# Reducing SpaceWire Time-code Jitter

Barry M Cook

***4Links Limited***
***The Mansion,***
***Bletchley Park,***
***Milton Keynes,***
***MK3 6ZP,***
***UK***
***Email: barry@4Links.co.uk***

## INTRODUCTION

Standards ISO/IEC 14575[1] and IEEE 1355[2] define an inter-processor communication link suitable for multi-processor applications. Data transfer is high speed and, more importantly, low latency. This standard was used to create links integrated with high performance processors and a low-latency routing switch, using *cut-through* (also known as *wormhole*) routing and outstanding aggregate performance demonstrated.

Some aspects of IEEE 1355, such as the signal levels (PECL for wired connections) and connectors used are not suited to Space applications and led to a variant being defined which has become known as *SpaceWire.*

SpaceWire currently defines only twisted-pair wired connections using LVDS signaling and specifies connectors that are available in Space qualified versions (IEEE 1355 also contains co-axial cable and optical transmission specifications). One extension made to IEEE 1355 is the inclusion of a mechanism for global distribution of values for use as time-codes.

Time-codes are broadcast to all end nodes in the system. The time-code source and each routing switch does, however introduce some jitter on the received time code, which can amount to several microseconds. It is possible, completely within the standard, to send, forward and receive time-codes in such a way that the jitter is reduced to the order of tens of nanoseconds. Furthermore, it is possible to align received time-code indications so that all nodes are synchronized to within a few tens of nanoseconds.

We discuss the time codes themselves and the source of time uncertainty – jitter. An experimental test-set, Fig. 1., was used to verify the predicted behaviour and a probability density function (PDF) of tick arrival times plotted, Fig 2. A technique for reducing jitter is described and a typical resulting signal waveform shown in Fig. 4., with a normal waveform for reference, Fig. 3. The resulting PDF is dramatically improved, as shown in Fig. 5. A further extension allowing synchronized time code delivery is described and the resulting arrival time PDF shown in Fig. 6.

## TOKENS

SpaceWire transmits basic tokens of either 4- or 10-bits. Each token contains a flag bit indicating its type and a parity bit for detecting transmission errors. Short, 4-bit, tokens are used for control – encoding end-of-packet marks, flow control credit and one *escape* token. Long, 10-bit, tokens are used for data. Compounds are formed with the escape control token: escape with flow control for a null token, and escape with data for a time code – see Table 1. Escape with end-of-packet or another escape are not currently used and their detection results in an error condition which, like parity errors, causes the link to be reset.

In order to confirm connection and to be able to detect loss of connection, the link is always sending tokens. Priority is given to time-code and flow control tokens, then data and flow control tokens, and finally, if there is nothing of more importance then a Null token is sent in order to maintain continuous transitions and indicate that the link is still connected.

Table 1. SpaceWire tokens

| | Use | Length |
|---|---|---|
| EOP | End-of-packet marker | 4-bits |
| EEP | End-of-packet marker | 4-bits |
| FCT | Flow control | 4-bits |
| ESC | (Only as part of compound) | |
| Data | Byte-wide data | 10-bits |
| ESC-FCT | Null | 8-bits |
| ESC-Data | Time code | 14-bits |

## SPACEWIRE TIME CODES

Time codes are globally transmitted – broadcast – from a single source to all nodes in the network. SpaceWire networks are permitted to have multiple connections in order to provide redundancy and fault tolerance. These alternate paths form loops that can lead to livelock with a simple broadcast mechanism – SpaceWire routing switches may receive a time code already sent and must only selectively transmit new values. A time code, therefore, contains a sequence number in support of this requirement.

Six-bits are currently used to define the sequence number and, together with two reserved bits, are contained in a normal data token. An escape token precedes this data token in order to identify it as a time code.

In normal usage, time codes are transmitted from the time source at regular intervals. Each node receives both the sequence number (and reserved bits) and a *tick* indicating its arrival time.

## JITTER PERFORMANCE

Jitter, uncertainty, in the time-code indication – the time code "tick" – has two sources. The SpaceWire transmitter being the major contributor with a small addition from the receiver. It is assumed that a well-designed routing switch has a fixed delay and adds no further uncertainty.

### Transmitter Contribution

Requests to send a time code will not occur at a regular time relative to the transmitted token stream. We may expect time ticks to be generated at regular intervals but the tokens, being of variable length may not stop and start at the same times.

