Developing and Testing SpaceWire Devices and Networks

Steve Parkes and Stuart Mills

STAR-Dundee Ltd, STAR House, 166 Nethergate, Dundee, DD1 4EE, Scotland, UK.

Email: steve.parkes@star-dundee.com; stuart.mills@star-dundee.com

ABSTRACT

1 INTRODUCTION

SpaceWire is a data-handling network for use on-board spacecraft, which connects together instruments, mass-memory, processors, downlink telemetry, and other on-board sub-systems [1]. SpaceWire is simple to implement and has some specific characteristics that help it support data-handling applications in space: high-speed, low-power, simplicity, relatively low implementation cost, and architectural flexibility making it ideal for many space missions. SpaceWire provides high-speed (2 Mbits/s to 200 Mbits/s), bi-directional, full-duplex data-links, which connect together SpaceWire enabled equipment. Data-handling networks can be built to suit particular applications using point-to-point data-links and routing switches.

Since the SpaceWire standard was published in January 2003, it has been adopted by ESA, NASA, JAXA and RosCosmos for many missions and is being widely used on scientific, Earth observation, commercial and other spacecraft. High-profile missions using SpaceWire include: Gaia, ExoMars rover, Bepi-Colombo, James Webb Space Telescope, GOES-R, Lunar Reconnaissance Orbiter and Astro-H.

The development and testing of the SpaceWire links and networks used on these and many other spacecraft currently under development, requires a comprehensive array of test equipment. In this paper the requirements for test equipment fulfilling key test functions are outlined and then equipment that meets these requirements is described. Finally the all-important software that operates with the test equipment is introduced.

2 DEVELOPING AND TESTING SPACEWIRE SYSTEMS

SpaceWire is currently being used both as a point-to-point link and as a network with routing switches interconnecting SpaceWire units. To support the development of SpaceWire systems several different types of test equipment are required. A SpaceWire interface board or unit can provide a test PC with SpaceWire capability so it can be used with associated software to exercise and test SpaceWire units under development. For real-time emulation of high-speed SpaceWire units or those that have some tricky timing constraints that cannot be met readily with an interface card and real-time software, a hardware-based emulation unit is required. To check that a link is operating correctly, to inject errors, and to check that application software is behaving as it should when sending and receiving SpaceWire packets, a Link Analyser is required that can monitor, record and analyse information flowing on the SpaceWire link. When working with a complete SpaceWire network it is important to be able to relate packets flowing over several SpaceWire links. A SpaceWire multi-link recorder is required that can collect possibly hours of data from several links in a network for subsequent analysis and display.

The principal requirements of these different types of test equipment are outlined below:

- SpaceWire Interface:
  - High performance for both large and small packets;
  - Various form factors (PCI, PCIe, cPCI, PXI, USB, etc.);
  - Comprehensive driver support;
  - Easy to move from one operating system to another;
  - Easy to move from one type of interface board to another without having to re-write the software.

- SpaceWire Equipment Emulator:
  - Operate in real-time at full link speeds while sending large or small packets;
  - High resolution timing accuracy;
  - Simple, rapid means of specifying the packets to be sent and the related timing;
  - Emulate SpaceWire behaviour of an instrument in response to external signals, SpaceWire commands and host software.

- SpaceWire Link Analyser:
  - Unobtrusive monitoring of a SpaceWire link;
  - Large recording memory;
  - Different views of recorded data, including low-level character view and high-level packet view;
  - Ability to decode SpaceWire packets that carry protocols like RMAP.
  - Configurable triggering options including pre and post trigger, triggering on an error, triggering on defined data, triggering on a sequence of events.

- SpaceWire Recorder
  - Record data from several SpaceWire links;
  - High overall data collection rate to memory;
  - Large capacity memory, able to store several hours of network data;
  - Pre and post triggering capabilities;
  - Display and network traffic analysis software for displaying the data gathered in a meaningful way.

3 SPACEWIRE DEVELOPMENT AND TEST EQUIPMENT

In this section new SpaceWire test equipment from STAR-Dundee Ltd. that addresses the requirements given in section 3 are described.

