SpaceWire ECSS-E50-12A
International SpaceWire Seminar (ISWS 2003)

4-5 November 2003, ESTEC Noordwijk, The Netherlands

Steve Parkes [(1)], Josep Rosello [(2)]

[(1)] University of Dundee,
Applied Computing, Dundee, DD1 4HN, Scotland, UK.
Email:sparkes@computing.dundee.ac.uk

[(2)] European Space Agency
ESTEC, Noordwijk, The Netherlands
Email: josep.rosello@esa.int

ABSTRACT

SpaceWire[1] is a communications network for use onboard spacecraft. It is designed to connect high data-rate sensors, large solid-state memories, processing units and the downlink telemetry subsystem providing an integrated onboard, data-handling network. SpaceWire links are serial, high-speed (2 Mbits/sec to 400 Mbits/sec), bi-directional, full-duplex, point-to-point data links which connect together SpaceWire equipment. Application information is sent along a SpaceWire link in discrete packets. Control and time information can also be sent along SpaceWire links. SpaceWire is defined in the European Cooperation for Space Standardization ECSS-E50-12A standard.

SpaceWire is based on the “DS-DE” part of the IEEE-1355 standard [2] combined with the TIA/EIA-644 Low Voltage Differential Signalling (LVDS) standard [3]. Several problems with IEEE-1355 have been solved in the SpaceWire standard and connectors and cables suitable for space application are defined.

The SpaceWire standard is divided into several protocol levels:

- Physical Level which provides connectors, cables and EMC specifications.
- Signal Level which defines signal encoding, voltage levels, noise margins and data rates.
- Character Level which specifies the data and control characters used to manage the flow of data across a link.
- Exchange Level which covers the protocol for link initialisation, flow control, fault detection and link restart.
- Packet Level which details how a message is delivered from a source node to a destination node.

This paper provides an introduction to SpaceWire and the SpaceWire standard. It aims to build an understanding of the operation of a SpaceWire link by describing each protocol level in turn. The physical cable and connector, the LVDS signalling, the encoding of data and control characters, link initialization and the management of data flow across a SpaceWire link are explained. Finally the simple and flexible packet structure used by SpaceWire is described.

SPACEWIRE PURPOSE

SpaceWire [1] is a unified, high-speed data-handling network for connecting together sensors, processing elements, mass-memory units, downlink telemetry subsystems and EGSE equipment. SpaceWire provides a means of sending packets of information from a source node to a specified destination node, where a node is a hardware or software sub-system that wishes to use the services of the SpaceWire network. The SpaceWire standard defines high-speed data links and data-handling networks for handling payload data and control information onboard a spacecraft. It aims to meet the needs of future, high capability, remote sensing instruments and other space missions.

The purpose of the SpaceWire standard is:

- to facilitate the construction of high-performance on-board data handling systems,
- to help reduce system integration costs,
- to promote compatibility between data handling equipment and subsystems,

- to encourage re-use of data handling equipment across several different missions.

SpaceWire links are full-duplex, bi-directional point-to-point links that can operate at between 2 and 400 Mbits/s. Current radiation tolerant implementations are capable of up to 200 Mbits/s baud rate and a corresponding data-rate of 160 Mbits/s. This level of performance meets the demands of many current and future missions.

Integration and test of complex onboard systems is supported by SpaceWire with ground support equipment plugging directly into the onboard data handling system. Monitoring and testing can be carried out with a seamless interface into the onboard system.

Use of the SpaceWire standard ensures that equipment is compatible at both the component and sub-system levels. Processing units, mass-memory units and down-link telemetry systems using SpaceWire interfaces developed for one mission can be readily used on another mission reducing the cost of development, improving reliability and most importantly increasing the amount of scientific work that can be achieved within a limited budget.

SPACEWIRE STANDARD

The SpaceWire standard is specifically for use onboard spacecraft. It is based on two previous standards IEEE 1355-1995 [2] and ANSI/TIA/EIA-644 [3].