It may be possible to send a time-code at the instant a tick request is made, or it may be necessary for the currently being sent token to finish first – a delay of up to 10-bit times (we assume that a time tick will not be so soon after a previous time tick that it must wait 14 bit times). Thus, we see an introduced jitter of 10 bit-times.

The approved standard [3] considers only this contribution in its example jitter calculation. It takes the default transmit rate of 10Mb/s to give a transmit uncertainty of 1μs at each link. Across a reasonable system having ten such links we see an accumulated jitter expectation of some 10μs.

### Receiver Contribution

Link ends are not required to share a common clock signal, and in general will not do so, with the result that the SpaceWire transmitter and receiver are not synchronized. The receiver must resynchronize what is received with its own clock and an uncertainty of one sampling period is inevitable. The receive sampler may operate at a frequency corresponding to the link bit rate but need not do so. For example, the XOR-recovered clock may be used to group received bits into tokens and these tokens passed to the local clock domain, which may therefore be at a lower frequency than the bit-rate might suggest. The sampling uncertainty occurs at this last transition and can thus be greater than one bit time.

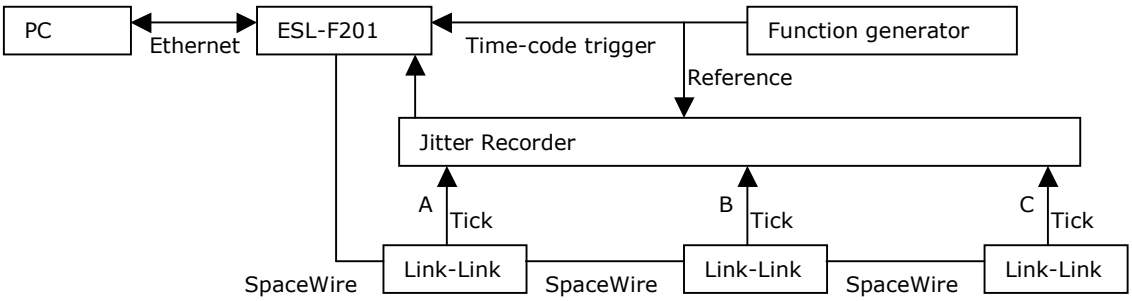


Fig. 1. Experimental test harness

**Experimental Verification**

In order to test this theoretical prediction, and to provide a reference for comparison with an improved scheme, a test harness was constructed, as shown in Fig. 1. A PC connected, via Ethernet, to an EtherSpaceLink ESL-F201 able to generate time-codes at intervals set by an external function generator in turn connected by SpaceWire link to a unit containing two link engines to copy data to another SpaceWire link and again on to a third link. Reference signals and received time ticks were collected in a jitter recorder that accumulated the number of ticks received at each of 1023 intervals of 10ns after the reference – a probability density function – the ESL-F201 was used to pass the recorded data back to the PC. An external function generator was used in preference to the internal time-code generator in the ESL-F201 in order to de-synchronize the tick requests from the link clock.

Links were set to transmit at 12.5Mb/s, for reasons explained later, and a large number of time-ticks were generated and recorded.

The resulting distributions are shown in Fig. 2. The top trace shows the distribution after one SpaceWire link (point A in Fig. 1.), the middle trace after two links (point B in Fig. 1.) and the bottom trace after three links (point C in Fig. 1.). Due to the need to scale the number of ticks received at each time point, it is hard to see points at which only a relatively few ticks were received – a bar appears under each group of received ticks indicating the range from earliest to latest, together with text giving the difference between these times – the peak-to-peak jitter.

