

# Definition, Implementation and Testing of an XML-based Packet Description Language for Traffic Analysis in a SpaceWire Communication

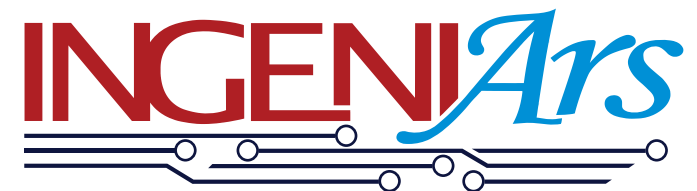
Roberto Ciardi\*<sup>°</sup>, Simone Vagaggini\*<sup>°</sup>, Antonino Marino<sup>°</sup>, Daniele Davalle<sup>°</sup>, Gionata Benelli<sup>°</sup>, Luca Fanucci\*

*\*University of Pisa*

*°IngeniArs S.r.l.*



UNIVERSITÀ DI PISA



*The Art of Engineering*

# Roberto Ciardi

- PhD Student @ University of Pisa
- Department of Information Engineering
- Industrial PhD
- Embedded Software Engineer @ IngeniArs S.r.l.



# Outline

- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion



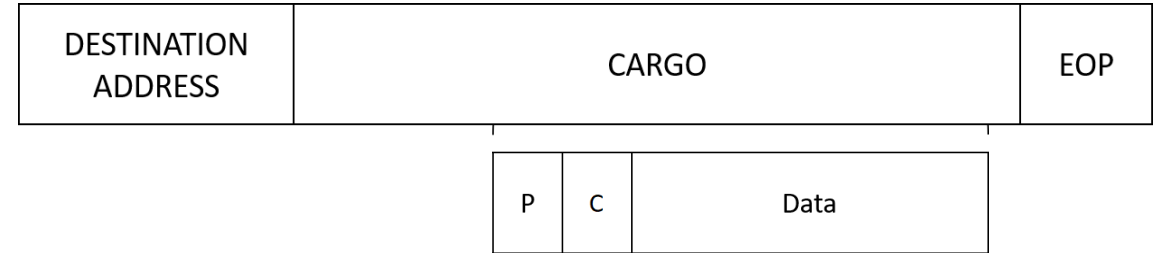
# Outline

- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion



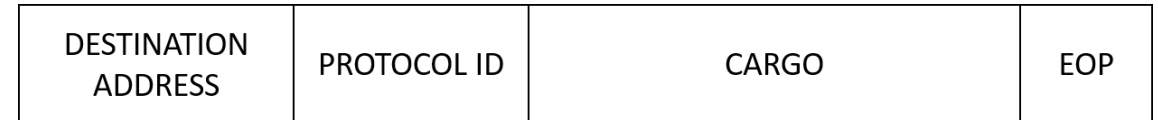
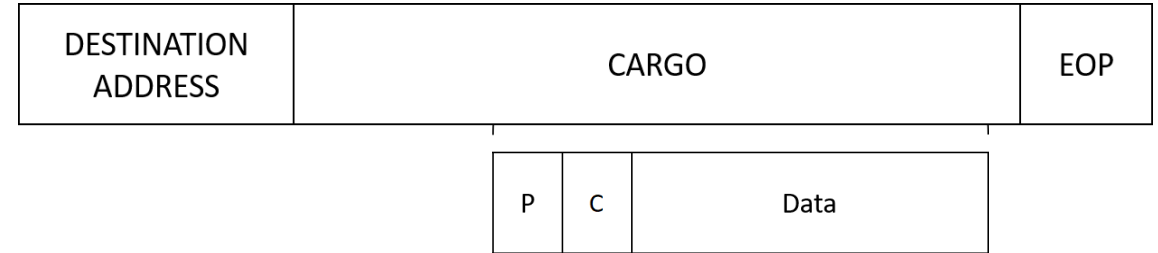
# SpaceWire packet format

- SpW Packet Format
  - Destination Address
    - Logical Address
    - Path Address
  - Cargo
  - EOP