The standard covers the physical connectors and cables, electrical properties, and logical protocols that comprise SpaceWire data links and networks which are defined in the following normative protocol levels

- **Physical Level** – SpaceWire connectors, cables, cable assemblies and printed circuit board tracks.
- **Signal Level** – signal encoding, voltage levels, noise margins, and data signalling rates used in SpaceWire.
- **Character Level** – data and control characters used to manage the flow of data across a SpaceWire link.
- **Exchange Level** – protocols for link initialisation, flow control, link error detection and link error recovery.
- **Packet Level –** definition of how data for transmission over a SpaceWire link is split up into packets
- **Network Level** – structure of a SpaceWire network and the way in which packets are transferred from a source node to a destination node across a network. The network level also defines how link errors and network level errors are handled.

Each of these protocol levels will be considered in turn up to and including the packet level. The network level is described in [4] and time-codes are explained in [5].

PHYSICAL LEVEL

The physical level of the SpaceWire standard covers cables, connectors, cable assemblies and printed circuit board tracks. SpaceWire was developed to meet the EMC specifications of typical spacecraft.

Cables

The SpaceWire cable comprises four twisted pair wires with a separate shield around each twisted pair and an overall shield. To achieve a high data signalling rate with SpaceWire over distances up to 10 m a cable with the following characteristics is used:

- Characteristic impedance of 100 ohms differential impedance, which is matched to the line termination impedance.
- Low signal-signal skew between each signal in a differential pair and between Data and Strobe pairs.
- Low signal attenuation.
- Low cross-talk.
- Good EMC performance.

Connectors

The SpaceWire connector has eight signal contacts plus a screen termination contact. A nine pin micro-miniature D-type is specified as the SpaceWire connector. This type of connector is available qualified for space use.

Cable assemblies

SpaceWire cable assemblies are made from a length of SpaceWire cable of up to 10 m length terminated at each end by nine-pin micro-miniature D-type plugs.

Fig. 1 SpaceWire Cable Assembly

Printed circuit board tracks

SpaceWire can also be run over printed circuit boards (PCBs) including backplanes. The PCB tracks must have 100 ohm differential impedance.

SIGNAL LEVEL

The signal level part of the SpaceWire standard covers signal voltage levels, noise margins and signal encoding.

Signal level and noise margins

Low Voltage Differential Signalling (LVDS) as defined in ANSI/TIA/EIA-644 [3] is specified as the signalling technique for SpaceWire. LVDS uses balanced signals to provide very high-speed interconnection using a low voltage swing (350 mV typical). The balanced or differential signalling provides adequate noise margin to enable the use of low voltages in practical systems. The low voltage swing results in relatively low power consumption at high speed. LVDS is appropriate for connections between chips on a board, boards in a unit, and unit to unit interconnections over distances of 10 m or more. The LVDS signalling levels are illustrated in Fig. 2.

