SpaceWire EGSE: Real-time instrument simulation in a day

Stephen Mudie, Steve Parkes, Martin Dunstan

Introduction

- What is the SpaceWire EGSE?
- How does it work?
  - Hardware
  - Software
- Scripting Language
  - Example scripts
- Capabilities/Benefits
- Demo

STAR-Dundee

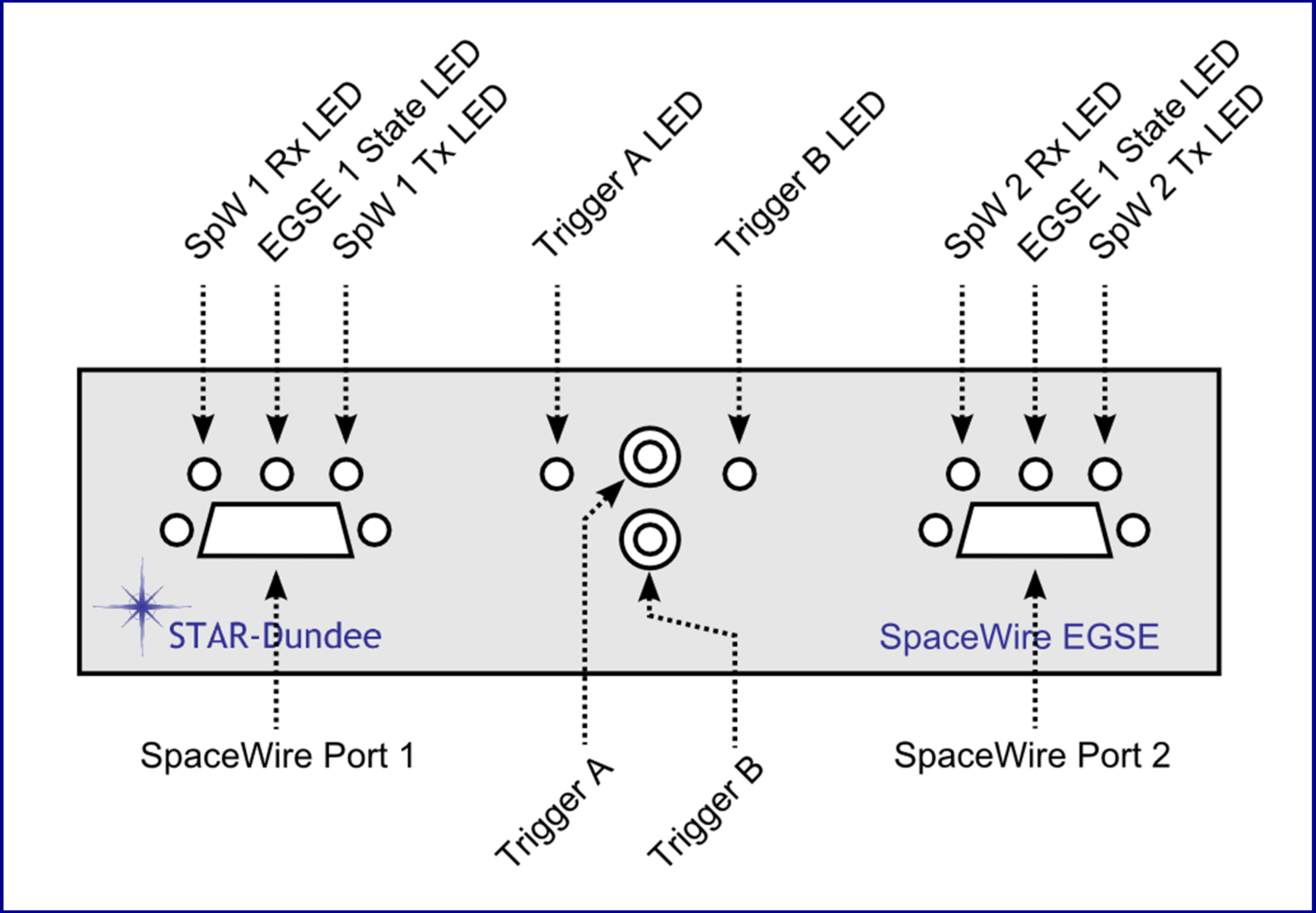
SpaceWire EGSE

- What it is: SpaceWire test and development unit developed by STAR-Dundee
- Purpose: Simulate instruments or other SpaceWire equipment in real-time during testing and integration
- Generates user defined packets in pre-defined sequences at specific times and data rates
  - i.e. packet 1 followed by packet 2 10ms later at 100Mbps