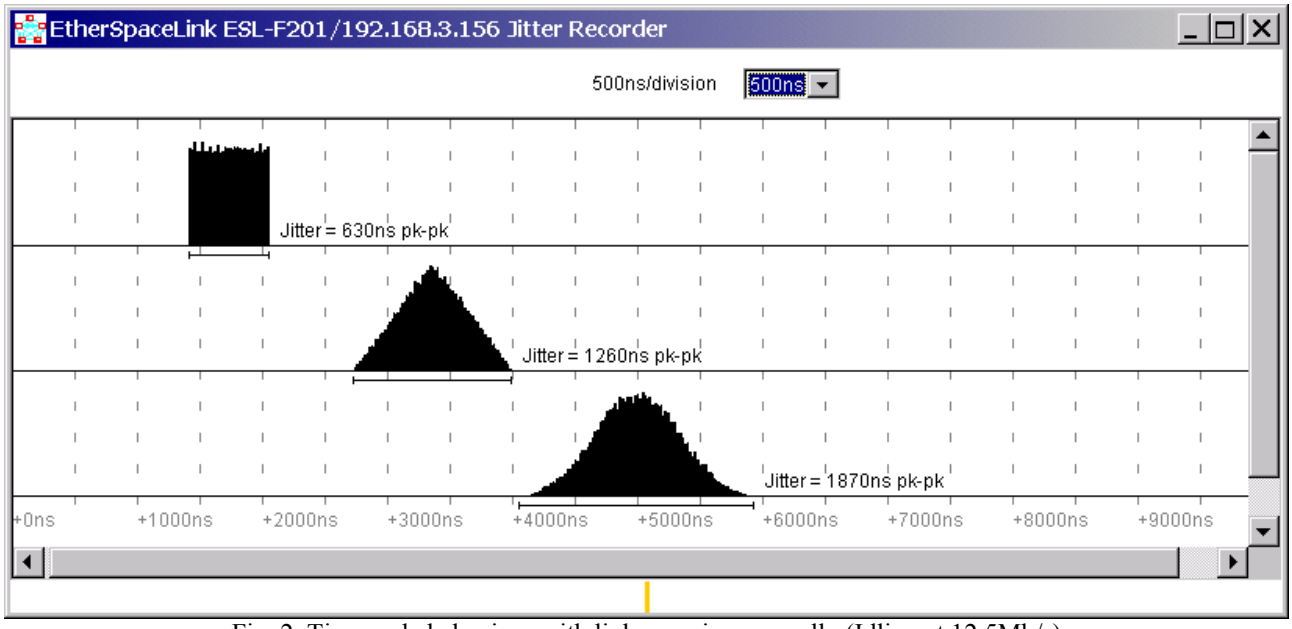


Fig. 2. Time code behaviour with links running normally (Idling at 12.5Mb/s)

All the SpaceWire links are idling, sending NULLs but no data. The transmit jitter introduced is thus only 8-bits, rather then up to 10-bits as would be found in an active link. Measured jitters should be scaled by 10 / 8 to get true values.

At 12.5Mb/s we have a bit time of 80ns and would expect to see a jitter of 8 * 80ns = 640ns. As seen in the top line of Fig. 2., our experiment confirms this. Traversing the second link doubles the jitter and the third link triples it. It is interesting to note that the distribution is rectangular at the first stage, triangular at the second and reaches a fair approximation of a normal distribution after three stages.

**LOW-LEVEL CODING – AN OPPORTUNITY TO DO BETTER**

SpaceWire uses a Gray-code on two pairs of wires to transfer both data and their clock signal. Fig. 3. shows a sample of the signals on the "Data" (top line) and "Strobe" (bottom line) lines. As can be seen the bits are sent at regular intervals – every 80ns in this instance, corresponding to a data rate of 12.5Mb/s. Time codes are inserted into the stream at the next available regular interval.

Such regular bit times are not, however, obligatory. A benefit of carrying the timing data in the signals is that bits can be conveyed at irregular intervals, so long as each is interval is long enough to be identified by the receiver and not so long as to trigger a disconnect timeout. We can use irregular intervals to remove the major source of time-code jitter.

The interval between a time-code request and its being sent varies depending on how long the transmitter is occupied sending its current token. This variable interval causes the large jitter observed.

It is not necessary to send a time-code as soon as it is possible to do so. We may delay sending the code, until some time after the transmitter is able to send it. By doing so we may arrange for the interval between the request for time-code transmission and its being sent to be constant – and jitter-free. See Fig. 4.

