STAR-Dundee

SpaceWire Engineering Excellence

SpaceWire EGSE: Simulating a Camera

The SpaceWire Electronic Ground Support Equipment (EGSE) is a test and development unit that simulates instruments or other SpaceWire equipment in real-time. The EGSE is configured using a simple yet powerful scripting language designed specifically for SpaceWire applications. Once configured the EGSE operates independent of software resulting in real-time performance. This can be used to rapidly mimic the behaviour of SpaceWire equipment, vastly reducing traditional development time, risk and cost associated with writing equivalent software in a real-time operating system.

This application note provides an example of how a camera may be simulated using a SpaceWire EGSE. Comparing this to traditional EGSE which requires complex and expensive real-time software development, the time saving, risk reduction and cost benefits provided by the SpaceWire EGSE should become clear.

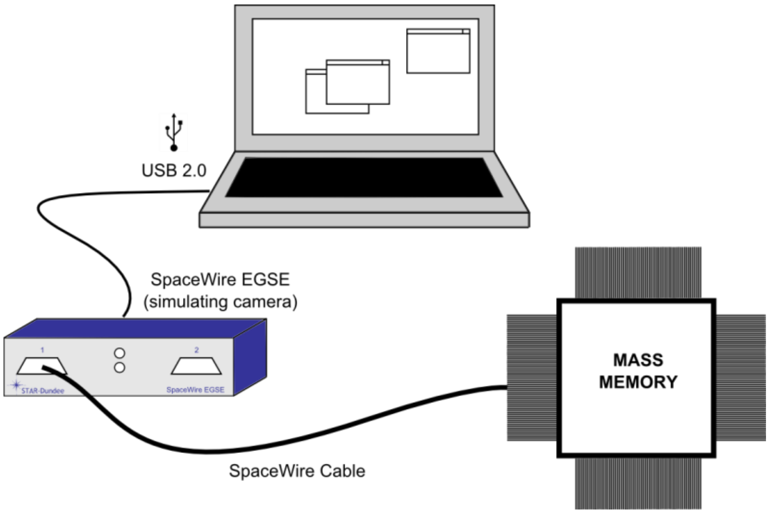
Scenario

“Company A” is responsible for developing mass memory that will ultimately be connected to a camera via SpaceWire. Simultaneously “Company B” is responsible for the camera development. Whilst the camera is in development it is unavailable to “Company A”, therefore to successfully create and test the mass memory “Company A” needs to accurately simulate the SpaceWire traffic that will be produced by the camera in real-time.

To accurately simulate the camera a series of eight packets, each containing the data of one image, must be transmitted with a 100ms interval between each packet with a link speed of 200Mbits/s at the maximum data rate possible.

Test Setup

The SpaceWire EGSE is connected to the host PC via USB and powered by a 5V power brick. A SpaceWire cable connects interface one of the EGSE to the mass memory. The diagram below illustrates this configuration.



Camera Simulation Test Setup

Application Note

Scripting the Camera Simulation

In order to configure the SpaceWire EGSE to simulate the camera, a script must first be written that defines the camera behavior. In this example the link speed is first stipulated:

```
config
       spw_tx_rate(1, 200Mbps)
end config
```

The above statement sets the line rate of SpaceWire link one to 200Mbits/s.

The packets containing the image data are then defined:

```
packet image_001
       file(“image_001.ppm”)
       eop
end packet

packet image_002
       file(“image_002.ppm”)
       eop
end packet

packet image_003
       file(“image_003.ppm”)
       eop
end packet

packet image_004
       file(“image_004.ppm”)
       eop
end packet

packet image_005
       file(“image_005.ppm”)
       eop
end packet

packet image_006
       file(“image_006.ppm”)
       eop
end packet

packet image_007
       file(“image_007.ppm”)
       eop
end packet

packet image_008
       file(“image_008.ppm”)
       eop
end packet
```

Eight packets are defined. Each packet consists of data imported from an image file followed by an EOP marker. The first packet defined is named “image_001” and consists of data imported from the image file “image_001.ppm” followed by an EOP marker. The next seven packet definitions follow a similar structure but each has a unique name and imports data from a unique image file.