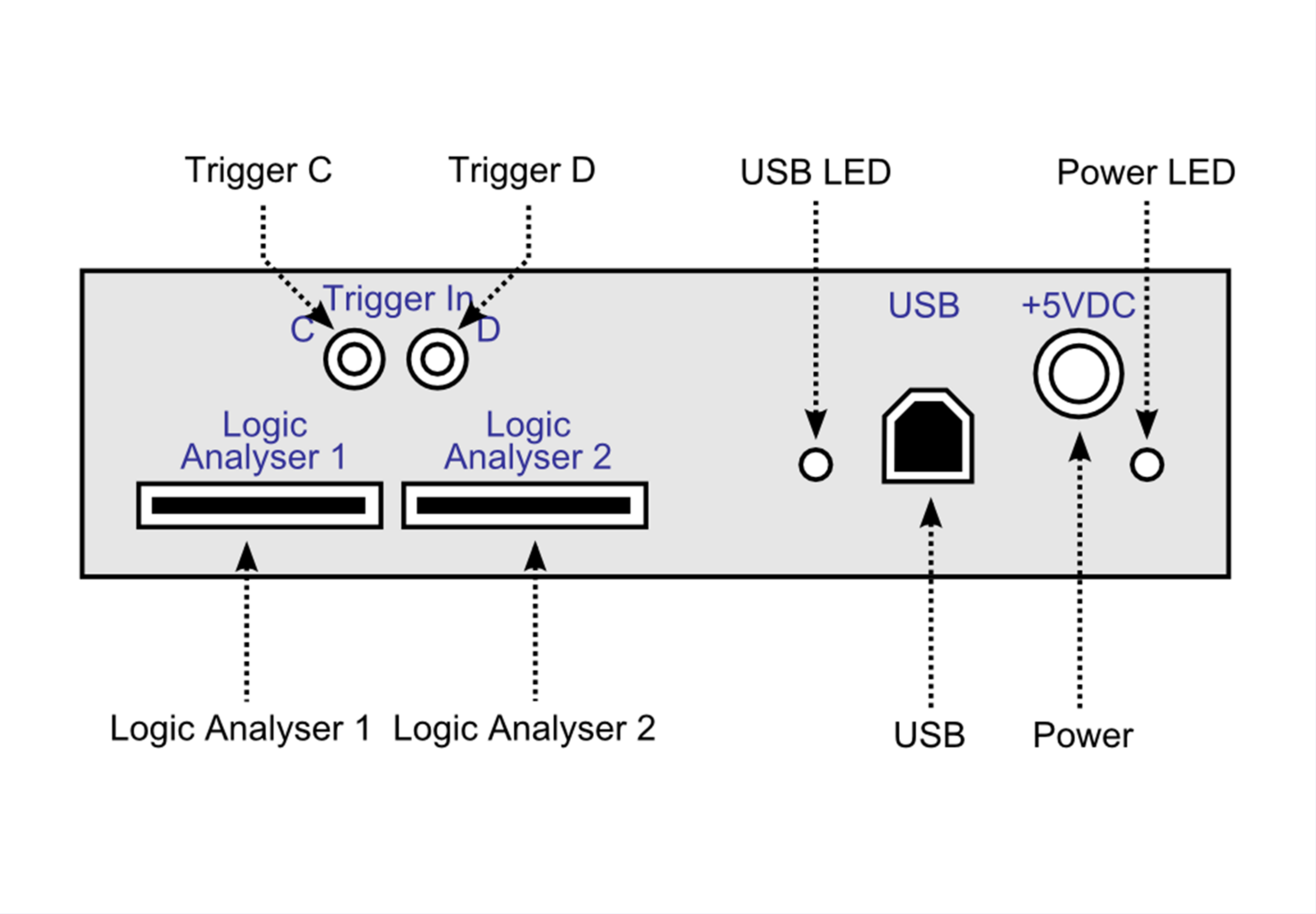
SpaceWire EGSE Hardware

- 2 SpaceWire interfaces
- 4 External Triggers (3 IN, 1 OUT)
- Indicator LEDs
- 128MB Memory
- USB connection to host PC