3.1 Interfacing to SpaceWire Devices

STAR-Dundee has a comprehensive range of interface devices for SpaceWire including PCI, PCIe, cPCI, PXI and USB 2.0 devices. A new version of the widely used SpaceWire-USB Brick device will shortly be available along with a versatile cPCI/PXI board.

3.1.1 cPCI/PXI MkIII

The SpaceWire cPCI/PXI MkIII board is an extremely versatile platform for SpaceWire test equipment. Variants of this board can be used to provide:

- A four port SpaceWire cPCI/PXI interface;
- An eight or more port router;
- A rack mounted link analyser;
- A four link SpaceWire recorder;
- SpaceWire to SpaceFibre bridge;
- Many other test and development boards.

The cPCI MkIII board comprises a 3U cPCI/PXI board containing an FPGA, a selection of flexible interface boards, a custom front panel for that set of interfaces. The FPGA on the board is configured for a particular application, the required flexible interface boards added, and a front panel provided for those interfaces. For example an 8-port SpaceWire router board would have eight SpaceWire interface boards. A photograph of the cPCI/PXI MkIII board is shown in Figure 3-1.

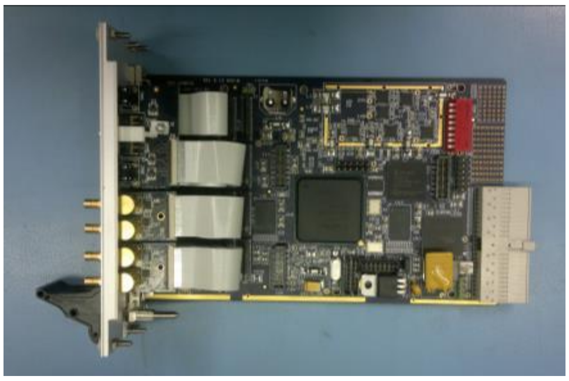


Figure 3-1 Photograph of cPCI/PXI MkIII board

3.1.2 Brick MkIII

The Brick MKIII will be described in detail in the full paper.

3.2 Real-Time Instrument Emulation

The SpaceWire EGSE is a real-time SpaceWire instrument emulator. It has been designed to rapidly emulate many different SpaceWire instruments in real-time with microsecond accuracy. A typical emulation takes about one day to develop. A special SpaceWire emulation language has been written that describes the SpaceWire packets to be sent, the link speed, actual data-rate, and timing of a sequence of packets, and the internal state of the instrument. For example in a “stand-by” state the instrument may be waiting to start sending packets. Every ten seconds it may enter an automatic housekeeping “keep-alive” indication state, where it sends a short SpaceWire packet to a central controller to indicate that the instrument is still alive. When a command is received via SpaceWire or via some other means the instrument enters an “active” state sending SpaceWire packets according to a predefined schedule. The SpaceWire EGSE language can be used to rapidly describe instrument behaviour including the schedule of packets to be sent and the detailed contents of those packets.

Once the emulation script has been written it is compiled and downloaded into the EGSE hardware which executes the required behaviour in real-time. When configured, the SpaceWire EGSE hardware operates independently of the software resulting in real-time performance. The EGSE hardware can interact with other emulation software using software events that are passed via an API to the EGSE unit. The software event can cause a state transition in the EGSE unit, changing the perceived behaviour of the emulated instrument. External hardware triggers can be used in a similar way.

The SpaceWire EGSE unit is illustrated in Figure 3-2. As can be seen in the photograph the SpaceWire EGSE has two SpaceWire ports so is capable of emulating two separate SpaceWire instruments concurrently.

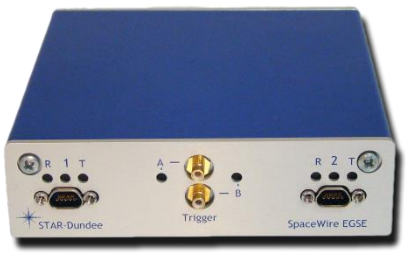


Figure 3-2 SpaceWire EGSE front view

An example of the SpaceWire emulation language is shown in Figure 3-3. The simplicity of the language and the ease with which it can be understood and used to create fairly sophisticated instrument emulations is apparent from this short example.