An empty schedule and the schedule used to define the packet transmission timing are then defined:

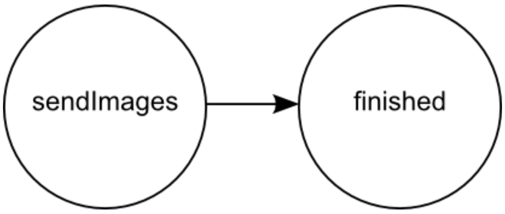
```
schedule nothing
end schedule

schedule sendImages
       100ms send image_001
       200ms send image_002
       300ms send image_003
       400ms send image_004
       500ms send image_005
       600ms send image_006
       700ms send image_007
       800ms send image_008
end schedule
```

The first schedule is named “nothing” and transmits nothing (the purpose of this schedule will become clear below). The second schedule is named “sendImages” and specifies that the packets named “image_001” through to “image_008” should be transmitted with a 100ms interval between the start of each packet.

Finally a state machine is defined:

```
statemachine 1
       initial state sendImages
              do sendImages
              LED colour is green
              goto finished
       end state
       state finished
              do nothing
              LED colour is red
       end state
end statemachine
```

A state machine is defined that is associated with SpaceWire interface one. It contains two states named “sendImages” and “finished”. The state named “sendImages” executes the schedule named “sendImages” then transitions to the state named “finished”. The state named “finished” executes the schedule named “nothing”.



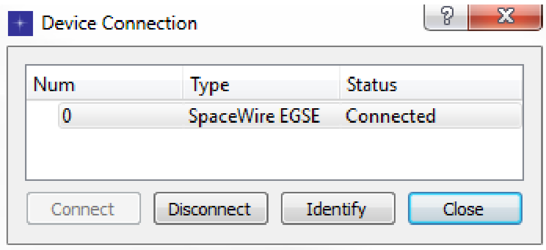
SpaceWire EGSE Camera Simulation State Diagram

When the SpaceWire EGSE is configured using this script, eight packets are transmitted from interface one with a 100ms interval between each. Each packet contains the data held within the referenced image file on disk.

The optional “LED colour is green” and “LED colour is red” statements in the state machine provide a simple indicator of the current executing state. Whilst the “sendImages” state is executed, the central LED above SpaceWire interface one is green and whilst the “finished” state is executed, the LED is red.

Compiling the Script

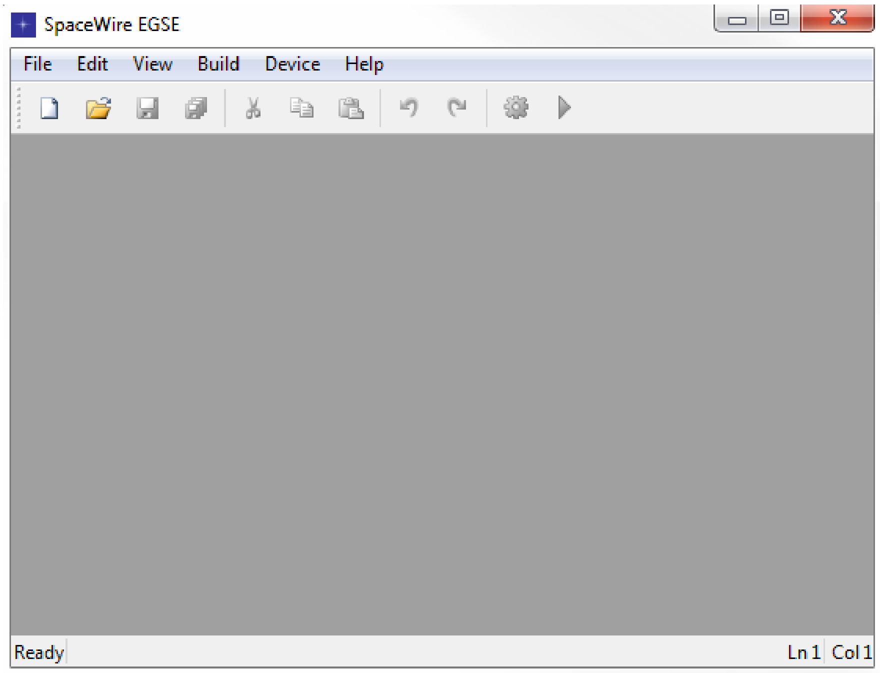
A script must be compiled before the SpaceWire EGSE can be configured. The SpaceWire EGSE comes with both a command line application and a GUI application that can be used to do this. In this example the GUI application will be used. Once the SpaceWire EGSE is connected and powered on, the "egse_gui" application is launched. A "Device Connection" window is presented where a connection to the SpaceWire EGSE is opened.