# SpaceWire packet format

- SpW Packet Format
  - Destination Address
    - Logical Address
    - Path Address
  - Cargo
  - EOP
- Protocol ID
- Remote Memory Access Protocol (RMAP)
  - Write to/read from memory on a SpW Node



# SpW RMAP Packets

- Write Command

- 16 bytes header
  - Instruction byte indicates RMAP packet type
  - Memory address to write to
  - Length of data to be written
- Data to be written
- Verified

DESTINATION ADDRESS	PROTOCOL ID	INSTRUCTION BYTE	DESTINATION KEY
SOURCE ADDRESS	TRANSACTION ID (MS)	TRANSACTION ID (LS)	EXT. WRITE ADDRESS
WRITE ADDRESS (MS)	WRITE ADDR	WRITE ADDR	WRITE ADDRESS (LS)
DATA LENGTH (MS)	DATA LENGTH	DATA LENGTH (LS)	HEADER CRC
DATA	DATA	DATA	DATA
DATA	DATA	DATA CRC	EOP

# Outline

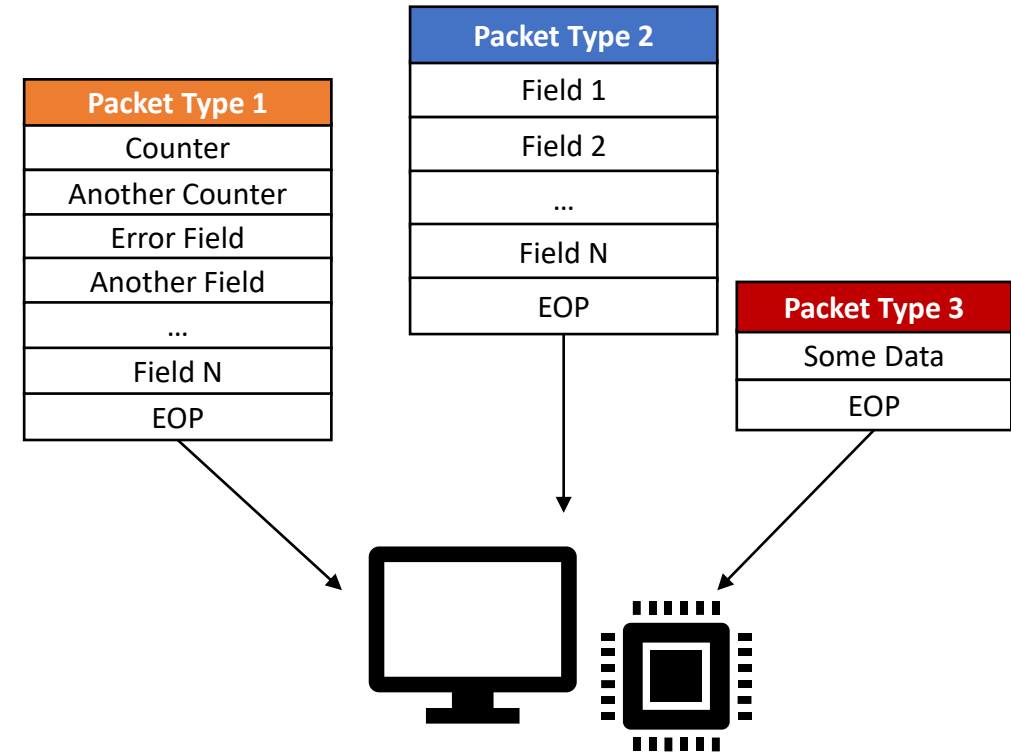
- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion