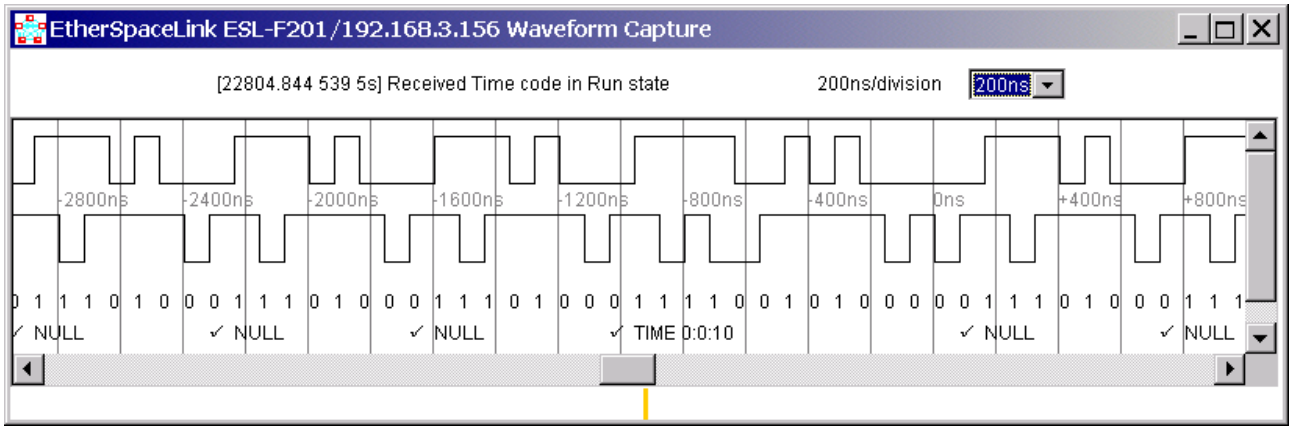

Fig. 3. Physical layer waveforms at 12.5Mb/s, showing an inserted time-code.

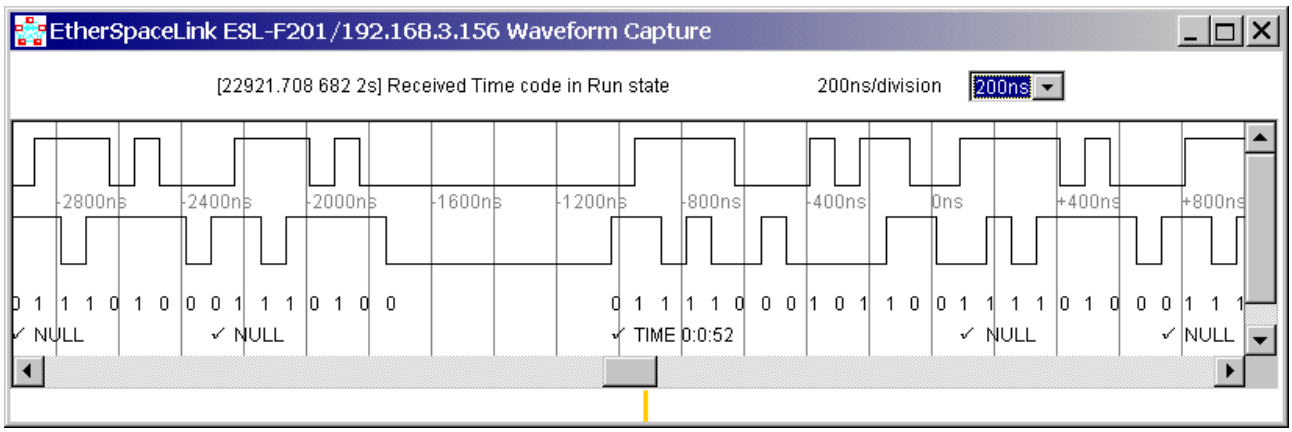

Fig. 4. Physical layer waveforms at 12.5Mb/s, showing a delayed inserted time-code.

The delay introduced must be as long as the longest token that may be being transmitted when the time-code request arrives – 10-bits – in order to be sure that it has been completely sent before the time-code must be sent. The period for which the data and strobe lines is held will be between zero (when the time-code request arrives exactly 10-bit times before the last token is completely sent) and ten bit times (when the last token finishes just as the time-code request arrives). In order to avoid a disconnect detection this delay must not exceed the timeout period of 850ns nominal. Taking a practical limit of 800ns over 10-bit times requires the link to be running at 80ns / bit or less – 12.5Mb/s or faster.

Time-code jitter is dramatically reduced by this technique, as can be seen in Fig. 5.

## ACHIEVABLE PERFORMANCE

Delayed time-code transmission totally removes the transmitter contribution to jitter leaving only the receiver to transmitter synchronization uncertainty. A jitter performance of only 10ns per link is easily achievable.