Num	Type	Status
0	SpaceWire EGSE	Connected

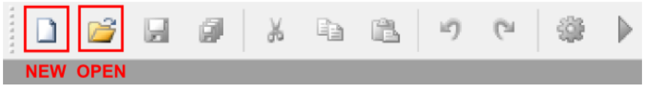
Device Connection Window

When the "Device Connection" window is closed the main window is displayed.



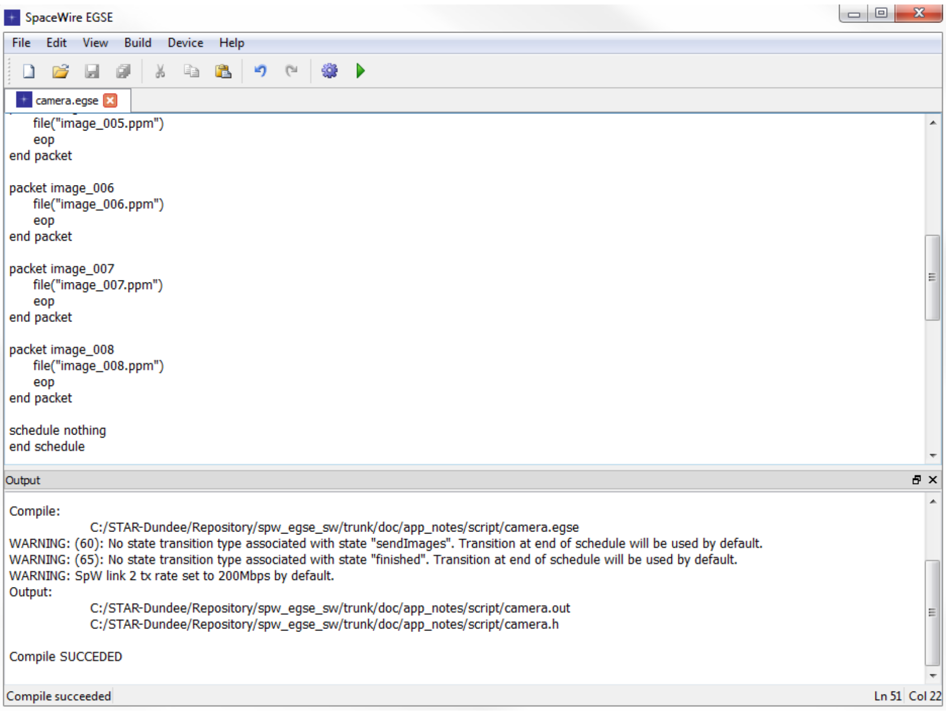
Main Window

To create the new camera script the "New" toolbar button is selected. Alternatively if the script was already created using a different text editor it can be opened using the "Open" toolbar button.