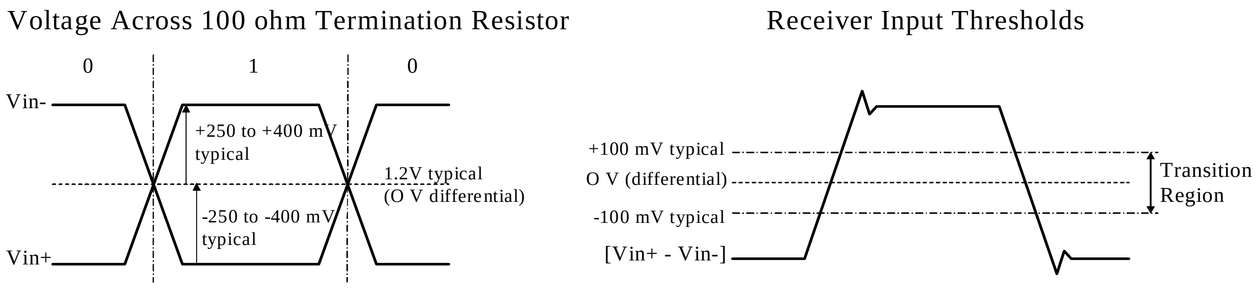


Fig. 2 LVDS signalling levels

A typical LVDS driver and receiver are shown in Fig. 3, connected by a media (cable or PCB traces) of 100 ohm differential impedance. The LVDS driver uses current mode logic. A constant current source of around 3.5 mA provides the current that flows out of the driver, along the transmission medium, through the 100 ohm termination resistance and back to the driver via the transmission medium. Two pairs of transistor switches in the driver control the direction of the current flow through the termination resistor. When the driver transistors marked “+” in Fig. 3 are turned on and those marked “-” are turned off, current flows as indicated by the arrows on the diagram creating a positive voltage across the termination resistor. When the two driver transistors, marked “-”, are turned on and those marked “+” are turned off, current flows in the opposite direction producing a negative voltage across the termination resistor. LVDS receivers are specified to have high input impedance so that most of the current flows through the termination resistor to generate around ±350 mV with the nominal 3.5 mA current source.

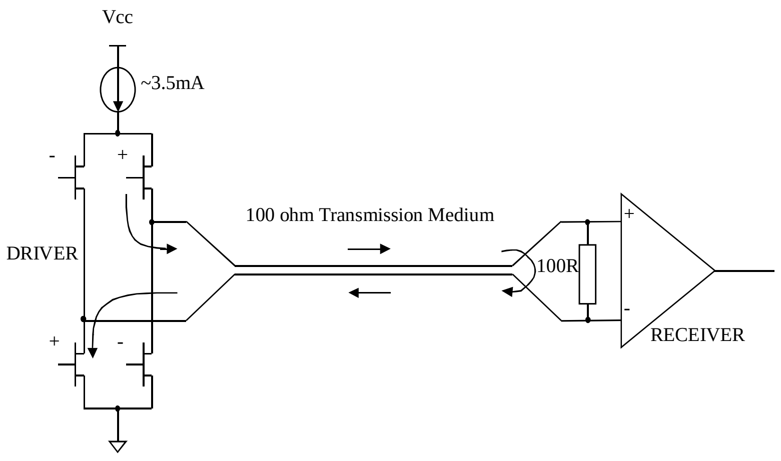


Fig. 3 LVDS Operation

LVDS has several features that make it very attractive for data signalling [6]:

- Near constant total drive current (+3.5 mA for logic 1 and -3.5 mA for logic 0) which decreases switching noise on power supplies.
- High immunity to ground potential difference between driver and receiver, LVDS can tolerate at least ±1 V ground difference.
- High immunity to induced noise because of differential signalling, normally using twisted-pair cable.
- Low EM emission because small equal and opposite currents create small electromagnetic fields which tend to cancel one another out.
- Not dependent upon particular device supply voltage(s).
- Simple 100 ohm termination at receiver.
- Failsafe operation - the receiver output goes to the high state (inactive) whenever
  - o the receiver is powered and the driver is not powered,
  - o the inputs short together,
  - o input wires are disconnected.
- Power consumption is typically 50 mW per driver/receiver pair for LVDS compared to 120 mW for ECL/PECL.

The signal levels and noise margins for SpaceWire are derived from ANSI/TIA/EIA-644 [3] which defines the driver output characteristics and the receiver input characteristics. The eye diagram for SpaceWire signals sent over 10 m of cable at 200 Mbits/s baud rate is shown in Fig. 4.

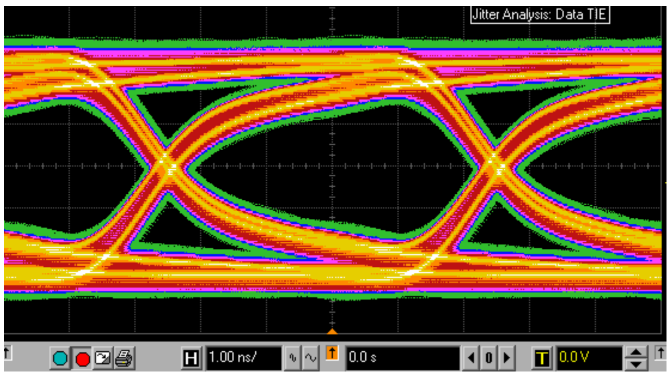


Fig. 4 SpaceWire Eye Diagram (200 MHz, 10 m cable)

Data encoding

SpaceWire uses Data-Strobe (DS) encoding. This is a coding scheme which encodes the transmission clock with the data into Data and Strobe so that the clock can be recovered by simply XORing the Data and Strobe lines together. The data values are transmitted directly and the strobe signal changes state whenever the data remains constant from one data bit interval to the next. This coding scheme is illustrated in Fig. 5. The DS encoding scheme is also used in the IEEE 1355-1995 [1] and IEEE 1394-1995 (Firewire) standard [7].