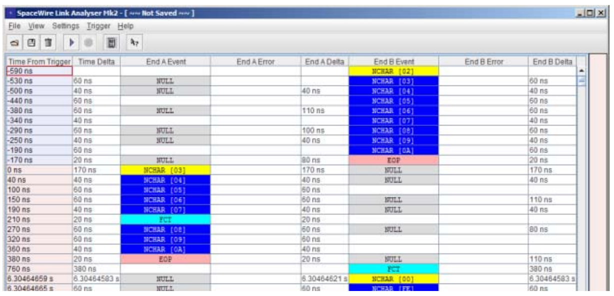
```
Send Multiple Packets
# Packet Declarations
packet myPkt
   hex(0A 0B 0C 0D)
   eop
end packet

packet myPktWithCRC
   start(crc8)
   hex(0A 0B 0C 0D)
   stop(crc8)
   crc8
   eop
end packet

# Schedule Declaration
schedule mySchedule
   5ms send myPkt * 2
   10ms send myPktWithCRC * 2
end schedule

# State machine declaration
statemachine 1
   initial state state1 @ 100Mbps
      do mySchedule repeatedly
      transition at end of schedule
   end state
end statemachine
```

Figure 3-3 SpaceWire EGSE scripting language

3.3 Link Analysis

The SpaceWire Link Analyser Mk2 provides several capabilities for testing a SpaceWire and debugging hardware and related software. The link analyser is inserted in the SpaceWire link between two SpaceWire units. It monitors the link and records data when a trigger event occurs. The data can be captured and displayed at various levels.

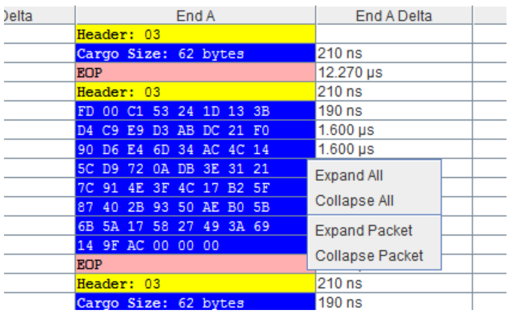
Link level trace: Monitoring, tracing and recording traffic at the link level. This may be used to confirm SpaceWire link start-up, flow-control, data transfer, and error recovery. An example is shown in Figure 3-4.



Time From Trigger	Time Delta	End A Event	End A Error	End A Delta	End B Event	End B Error	End B Delta
-590 ns					NCHAR [02]		
-530 ns	60 ns	NULL			NCHAR [03]		60 ns
-500 ns	40 ns	NULL		40 ns	NCHAR [04]		40 ns
-440 ns	60 ns				NCHAR [05]		60 ns
-380 ns	60 ns	NULL		110 ns	NCHAR [06]		60 ns
-340 ns	40 ns				NCHAR [07]		40 ns
-290 ns	60 ns	NULL		100 ns	NCHAR [08]		60 ns
-250 ns	40 ns	NULL		40 ns	NCHAR [09]		40 ns
-190 ns	60 ns				NCHAR [0A]		60 ns
-170 ns	20 ns	NULL		80 ns	EOP		20 ns
0 ns	170 ns	NCHAR [03]		170 ns	NULL		170 ns
40 ns	40 ns	NCHAR [04]		40 ns	NULL		40 ns
100 ns	60 ns	NCHAR [05]		60 ns			
150 ns	60 ns	NCHAR [06]		60 ns	NULL		110 ns
190 ns	40 ns	NCHAR [07]		40 ns	NULL		40 ns
210 ns	20 ns	FCT		20 ns			
270 ns	60 ns	NCHAR [08]		60 ns	NULL		80 ns
320 ns	60 ns	NCHAR [09]		60 ns			
360 ns	40 ns	NCHAR [0A]		40 ns			
380 ns	20 ns	EOP		20 ns	NULL		110 ns
760 ns	380 ns				FCT		380 ns
6.30464659 s	6.30464583 s	NULL		6.30464621 s	NCHAR [00]		6.30464583 s
6.30464665 s	60 ns	NULL		60 ns	NCHAR [FE]		60 ns

