SpaceWire EGSE

Session: SpaceWire Test and Verification

Short Paper

Stephen Mudie, Paul E. McKechnie

STAR-Dundee, c/o University of Dundee, School of Computing, Dundee, DD1 4HN

Steve Parkes, Martin Dunstan

University of Dundee, School of Computing, Dundee, DD1 4HN

E-mail: stephen.mudie@star-dundee.com, paul.mckechnie@star-dundee.com, sparkes@computing.dundee.ac.uk, mdunstan@computing.dundee.ac.uk

Abstract

The SpaceWire Electronic Ground Support Equipment (EGSE) is a STAR-Dundee product [1] designed to simulate and stimulate SpaceWire devices. It provides a means of generating user defined packets in pre-defined sequences at specific times and data rates. The SpaceWire EGSE is configured using a script that is compiled and loaded onto the SpaceWire EGSE unit. Once configured, the EGSE can generate complex SpaceWire packet sequences without further interaction from host PC software.

Real-time SpaceWire Electronic Ground Support Equipment can be implemented easily with the SpaceWire-EGSE unit, avoiding the need for complex and expensive, real-time software development.

1 Introduction

SpaceWire Electronic Ground Support Equipment (EGSE) is needed to support the integration and testing of spacecraft that use SpaceWire. The EGSE has to simulate instruments and other equipment during integration and test, and has to do this with similar if not identical timing. Furthermore the SpaceWire EGSE has to integrate with other test equipment, either responding to events or triggering other pieces of equipment. Typically SpaceWire EGSE is implemented using a SpaceWire interface board in a rack with a host computer which controls the SpaceWire interface, often at the same time as controlling other EGSE interfaces. To provide representative timing of SpaceWire packets, the software has to operate in real time, which is both costly and difficult to develop. Last minute changes are very difficult to implement, especially when the software is controlling multiple interfaces.

What is needed is a unit that will allow arbitrary SpaceWire packets to be transmitted, in a predefined sequence, at a specified user data rate. It should initiate the sending of a packet sequence on command from the host software, when a particular SpaceWire packet is received, or when an external trigger is asserted. It should do this without the need for any real time software development.

The new STAR-Dundee SpaceWire-EGSE unit is just such a unit. It is provided with a special scripting language which allows SpaceWire packets to be defined using easy to understand terms. This language also specifies the time sequencing of packets and the event or series of events that cause various packet sequences to be sent. The information thus provided is compiled and loaded on to the SpaceWire-EGSE hardware. Thereafter the only interaction with user real-time software controlling the SpaceWire-EGSE and other equipment is through software events that can be asserted by the user application or indicated by the SpaceWire-EGSE.

2 HARDWARE

The SpaceWire EGSE is configured via a USB connection to the host PC. It has two SpaceWire ports from which packets can be generated and received. It has four external triggers, three input and one output. It also has a large memory for storing packet definitions.

3 SPACEWIRE EGSE SCRIPTING LANGUAGE

The SpaceWire EGSE is configured using a simple yet powerful scripting language. The language can be used to define variables, events, packets, packet generation schedules and state machines.

3.1 PACKET DEFINITION

Packet definitions can consist of data defined in hexadecimal or decimal bytes, variable references, CRC and checksum calculations and EEP and EOP control characters.

Example	Description

`packet myPkt`
`      hex(0A 0B 0C 0D)`
`      eop`
`end packet`
	Defines a packet named “myPkt” consisting of data specified in hexadecimal bytes (0A 0B 0C 0D) followed by an end of packet marker.

Above is a very basic packet definition. As with much of the SpaceWire EGSE language, packets are defined using a header, body and footer. The header consists of the “packet” keyword followed by the packet name and indicates the start of a packet definition. The packet body defines the packet contents. The packet footer consists of the keywords “end packet” and indicates the packet definition end.

Example	Description

`packet myPkt`
`      start(crc8)`
`      hex(0A 0B 0C 0D)`
`      dec(01 02 03 04) * 2`
`      stop(crc8)`
`      crc8`
`      eop`
`end packet`
	Defines a similar packet to the previous example but contains additional data specified in decimal. It also contains a CRC calculation and reference.

The start and stop statements in the example above can be used to calculate CRC and checksum values for the data between them. The CRC or checksum value can then be referenced in the packet definition.

3.2 Variables

The SpaceWire EGSE provides variables that are used to define packets with dynamic data. Variables have names by which they can be referenced in packet definitions along with a type and (optionally) an initial value. Each variable performs a function upon its value when read, based upon its type. The variable types available are increment (increments variable value by one when read), decrement (decrements variable value by one when read), rotate left (performs rotate left bit shift to variable value when read), rotate right (performs rotate right bit shift to variable when read) and random (assigns a random value to the variable). The example below demonstrates the use of a variable to dynamically set the ID of each packet sent.

Example	Description

`variables`
`      transactionID inc8 = 0`
`end variables`
 	
`packet myPkt`
`      hex(0A 0B 0C 0D)`
`      transactionID`
`      eop`
`end packet`
	Defines an increment variable named “transactionID” with an initial value of 0.
	