The cost of this approach is the introduction of up to 10 bit-periods of inactivity, average 5 bit-periods, on the data and strobe lines. It is almost certain that time-codes constitute only a small fraction of the traffic on a link and their extension from 12- to 17-bits (average) will have minimal affect on link throughput.

Although the effect is most dramatic at low speeds, it is still useful at higher speeds. A 200Mb/s link has a bit-period of 5ns and 10-bits of jitter amount to 50ns. Reducing this to 10ns (or less, depending on the link system clock speed) is still a worthwhile gain – especially when multiplied by the number of hops in a network.

Links are required to start at 10Mb/s [3] but time-codes are not permitted to be sent until the link is running, at which time the link speed can be increased to at least 12.5Mb/s.

Lower speed links can also benefit from a modified form of this technique, as described in the next section.

## REDUCING THE LOWER SPEED LIMIT

We assume, in the above calculation of the lower speed limit, that the delay is introduced only in one place – before the start of the time code. It would also be possible to add a little bit of delay to each bit of the time code itself to achieve a total of the required delay.

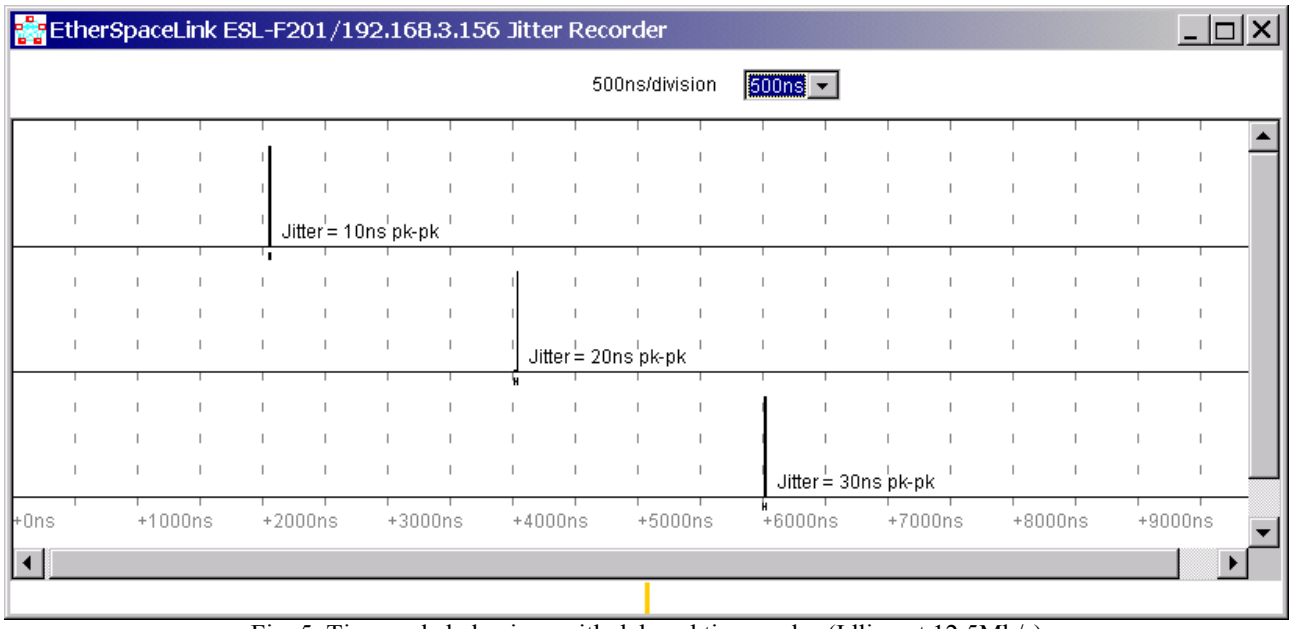


Fig. 5. Time code behaviour with delayed time-codes (Idling at 12.5Mb/s)