Figure 3-4 Example link level display

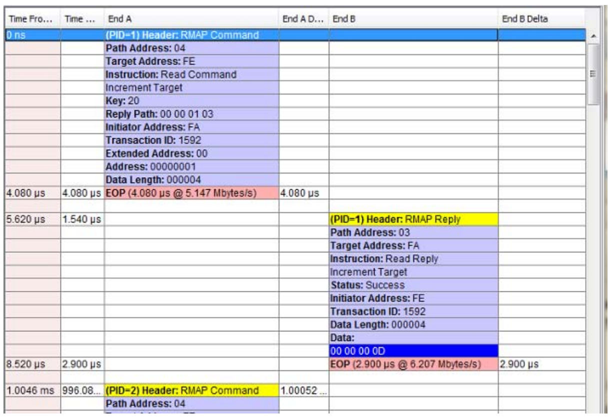
Packet level trace: Monitoring, tracing and recording of SpaceWire packets. This is used to monitor the flow of packets exchanged across a SpaceWire link, the response of a system to packet errors, and the control of SpaceWire systems using control packets. An example is shown in Figure 3-5.



elta	End A	End A Delta
	Header: 03	
	Cargo Size: 62 bytes	210 ns
	EOP	12.270 µs
	Header: 03	210 ns
	FD 00 C1 53 24 1D 13 3B	190 ns
	D4 C9 E9 D3 AB DC 21 F0	1.600 µs
	90 D6 E4 6D 34 AC 4C 14	1.600 µs
	5C D9 72 0A DB 3E 31 21	
	7C 91 4E 3F 4C 17 B2 5F	
	87 40 2B 93 50 AE B0 5B	
	6B 5A 17 58 27 49 3A 69	
	14 9F AC 00 00 00	
	EOP	
	Header: 03	210 ns
	Cargo Size: 62 bytes	190 ns

Figure 3-5 Detail of packet level display

RMAP analysis: Display of RMAP transactions [2] in a readily understandable form with each field of the protocol data unit clearly identified, simplifying RMAP analysis immensely. This is illustrated in Figure 3-6.



Time Fro...	Time ...	End A	End A D...	End B	End B Delta
0 ns		(PID=1) Header: RMAP Command			
		Path Address: 04			
		Target Address: FE			
		Instruction: Read Command			
		Increment Target			
		Key: 20			
		Reply Path: 00 00 01 03			
		Initiator Address: FA			
		Transaction ID: 1592			
		Extended Address: 00			
		Address: 00000001			
		Data Length: 000004			
4.080 µs	4.080 µs	EOP (4.080 µs @ 5.147 Mbytes/s)	4.080 µs		
5.620 µs	1.540 µs			(PID=1) Header: RMAP Reply	
				Path Address: 03	
				Target Address: FA	
				Instruction: Read Reply	
				Increment Target	
				Status: Success	
				Initiator Address: FE	
				Transaction ID: 1592	
				Data Length: 000004	
				Data:	
				00 00 00 0D	
8.520 µs	2.900 µs			EOP (2.900 µs @ 6.207 Mbytes/s)	2.900 µs
1.0046 ms	996.08...	(PID=2) Header: RMAP Command	1.00052 ...		
		Path Address: 04			

Figure 3-6 Interpreting RMAP transactions

Recording of events is started by a trigger condition, which can be a specific event (e.g. disconnect, EEP, external trigger), after a character sequence (e.g. EOP plus header/address of a packet), or after a specific sequence of events. Pre- and post- event triggering are supported with the trigger position being indicated on the traffic display.

To support the testing and debugging of FPGAs and other board-level units with SpaceWire interfaces, a 38pin Mictor logic analyser connector is provided on the rear panel of the Link Analyser Mk2. The SpaceWire traffic in each direction of the link is decoded into a set of characters which are provided on the logic analyser connector. The logic analyser can have one pod monitoring the SpaceWire information from the Link Analyser and other pods monitoring pins on the FPGA or other components of the board. This dramatically simplifies the testing of units whose function may be affected by SpaceWire commands, or units that send SpaceWire packets on particular conditions.