	A packet definition containing a reference to the incrementing variable “transactionID”.

3.3 Packet Generation Schedules

A schedule is used to define a timed sequence of packets for packet generation. The schedule references packets defined earlier in the script. Packets can be sent relative to the start of the schedule or relative to the previous packet. We can also specify the number of times the packet is sent.

Example	Description

`schedule mySchedule1`
`      5ms send myPkt1 * 2`
`      10ms send myPkt2`
`end schedule`
 	
`schedule mySchedule2`
`      5ms send myPkt1`
`      +10ms send myPkt2`
`end schedule`
	Schedule named “mySchedule1” sends packet “myPkt1” twice, 5ms after the schedule starts, and “myPkt2” once 10ms after the schedule starts.
	
	Schedule “mySchedule2” sends “myPkt1” 5ms after the schedule starts then “myPkt2” 10ms after “myPkt1” is sent.

3.4 STATE MACHINE

The state machine definition is responsible for control of the EGSE state. The state machine consists of state definitions. Each state has an associated schedule which is run when the state is entered at the data rate specified. Along with a schedule each state contains statements that determine when to change state and when to transition from one state to another.

```
statemachine 1
      initial state state1
            do mySchedule1 @ 20Mbps
            transition at end of schedule
            on myTrigIn1 goto state2
      end state
      state state2
            do mySchedule2 @ 50Mbps repeatedly
            transition at end of packet
            on myTimer goto state1
      end state
end statemachine
```

The above example is a state machine definition for SpaceWire link 1 of the SpaceWire EGSE. Two states are defined named “state1” and “state2”. On entering “state1” the schedule named “mySchedule1” is run once at a data rate of 20Mbps. If the event named “myTrigIn1” is received then at the end of the current schedule the state will change to “state2”. On entering “state2” the schedule named “mySchedule2” is run repeatedly at a data rate of 50Mbps. If the event named “myTimer” is received then once the current packet is sent the state will change to “state1”.

3.5 EVENTS

As seen in the state machine definition above, events are used to control the current state and therefore the current packet generation schedule. There are pre-defined events and user defined events. Predefined events include link started, link errors (parity, escape, credit, and disconnect), time-code received, and packet generation events. User defined events are timers, counters, software and external triggers.

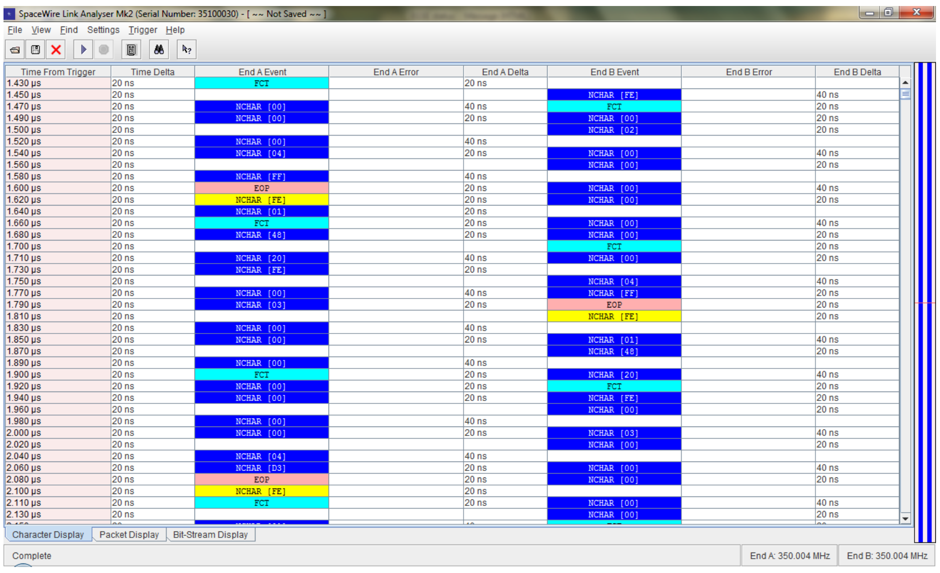
Example	Description

`timers`
`      myTimer 10ms start on mySWEvent1`
`end timers`
 	
`counters`
`      myCounter 10 on myTrigIn1`
`end counters`
 	
`software`
`      mySWEvent1 1`
`end software`
 	
`triggers`
`      myTrigIn1 input 1 rising`
`      output high on myTimer`
`end triggers`
	A 10ms timer that begins when the event "mySWEvent1" is received.
	
	A counter that generates an event when the event "myTrigIn1" is received 10 times.
	
	Declaration of a software event.
	
	Generates "myTrigIn1" event when rising signal received on external input trigger pin 1. Generates a high external output trigger signal when "myTimer" event is received.

Timers generate an event when a specified time is reached. The timer starts when the associated event is received. A counter has an initial value that is decremented each time an associated event is received. When the counter reaches zero it generates an event.