New and Open Toolbar Buttons

Once the camera simulation has been scripted, it is compiled using the "Compile" toolbar button. If the script has been newly created, a save window will prompt the user to save it. When the compile completes, an output window is displayed that shows any compiler errors or warnings along with the final compile status i.e. compile succeeded or failed.



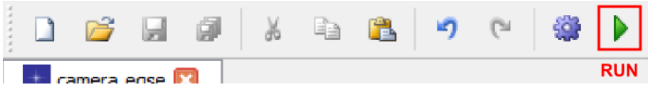
```
    file("image_005.ppm")
    eop
end packet

packet image_006
    file("image_006.ppm")
    eop
end packet

packet image_007
    file("image_007.ppm")
    eop
end packet

packet image_008
    file("image_008.ppm")
    eop
end packet

schedule nothing
end schedule
```

```
Compile:
            C:/STAR-Dundee/Repository/spw_egse_sw/trunk/doc/app_notes/script/camera.egse
WARNING: (60): No state transition type associated with state "sendImages". Transition at end of schedule will be used by default.
WARNING: (65): No state transition type associated with state "finished". Transition at end of schedule will be used by default.
WARNING: SpW link 2 tx rate set to 200Mbps by default.
Output:
            C:/STAR-Dundee/Repository/spw_egse_sw/trunk/doc/app_notes/script/camera.out
            C:/STAR-Dundee/Repository/spw_egse_sw/trunk/doc/app_notes/script/camera.h

Compile SUCCEDED
```

Compiler Output

Configure the SpaceWire EGSE

Once a script has been compiled successfully the SpaceWire EGSE can be configured. With a connection to the EGSE having previously been opened and the camera script open, the “Run” toolbar button is selected.