The reason for using DS encoding is to improve the skew tolerance to almost 1-bit time, compared to 0.5 bit time for simple data and clock encoding.

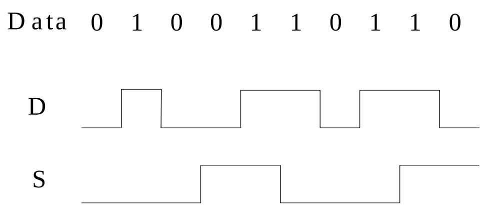


Fig. 5 Data-Strobe (DS) Encoding

A SpaceWire link comprises two pairs of differential signals, one pair transmitting the D and S signals in one direction and the other pair transmitting D and S in the opposite direction. That is a total of eight wires for each bi-directional link.

CHARACTER LEVEL

SpaceWire is based on the character level protocol defined in IEEE 1355-1995 [2]. Time-codes characters have been added to support the distribution of system time [5].

There are two types of characters: data and control characters which are illustrated in Fig. 6.

- Data characters which hold an eight-bit data value transmitted least-significant bit first. Each data character contains a parity-bit, a data-control flag and the eight-bits of data. The parity-bit covers the previous eight-bits of data or two-bits of control code, the current parity-bit and the current data-control flag. It is set to produce odd parity so that the total number of 1’s in the field covered is an odd number. The data-control flag is set to zero to indicate that the current character is a data character.
- Control characters which hold a two-bit control code. Each control character is formed from a parity-bit, a data-control flag and the two-bit control code. The data-control flag is set to one to indicate that the current character is a control character. Parity coverage is similar to that for a data character. One of the four possible control characters is the escape code (ESC). This can be used to form longer control codes. Two longer control codes are specified and valid which are the NULL code and the Time-Code.

In addition to the data and control characters there are two control codes: NULL and time-codes.

- NULL is formed from ESC followed by the flow control token (FCT). NULL is transmitted, whenever a link is not sending data or control tokens, to keep the link active and to support link disconnect detection.
- The Time-Code is used to support the distribution of system time across a network. A Time-Code is formed by ESC followed by a single data-character.