3.4 Recording Network Traffic

The SpaceWire Recorder is designed to support the validation and debugging of complete SpaceWire systems. The SpaceWire Recorder is shown in Figure 3-7.

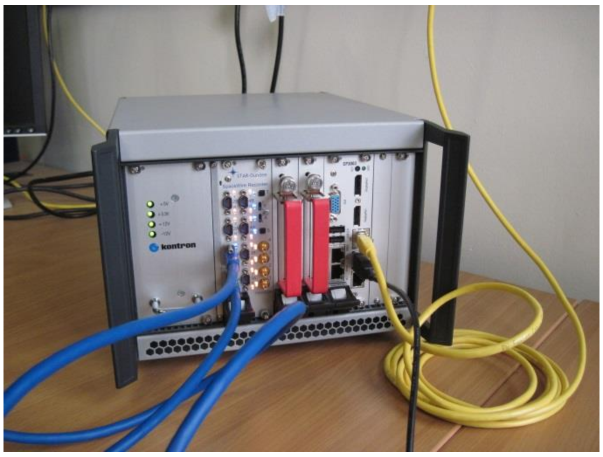


Figure 3-7 SpaceWire Recorder front view

The SpaceWire Recorder unit comprises a small 3U PXI/cPCI rack, fitted with a SpaceWire interface board, a control computer and one or more SATA solid state disc drives. The SpaceWire interface board has eight SpaceWire interfaces which are able to support recording of SpaceWire packets running in both directions of up to four links. Each SpaceWire link can operate at up to 200 Mbits/s link speed. The control computer provides a range of interfaces to the SpaceWire recorder including dual Gigabit Ethernet for remote control or data logging. The SpaceWire Recorder is fitted as standard with one 480 Gbyte SATA solid state disc drive to give high recording data-rate and reliability. A block diagram of the SpaceWire Recorder unit is illustrated in Figure 3-5.

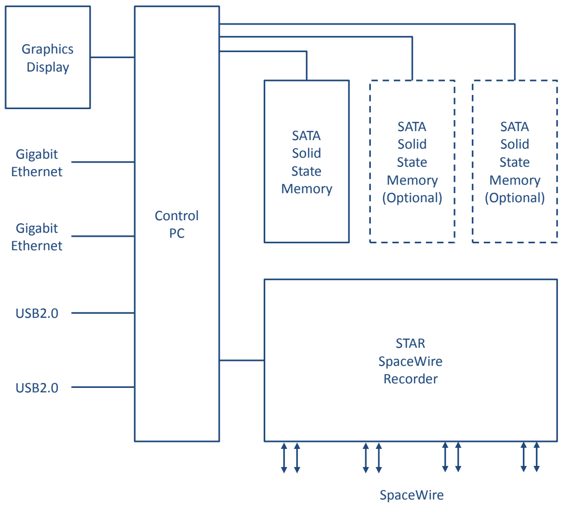


Figure 3-8 SpaceWire Recorder block diagram

4 SPACEWIRE SOFTWARE

Software for developing, testing and validating SpaceWire systems can be expensive to develop, especially when fast data-rates are required. In this section the new STAR-System software is explored.

4.1 STAR-System

STAR-System is the API and driver system provided with STAR-Dundee's new range of SpaceWire interface and router devices. It presents a common interface across operating systems, and allows different types of SpaceWire devices to be accessed in a consistent manner.

4.1.1 Application Programming Interface

A full API is provided to allow all functions of supported devices to be controlled from application software. Both C and C++ versions of the API are provided, which also allows the API to be imported into other languages. LabVIEW support is available to allow easy access to devices from LabVIEW applications. It is planned that a variety of programming languages will be supported in the future, including Python, .NET and Java.

The API is common across several of STAR-Dundee's products, and is consistent for each programming language and platform supported. This simplifies software development and allows migration of test software from one device to another and from one platform to another, enhancing software reuse and reducing the risk of schedule delays. The API also takes advantage of features provided by each language, making the API as efficient and easy to use as possible.