SpaceWire EGSE Front Panel



SpaceWire EGSE Rear Panel



SpaceWire EGSE Software

- Compiler
  - Compiles scripts into EGSE configuration files
- Loader
  - Loads EGSE configuration files onto hardware
- Software API
  - Create custom applications to control and interact with the EGSE
  - Can provide notifications of state changes and events

SpaceWire EGSE Demo Setup

- SpW cable connects links 1 and 2 of the EGSE
- SpW Link Analyser Mk2 displays traffic

SpaceWire EGSE – Simple Script

```
config
        spw_tx_rate(1, 200Mbps)
        spw_tx_rate(2, 200Mbps)
end config

packet pkt1
        dec(1 2 3 4)
        eop
end packet

schedule schedule1 @ 100Mbps
        500ms send pkt1
end schedule

statemachine 1
        initial state state1
                do schedule1 repeatedly
        end state
end statemachine
```

Set line rates to 200Mbit/s

SpaceWire EGSE – Simple Script

```
config
        spw_tx_rate(1, 200Mbps)
        spw_tx_rate(2, 200Mbps)
end config

packet pkt1
        dec(1 2 3 4)
        eop
end packet

schedule schedule1 @ 100Mbps
        500ms send pkt1
end schedule

statemachine 1
        initial state state1
                do schedule1 repeatedly
        end state
end statemachine
```

Define packet named “pkt1” with 4 decimal bytes followed by EOP

SpaceWire EGSE – Simple Script

```
config
       spw_tx_rate(1, 200Mbps)
       spw_tx_rate(2, 200Mbps)
end config

packet pkt1
       dec(1 2 3 4)
       eop
end packet

schedule schedule1 @ 100Mbps
       500ms send pkt1
end schedule

statemachine 1
       initial state state1
              do schedule1 repeatedly
       end state
end statemachine
```

Define schedule named “schedule1” that sends “pkt1” after 500ms at 100Mbit/s

SpaceWire EGSE – Simple Script

```
config
       spw_tx_rate(1, 200Mbps)
       spw_tx_rate(2, 200Mbps)
end config

packet pkt1
       dec(1 2 3 4)
       eop
end packet

schedule schedule1 @ 100Mbps
       500ms send pkt1
end schedule

statemachine 1
       initial state state1
              do schedule1 repeatedly
       end state
end statemachine
```

SpW link 1 state machine has a single state that executes “schedule1” repeatedly

SpaceWire EGSE – Simple Script

- Compile and run 001_simple.egse

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Declare random variable

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Declare constant variable value 2

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Declare incrementing variable value 0

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Declare CRC variable

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Define packet named “pkt1” with 4 hexadecimal bytes followed by EOP

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Define packet named “pkt2” that references random variable 32 times

SpaceWire EGSE – Schedules Script

```
variables
        random = rnd08()
        header = int08(2)
        id = inc08(0)
        crc = crc08()
end variables
packet pkt1
        hex(0A 0B 0C 0D)
        eop
end packet
packet pkt2
        random * 32
        eop
end packet
packet pkt3
        start(crc)
                header
                id
                random * 8
        stop(crc)
        crc
        eop
end packet
```

Define packet named “pkt3”:

- Constant header
- Incrementing ID
- 8 random bytes
- CRC calculation

SpaceWire EGSE – Schedules Script

```
schedule schedule1
        1s send pkt1 * 5
end schedule

schedule schedule2
         500ms send pkt1
        1000ms send pkt2
        1500ms send pkt3
end schedule

schedule schedule3
        0ms send pkt3
        +1s send pkt2
        +1s send pkt1
end schedule
```