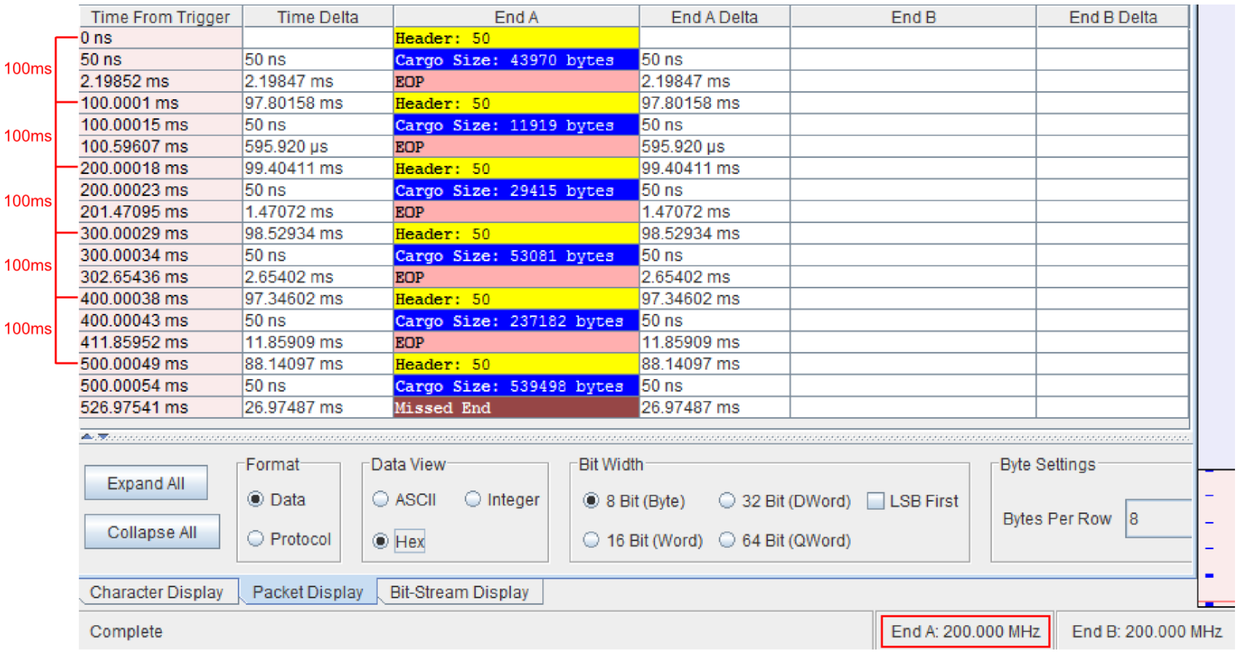


Run Toolbar Button

This configures the SpaceWire EGSE in such a way that it behaves as specified in the camera script. Once configured it operates independent of software resulting in real-time behavior.

Resulting SpaceWire Traffic

As soon as the SpaceWire EGSE is configured it operates as defined in the camera script: the link speed is set to 200Mbits/s and eight packets containing image data are transmitted from SpaceWire interface one at the maximum data rate with a 100ms interval between each. The screenshot below was taken using a Link Analyser Mk2 and shows the expected behavior.



Time From Trigger	Time Delta	End A	End A Delta	End B	End B Delta
0 ns		Header: 50			
50 ns	50 ns	Cargo Size: 43970 bytes	50 ns		
2.19852 ms	2.19847 ms	EOP	2.19847 ms		
100.0001 ms	97.80158 ms	Header: 50	97.80158 ms		
100.00015 ms	50 ns	Cargo Size: 11919 bytes	50 ns		
100.59607 ms	595.920 µs	EOP	595.920 µs		
200.00018 ms	99.40411 ms	Header: 50	99.40411 ms		
200.00023 ms	50 ns	Cargo Size: 29415 bytes	50 ns		
201.47095 ms	1.47072 ms	EOP	1.47072 ms		
300.00029 ms	98.52934 ms	Header: 50	98.52934 ms		
300.00034 ms	50 ns	Cargo Size: 53081 bytes	50 ns		
302.65436 ms	2.65402 ms	EOP	2.65402 ms		
400.00038 ms	97.34602 ms	Header: 50	97.34602 ms		
400.00043 ms	50 ns	Cargo Size: 237182 bytes	50 ns		
411.85952 ms	11.85909 ms	EOP	11.85909 ms		
500.00049 ms	88.14097 ms	Header: 50	88.14097 ms		
500.00054 ms	50 ns	Cargo Size: 539498 bytes	50 ns		
526.97541 ms	26.97487 ms	Missed End	26.97487 ms		

Expand All
Collapse All

Format: ● Data ○ Protocol

Data View: ○ ASCII ○ Integer ● Hex

Bit Width: ● 8 Bit (Byte) ○ 32 Bit (DWord) □ LSB First ○ 16 Bit (Word) ○ 64 Bit (QWord)