External trigger-in events specify the event to generate when a trigger-in signal is received on the associated input pin. The external output trigger generates a signal when it receives a specific event.

Software events permit host PC software to trigger a change in the state machine in the SpaceWire-EGSE. They provide a means of interaction with the host PC software. An API makes available functions the user can call upon to generate a software event. The host software can also be signalled when a specific event occurs or when a particular state is entered in the EGSE state machine.



SpaceWire Link Analyser Mk2 (Serial Number: 35100030) - [~~ Not Saved ~~]

File View Find Settings Trigger Help

Time From Trigger	Time Delta	End A Event	End A Error	End A Delta	End B Event	End B Error	End B Delta
1.430 µs	20 ns	FCT		20 ns			
1.450 µs	20 ns				NCHAR [FE]		40 ns
1.470 µs	20 ns	NCHAR [00]		40 ns	FCT		20 ns
1.490 µs	20 ns	NCHAR [00]		20 ns	NCHAR [00]		20 ns
1.500 µs	20 ns				NCHAR [02]		20 ns
1.520 µs	20 ns	NCHAR [00]		40 ns			
1.540 µs	20 ns	NCHAR [04]		20 ns	NCHAR [00]		40 ns
1.560 µs	20 ns				NCHAR [00]		20 ns
1.580 µs	20 ns	NCHAR [FF]		40 ns			
1.600 µs	20 ns	EOP		20 ns	NCHAR [00]		40 ns
1.620 µs	20 ns	NCHAR [FE]		20 ns	NCHAR [00]		20 ns
1.640 µs	20 ns	NCHAR [01]		20 ns			
1.660 µs	20 ns	FCT		20 ns	NCHAR [00]		40 ns
1.680 µs	20 ns	NCHAR [48]		20 ns	NCHAR [00]		20 ns
1.700 µs	20 ns				FCT		20 ns
1.710 µs	20 ns	NCHAR [20]		40 ns	NCHAR [00]		20 ns
1.730 µs	20 ns	NCHAR [FE]		20 ns			
1.750 µs	20 ns				NCHAR [04]		40 ns
1.770 µs	20 ns	NCHAR [00]		40 ns	NCHAR [FF]		20 ns
1.790 µs	20 ns	NCHAR [03]		20 ns	EOP		20 ns
1.810 µs	20 ns				NCHAR [FE]		20 ns
1.830 µs	20 ns	NCHAR [00]		40 ns			
1.850 µs	20 ns	NCHAR [00]		20 ns	NCHAR [01]		40 ns
1.870 µs	20 ns				NCHAR [48]		20 ns
1.890 µs	20 ns	NCHAR [00]		40 ns			
1.900 µs	20 ns	FCT		20 ns	NCHAR [20]		40 ns
1.920 µs	20 ns	NCHAR [00]		20 ns	FCT		20 ns
1.940 µs	20 ns	NCHAR [00]		20 ns	NCHAR [FE]		20 ns
1.960 µs	20 ns				NCHAR [00]		20 ns
1.980 µs	20 ns	NCHAR [00]		40 ns			
2.000 µs	20 ns	NCHAR [00]		20 ns	NCHAR [03]		40 ns
2.020 µs	20 ns				NCHAR [00]		20 ns
2.040 µs	20 ns	NCHAR [04]		40 ns			
2.060 µs	20 ns	NCHAR [D3]		20 ns	NCHAR [00]		40 ns
2.080 µs	20 ns	EOP		20 ns	NCHAR [00]		20 ns
2.100 µs	20 ns	NCHAR [FE]		20 ns			
2.110 µs	20 ns	FCT		20 ns	NCHAR [00]		40 ns
2.130 µs	20 ns				NCHAR [00]		20 ns

Character Display | Packet Display | Bit-Stream Display

Complete | End A: 350.004 MHz | End B: 350.004 MHz

The screenshot above is taken from a SpaceWire Link Analyser and shows the EGSE generating a sequence of small packets on both SpaceWire links, at the maximum rate possible on the link. Note that the link is running at 350MHz and no NULL characters are seen in the trace.

4 CONCLUSION

This paper has briefly described the SpaceWire EGSE and the scripting language used to configure it. The scripting language can quickly be used to configure the SpaceWire EGSE unit to mimic the behaviour of the SpaceWire device of interest. The SpaceWire EGSEs ability to generate packets independent of host software means it can produce very similar if not identical packet generation behaviour to the simulated instrument. The external input and output triggers on the SpaceWire EGSE provide a means by which to integrate with other test equipment. Such capabilities make the SpaceWire EGSE a quick and efficient way of simulating SpaceWire devices. Using the SpaceWire EGSE it is possible to develop a complete SpaceWire instrument or other device simulation with real-time behaviour, in little more than one day.

5 REFERENCES

1. STAR-Dundee, http://star-dundee.com/products.php, STAR-Dundee SpaceWire Products, STAR-Dundee Website.