“schedule1” sends “pkt1” 5 times, 1s after schedule starts

SpaceWire EGSE – Schedules Script

```
schedule schedule1
        1s send pkt1 * 5
end schedule

schedule schedule2
         500ms send pkt1
        1000ms send pkt2
        1500ms send pkt3
end schedule

schedule schedule3
        0ms send pkt3
        +1s send pkt2
        +1s send pkt1
end schedule
```

“schedule2” sends “pkt1” 500ms, “pkt2” 1000ms and “pkt3” 1500ms after schedule starts

SpaceWire EGSE – Schedules Script

```
schedule schedule1
        1s send pkt1 * 5
end schedule

schedule schedule2
         500ms send pkt1
        1000ms send pkt2
        1500ms send pkt3
end schedule

schedule schedule3
        0ms send pkt3
        +1s send pkt2
        +1s send pkt1
end schedule
```

“schedule3” sends “pkt3” immediately, “pkt2” 1s after “pkt3” tx begins and “pkt1” 1s after “pkt2” tx begins

SpaceWire EGSE – Schedules Script

```
statemachine 1
        initial state state1
                LED colour is blue
                do schedule1
                goto state2
        end state

        state state2
                LED colour is green
                do schedule2
                goto state3
        end state

        state state3
                LED colour is red
                do schedule3
                goto state1
        end state
end statemachine
```

“state1” executes “schedule1”once then transitions to “state2”

“state2” executes “schedule2”once then transitions to “state3”

“state3” executes “schedule3”once then transitions to “state1”

SpaceWire EGSE – Schedules Script

```
statemachine 1
       initial state state1
              LED colour is blue
              do schedule1
              goto state2
       end state

       state state2
              LED colour is green
              do schedule2
              goto state3
       end state

       state state3
              LED colour is red
              do schedule3
              goto state1
       end state
end statemachine
```

blue state

green state

red state

SpaceWire EGSE – Schedules Script

- Compile and run 002_schedules.egse

SpaceWire EGSE – FSM and Events Script

```
events
        swEvent0 = software_in(0)
end events

statemachine 1
        initial state state1
                LED colour is blue
                do schedule1
                on swEvent0 goto state2
        end state

        state state2
                LED colour is green
                do schedule2
                on swEvent0 goto state3
        end state

        state state3
                LED colour is red
                do schedule3
                on swEvent0 goto state1
        end state
end statemachine
```

Declare software event named “swEvent0”

SpaceWire EGSE – FSM and Events Script

```
events
        swEvent0 = software_in(0)
end events

statemachine 1
        initial state state1
                LED colour is blue
                do schedule1
                on swEvent0 goto state2
        end state

        state state2
                LED colour is green
                do schedule2
                on swEvent0 goto state3
        end state

        state state3
                LED colour is red
                do schedule3
                on swEvent0 goto state1
        end state
end statemachine
```

Change state when “swEvent0” is detected.

STAR-Dundee

SpaceWire EGSE – FSM and Events Script

- Compile and load 003_events_sw.egse

SpaceWire EGSE – Camera Example

- Requirements:
  - Simulate a camera that sends 8 images with a 100ms gap between each.

SpaceWire EGSE – Camera Script

```
packet image_001
       file("image_001.ppm")
       eop
end packet
…
packet image_008
       file("image_008.ppm")
       eop
end packet

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
```

Declare 8 packets that import data from image files

SpaceWire EGSE – Camera Script

```
packet image_001
        file("image_001.ppm")
        eop
end packet
…
packet image_008
        file("image_008.ppm")
        eop
end packet

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
```

Schedule to send 8 image packets with 100ms gaps.

SpaceWire EGSE – Camera Script

```
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
```

Send 8 images then move to a state where nothing is done

SpaceWire EGSE – Camera Script

- Compile and load 004_camera.egse

EGSE Capabilities

- Detailed packet definitions
  - Via raw data, variables, automatic CRC and checksum calculation
- Precise packet generation scheduling at specific data rates.
- Packet generation control
  - Via state machines and events

STAR-Dundee

EGSE Key Benefits

- Mimic real-time behaviour of SpaceWire units
- Integrate with equipment via external triggers
- Minimal development time

SpaceWire EGSE Conclusion

- Hardware
- Software
- Scripting Language
- Capabilities and Benefits
- Available now