr program, the name of the file is printed to stdout and the waveform file is prefixed with the epoch time. Allowing easy identification of the capture file in any logs.

SVG Waveform files

These are now scaled correctly, so there are no longer have large blocks of unreadable color.

Python 3.7 Support on Windows

Support for Python 3.7 on windows. If python 3.7 is required, set ESL_PYTHON_VER to 3.7 prior to running 4linksvars.

Python timecode example

examples/python/timecode is an example of how to send and listen for Spacewire timecodes.

Graphical Capture program on Windows

The graphical capture program uses GTK, however the GTK runtime ships with a DLL used by other packages and this can stop the program starting. The environment program 4linksvars.bat puts the GTK libraries in the path before other libraries.

Change of License for example programs

All examples are now shipped with the BSD license making copying and reuse of the examples easier.

Graphical Capture program / WIreshark Integration

It is now possible to capture directly into Wireshark from within the graphical capture program

Improved PCAP buffering

The PCAP file buffer has been made an optimal size.

API Changes

Return Values

Throughout a return value of –1 was used to indicate an error, this has been changed. If a function returns a value < 0 an error has occurred, the returned value being the error code as described in EtherSpaceLink_Constants.h. This change does not affect the C++ or Python bindings as error handling is now by exception.

EtherSpaceLink_read_packet_full

When called with **EtherSpaceLink_REPORT_SPECIAL_DATA** and **EtherSpaceLink_REPORT_EXTENSION_DATA** this function used to return –2 – <number of bytes>. This conflicts with the error handling as defined above. Instead the returned flags indicate whether a size is being returned. If the returned flags are **EtherSpaceLink_SPECIAL_SIZE** or **EtherSpaceLink_EXTENSION_SIZE** then the returned value is the number of special or extension bytes.

Example:

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

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

  bytes_received = EtherSpaceLink_read_packet_full ( esldev,
                                        rxbuf,
                                        sizeof(rxbuf),
                                        &flags,
                                        EtherSpaceLink_REPORT_EXTENSION_DATA
                                        | EtherSpaceLink_REPORT_SPECIAL_DATA
                                       );
If (bytes_received < 0)
{
    // Error condition
    fprintf (stderr, “Error %d \n”, bytes_received);
    return 1;
}
switch (flags)
{
    case EtherSpaceLink_SPECIAL_SIZE:
    case EtherSpaceLink_EXTENSION_SIZE:
          fprintf (stdout, “%d bytes of special data\n”, bytes_received);
}
```

receivePacket (java)

This function no longer reports the size of extension or special data as a negative number. Instead one must check the context of the return value by calling **get_terminator.** If this function returns **SPECIAL_SIZE** or **EXTENSION_SIZE** then the size returned is the extension size. Note that if **enable_callbacks** is called the application developer does not have to perform analysis of special and extension_data, and instead you get high level callbacks (e.g. link changed, error, or waveform data);

EtherspaceLink_fastclose

Introduced new API verb which shuts down the connection as soon as possible using SO_LINGER with a timeout of 0.

Language Implementations

C++ method fastclose

Java method fastclose

Python method fastclose

EtherspaceLink_sendtimecode

Explicit function to send a Spacewire timecode.

Language Implementations

C++ method send_timecode(timecode)

Java method send_timecode(timecode)

Python method send_timecode(timecode)

EtherspaceLink_get_error

This returns the last error to occur in the **current thread** in the EtherSpaceLink library, mostly it should not be required to call this as functions return the error code directly. This change ensures that error codes are thread safe between receiver and transmit threads.

EtherspaceLink_get_error_text

This returns a string description of the error last error to occur in the **current thread.**

This change ensures that error codes are thread safe between receiver and transmit threads.