A key feature of the API is that it not only provides functionality to transmit and receive packets, but also includes functions required when testing equipment. For example, the API makes it simple to transmit packets terminated with an EEP, and to determine the end of packet marker type of received packets. It simplifies the process of transmitting a stream of traffic (for example, from a file), and receiving a stream of traffic. If these streams were to include time-codes in the middle of packets, the position of these time-codes would be maintained by the API. This allows traffic streams to be recorded accurately and replayed, so multiple test runs will be consistent.

STAR-System is not only packed with features, it also provides these features with very high performance and low CPU usage. The graph below shows data rates and CPU usage when transmitting and receiving packets of various sizes on a 200 Mbits/s link. With packets as small as 50 bytes in length, the data rate achieved is close to the theoretical maximum of slightly less than 160 Mbits/s.

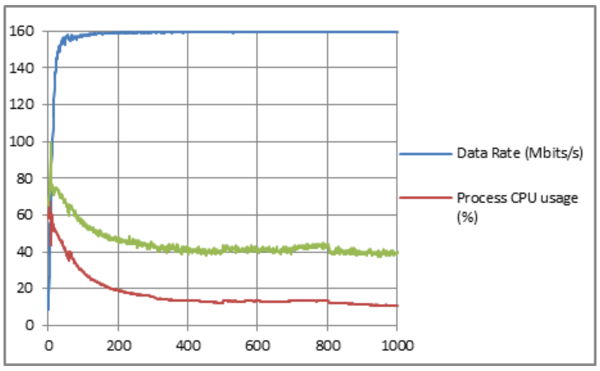


Figure 4-1 STAR-System performance

The API is provided with extensive searchable documentation and example code for each API function. Example test programs with source code are included for each language supported.

4.1.2 Supporting Modules

STAR-System is a modular system. This reduces the footprint of the system, which can be important where memory or storage is at a premium. If validation of the API is required, this also reduces the number of functions which must be validated.

In addition to the core components of the system, modules are provided to implement commonly required features. For example, users working with the Remote Memory Access Protocol can use the RMAP Packet Library to simplify the building and interpreting of RMAP packets.

4.1.3 LabVIEW

The LabVIEW software development environment (a visual, dataflow programming language provided by National Instruments Corporation) is supported by STAR-System via a LabVIEW wrapper. Features provided include transmitting and receiving SpaceWire traffic, creating RMAP packets, and performing device configuration. All functions provided by the STAR-System C API can be accessed through the LabVIEW wrapper.

The STAR-System LabVIEW wrapper provides example VIs demonstrating typical use cases for the STAR-System API. Examples provided include:

- Transmitting and receiving packets.
- Receiving time-codes and link events (speed change/state change).
- Building and decoding RMAP packets.
- Getting hardware information.
- Error injection.
- Building routing tables.
- Setting link speeds.
- RMAP Target and Initiator.

The library has been designed to be intuitive to LabVIEW users; with device access following the "Open, Perform Action, Close" architecture, and callback/notification functions implemented using LabVIEW user events. Example LabVIEW source code is shown in Figure 4-2.

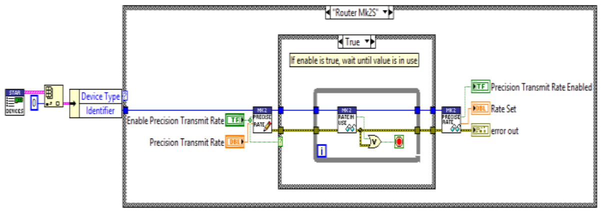


Figure 4-2 Source code for setting precision transmit rate of a Router Mk2S device

5 CONCLUSIONS

Outline requirements for SpaceWire equipment to fulfil the needs of various test and development activities have been provided. A range of SpaceWire test and development equipment has been described that meets these requirements. The full paper will provide further information on the Brick MKIII device.

6 REFERENCES

[1] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008, available from http://www.ecss.nl.

[2] ECSS Standard ECSS-E-ST-50-52C, "SpaceWire – Remote memory access protocol", Issue 1, European Cooperation for Space Data Standardization, 5 February 2010, available from http://www.ecss.nl.