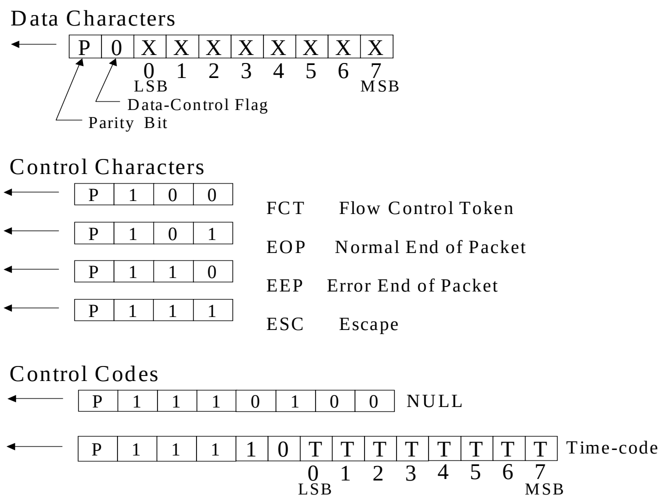


Fig. 6 Data and Control Characters and Control Codes

EXCHANGE LEVEL

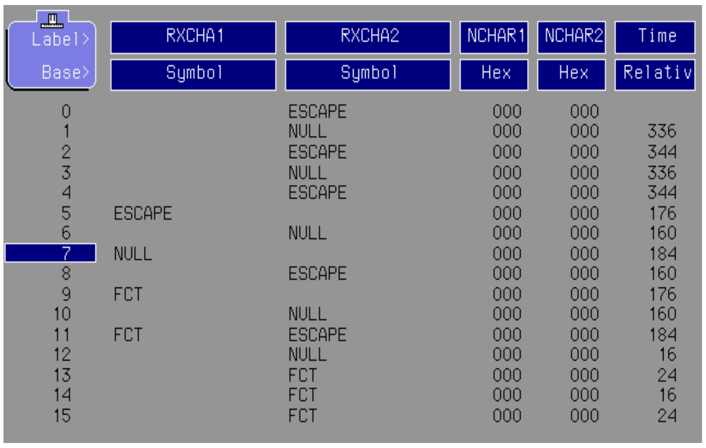
The exchange level protocol is a significantly more capable version than that defined in IEEE 1355-1995 [2] which fixes several problems with IEEE 1355. Backwards compatibility with IEEE 1355 has been maintained with the exception of time-codes. It is possible to use a SpaceWire link connected to an IEEE 1355 interface provided that time-codes are not sent to the IEEE 1355 interface.

The SpaceWire exchange level provides the following services: initialisation, flow-control and error-handling.

Link Initialisation

Following reset the link output is held in the reset state until it is instructed to start and attempt to make a connection with the link interface at the other end of the link. A connection is made following a handshake that ensures both ends of the link are able to send and receive characters successfully. Each end of the link sends NULLs, waits to receive a NULL, then sends FCTs and waits to receive an FCT. Since a link interface cannot send FCTs until it has received a NULL, receipt of one or more NULLs followed by receipt of an FCT means that the other end of the link has received NULLs successfully and that full connection has been achieved.

The link initialisation sequence is shown in Fig. 7. The sequence of characters under RXCHA1 is for one direction of the link while that under RXCHA2 is the other direction. The NULL control code is shown as ESCAPE followed by NULL in this figure. RXCHA2 starts sending NULLS. These are received by RXCHA1 which sends a NULL and two FCTs in response. RXCHA2 receives the NULLs and FCTs from RXCHA1 and starts sending FCTs. RXCHA2 is now ready to send data. When RXCHA1 receives an FCT it will also be ready to send data and the link is fully initialised.



Label>	RXCHA1	RXCHA2	NCHAR1	NCHAR2	Time
Base>	Symbol	Symbol	Hex	Hex	Relativ
0		ESCAPE	000	000	
1		NULL	000	000	336
2		ESCAPE	000	000	344
3		NULL	000	000	336
4		ESCAPE	000	000	344
5	ESCAPE		000	000	176
6		NULL	000	000	160
7	NULL		000	000	184
8		ESCAPE	000	000	160
9	FCT		000	000	176
10		NULL	000	000	160
11	FCT	ESCAPE	000	000	184
12		NULL	000	000	16
13		FCT	000	000	24
14		FCT	000	000	16
15		FCT	000	000	24

Fig. 7 Link Initialisation

Link Flow Control

A transmitter is only allowed to transmit N-Chars (normal characters, which are data characters, EOP or EEP) if there is space for them in the host system receive buffer at the other end of the link. The host system indicates that there is space for eight more N-Chars by requesting the link transmitter to send a flow control token (FCT). The FCT is received at the other end of the link (end B) enabling the transmitter at end B to send up to eight more N-Chars. If there is more room in the host receive buffer then multiple FCTs can be sent, one for every eight spaces in the receive buffer. Correspondingly, if multiple FCTs are received then it means that there is a corresponding amount of space available in the receiver buffer e.g. four FCTs means that there is room for 32 N-Chars.

Link Error Handling

Link disconnection is detected when following reception of a data bit no new data bit is received within a link disconnect timeout window (850 ns). Once a disconnection error has been detected, the link attempts to recover from the error.

Parity errors occurring within a data or control character are detected when the next character is sent, since the parity bit for a data or control character is contained in the next character. Once a parity error has been detected, the link attempts to recover from the error.

Following an error or reset the link attempts to re-synchronise and restart using an "exchange of silence" protocol (see Fig. 8). The end of the link that is either reset or that finds an error, ceases transmission. This is detected at the other end of the link as a link disconnect and that end stops transmitting too. The first link resets its input and output for 6.4 µs to ensure that the other end detects the disconnect. The other end also waits for 6.4 µs after ceasing transmission. Each link then waits a further 12.8 µs before starting to transmit. These periods of time are sufficient to ensure that the receivers at both ends of the link are ready to receive characters before either end starts transmission. The two ends of the link go through the NULL/FCT handshake (see Fig. 7) to re-establish a connection and ensure proper character synchronisation.