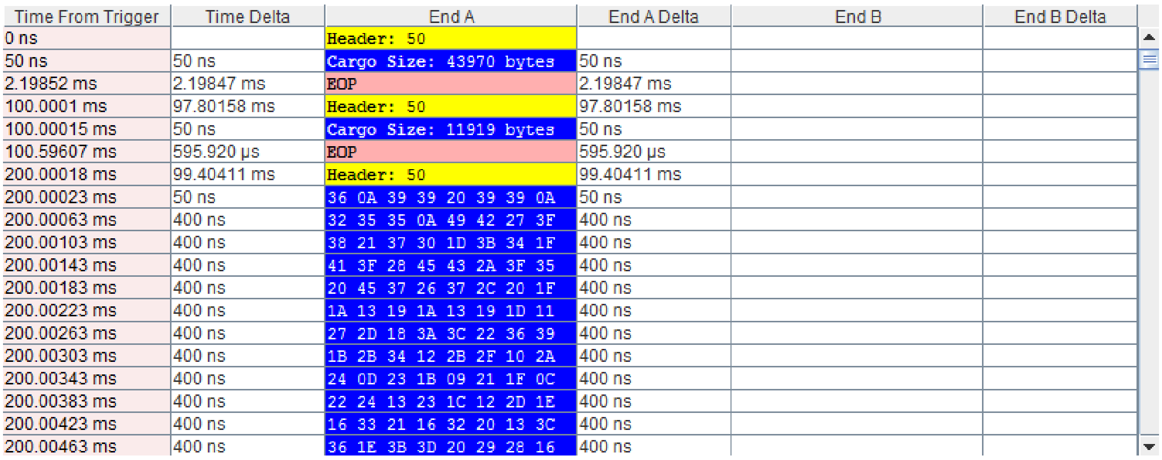
Byte Settings: Bytes Per Row 8

Character Display | Packet Display | Bit-Stream Display

Complete | End A: 200.000 MHz | End B: 200.000 MHz

LA Mk2 Screenshot Showing 100ms Packet Interval and 200Mbits/s Link Speed

The screenshot above shows the first five packets and a partial sixth packet transmitted from interface one of the EGSE (all eight packets could not be captured as they exceed the Link Analyser Mk2 memory size available to the software). The link speed shown in the bottom right corner is 200Mbits/s. Between each packet there is an interval of 100ms.



Time From Trigger	Time Delta	End A	End A Delta	End B	End B Delta
0 ns		Header: 50			
50 ns	50 ns	Cargo Size: 43970 bytes	50 ns		
2.19852 ms	2.19847 ms	EOP	2.19847 ms		
100.0001 ms	97.80158 ms	Header: 50	97.80158 ms		
100.00015 ms	50 ns	Cargo Size: 11919 bytes	50 ns		
100.59607 ms	595.920 µs	EOP	595.920 µs		
200.00018 ms	99.40411 ms	Header: 50	99.40411 ms		
200.00023 ms	50 ns	36 0A 39 39 20 39 39 0A	50 ns		
200.00063 ms	400 ns	32 35 35 0A 49 42 27 3F	400 ns		
200.00103 ms	400 ns	38 21 37 30 1D 3B 34 1F	400 ns		
200.00143 ms	400 ns	41 3F 28 45 43 2A 3F 35	400 ns		
200.00183 ms	400 ns	20 45 37 26 37 2C 20 1F	400 ns		
200.00223 ms	400 ns	1A 13 19 1A 13 19 1D 11	400 ns		
200.00263 ms	400 ns	27 2D 18 3A 3C 22 36 39	400 ns		
200.00303 ms	400 ns	1B 2B 34 12 2B 2F 10 2A	400 ns		
200.00343 ms	400 ns	24 0D 23 1B 09 21 1F 0C	400 ns		
200.00383 ms	400 ns	22 24 13 23 1C 12 2D 1E	400 ns		
200.00423 ms	400 ns	16 33 21 16 32 20 13 3C	400 ns		
200.00463 ms	400 ns	36 1E 3B 3D 20 29 28 16	400 ns		

LA Mk2 Screenshot Showing Packet with Image Data Cargo

The screenshot above partially shows the image data cargo held in the third of the packets transmitted.

Conclusion

This application note demonstrates how the SpaceWire EGSE and its associated scripting language could be used to very quickly simulate the SpaceWire traffic generated by a camera. It has introduced some of the key concepts of the EGSE scripting language (link speed configuration, packet definitions, scheduling and state machines), shown one way in which the EGSE can be operated (script creation, compilation and EGSE configuration via the GUI application) and shown the performance possible thanks to the EGSE's ability to operate independent of software.

This example is very simple and only touches on the range of features both the EGSE hardware and software provide. For more information please visit our website at www.star-dundee.com or contact us at enquiries@star-dundee.com.

STAR-Dundee

STAR-Dundee
STAR House, 166 Nethergate
Dundee, DD1 4EE, Scotland, UK
Tel: +44 1382 201 755 Fax: +44 1382 388 838
Email: enquiries@star-dundee.com
Web: www.star-dundee.com