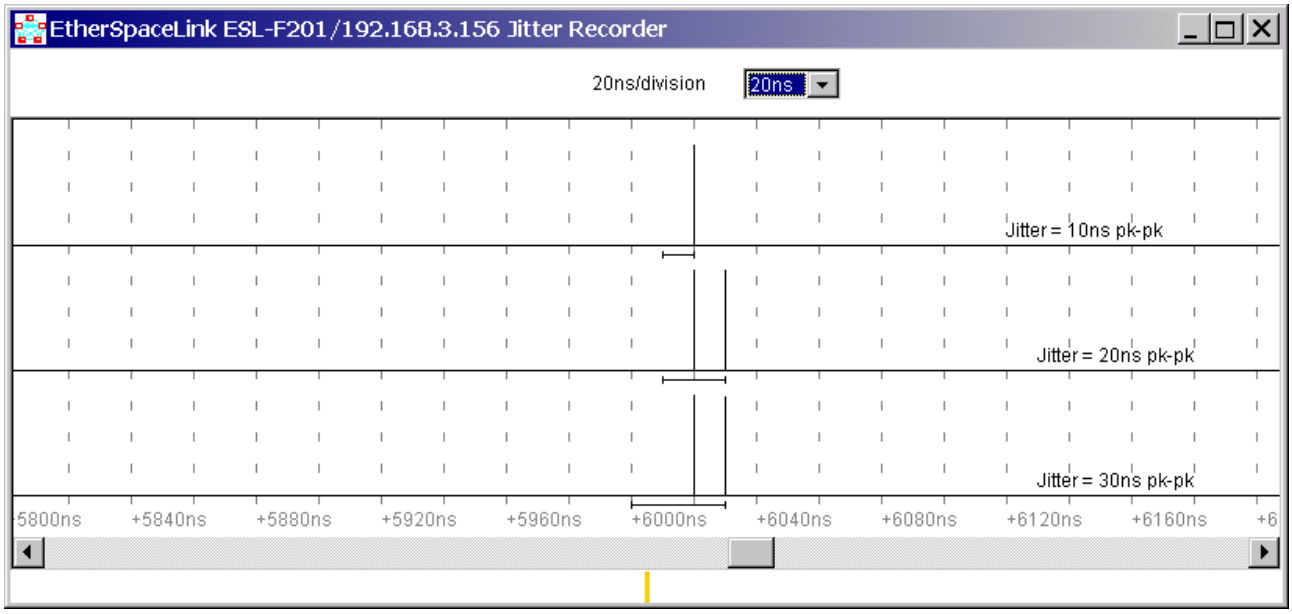


Fig. 6. Time code behaviour with output delays added

Before we can do this, however, we must be careful to consider when a time tick is generated from a received time code. All the added delay must occur before that point in time. It is not possible to know that a time-code has been correctly received until all its data bits and parity bit are seen and checked. At least ten bits are available for extension to give the required delay, allowing a link speed of down to 2.5Mb/s.

## GLOBAL SYNCHRONISATION

Given low-jitter times it is possible to consider synchronizing all nodes in the system. Some nodes receive their time-codes before others but the order of receipt will be deterministic – for a given time-code source.

It is thus possible to delay early ticks at the time-code output of the links so that all are produced at the same time. Fig. 6. shows the test system with ticks at point A delayed by 3970ns and those at point B delayed by 1990ns while those at point C are not delayed at all. The result of the delays is that all ticks are received at the same time, to within the jitter interval – note the much expanded time scale of this figure relative to those above. The delays added compensate for the latency of the routing switches and also of the connecting cables (at about 5ns/m these cannot be ignored).

If time-codes are sourced at a different point in the network (e.g. if the primary source fails and its function is taken by an alternate source) it will be necessary to change the output delays applied – inclusion of a writable register between the link receiver and tick output pin is suggested.

It might be preferable to fix the delay from time source to tick at *all* nodes, including the longest path, to a value longer than that of the longest path, *in all circumstances*. For example, we might set the tick arrival time to be, say, 25µs after it is sent and then to maintain that interval even if the time code source changes. This would avoid time jumps in the event of failure of, and recovery from, time master problems.

## CONCLUSION

We have shown, both theoretically and practically, that a small change in the time-code transmitter can give a very significant reduction in jitter. The receiver need not be altered, so long as it behaves deterministically.

Jitter performance of 10ns per link is easily achieved and significantly smaller jitters are possible.

It is possible to extend this benefit to produce accurately synchronized time signals at all nodes in a system.

**REFERENCES**


[1] ISO/IEC 14575 Heterogeneous InterConnect

[2] IEEE Std 1355-1995 IEEE Standard for Heterogeneous InterConnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction)

[3] ECSS-E-50-12A(24January2003) SpaceWire – Links, nodes, routers and networks