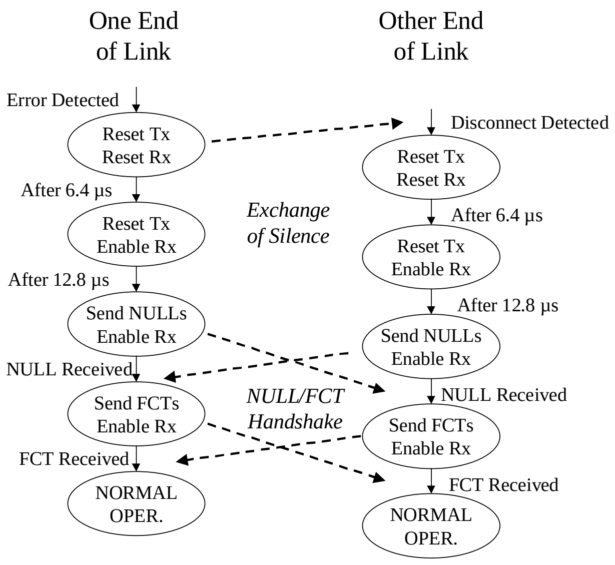


Fig. 8 Link Restart

PACKET LEVEL

The packet level protocol follows the packet level protocol defined in IEEE 1355-1995 [1]. It defines how data is encapsulated in packets for transfer from source to destination. The format of a packet is as follows:

<Destination Address> <Cargo> <End of Packet Marker>

The "Destination Address" is a list of zero or more data characters that represent the destination identity. This list of data characters represents either the identity code of the destination node or the path that the packet takes to get to the destination node. The destination address is used by SpaceWire routers [4] to direct a packet towards its destination node. Over a single point-to-point link there is no need to use a destination address.

The “Cargo” is the data to transfer from source to destination.

The “End of Packet Marker” is used to indicate the end of a packet. Two end of packet markers are defined.

- EOP normal end_of_packet marker - indicates end of packet
- EEP error end_of_packet marker - indicates that the packet has been terminated prematurely due to a link error.

Since there is no start_of_packet marker, the first data character following an end_of_packet marker (either EOP or EEP) is regarded as the start of the next packet.

CONCLUSION

This paper has introduced the various protocol levels of the SpaceWire standard covering the physical, signal, character, exchange and packet levels. SpaceWire is now being designed into several missions by both ESA and NASA. Radiation tolerant components and validated IP cores will soon be available, along with development, test and electronic ground support equipment.

ACKNOWLEDGEMENTS

The SpaceWire standard resulted from the work of the SpaceWire working group comprising many people across many countries, companies and organisations the authors are indebted to them for their continuous encouragement, participation and essential comments and corrections to the SpaceWire standard.

REFERENCES

[1] S.M. Parkes et al, “SpaceWire: Links, Nodes, Routers and Networks,” *European Cooperation for Space Standardization, Standard No. ECSS-E50-12A,Issue 1, January 2003.*

[2] IEEE Computer Society, *“IEEE Standard for Heterogeneous Interconnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction)”, IEEE Standard 1355-1995, IEEE, June 1996.*

[3] Tele-communications Industry Association, “Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits,” *Standard ANSI/TIA/EIA-644 1995, March 1996.*

[4] S.M. Parkes, C. McClements, G. Kempf, S. Fischer and A. Leon, “SpaceWire Router,” *International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003*

[5] S.M. Parkes, “The Operation and Uses of the SpaceWire Time-Code,” *International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003*

[6] S.M. Parkes, “High-Speed, Low-Power, Excellent EMC: LVDS for On-Board Data Handling”, *Proceedings of the 6th International Workshop on Digital Signal Processing Techniques for Space Applications, ESTEC, Sept. 1998.*

[7] IEEE Computer Society, “IEEE Standard for a High Performance Serial Bus”, *IEEE Standard 1394-1995, IEEE, August 1996*

.