# SpW Packet Description Language

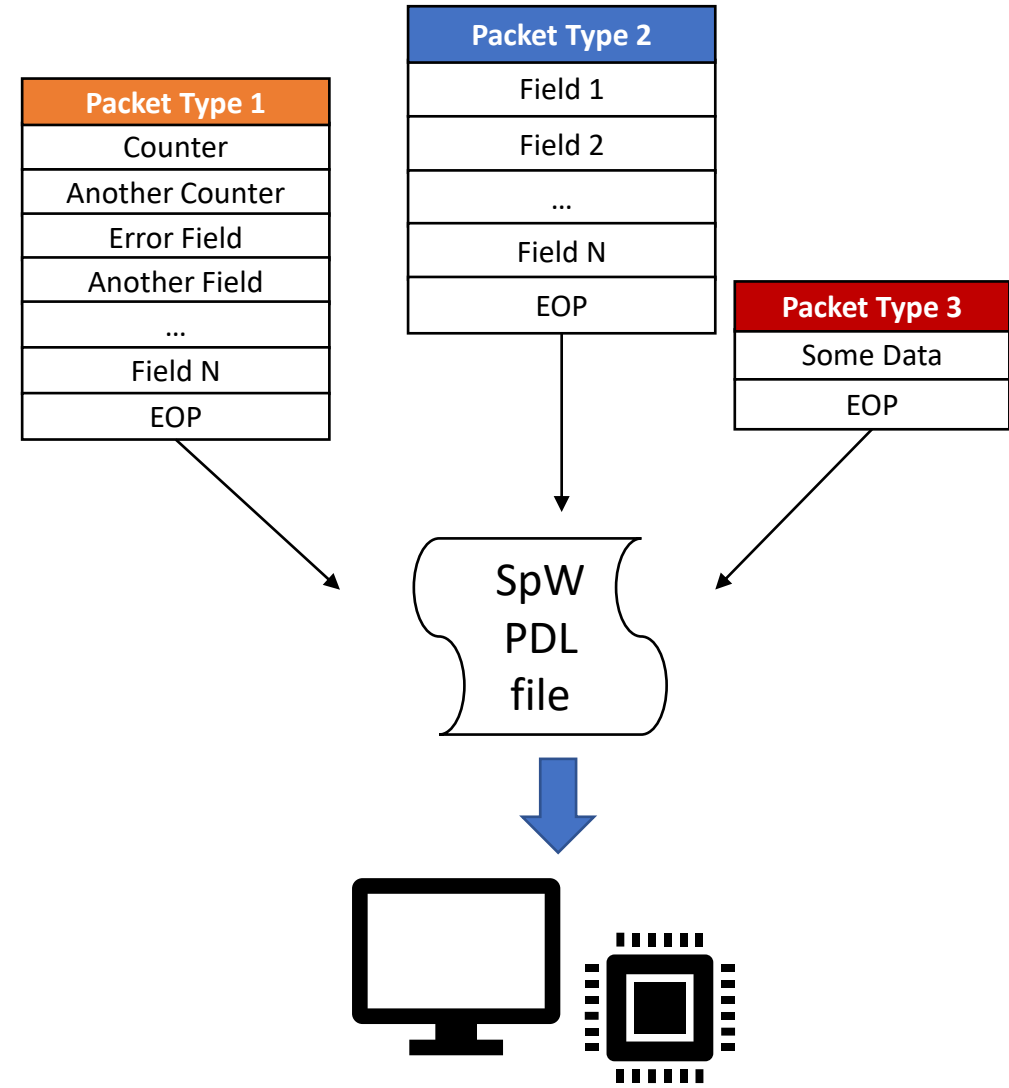
- SpW Packet are well-structured and flexible
- SpW-based missions use unique packet formats
- Precisely tailored for each mission
- Need for the development of special-purpose SpW packet manager for processing/generating SpW data
- Easy for machines, tedious for humans especially with several large packets



SpW Packet Management  
directly on SW/HW

# SpW Packet Description Language

- SpW Packet Description Language (SpW PDL)
- XML-Based description for enhancing automatic generation/parsing of SpW Traffic
- eXtensible Markup Language
  - Both machines- and human-readable
  - Supported by most programming languages (JAVA, C/C++, Python)
- Description SpW Packet structure
- Simplify human interaction
- Easy and flexible packet description



# SpW PDL – Format

- XML-based
- Address
  - Logical
  - Path
- Payload
  - Fields
- Can be used by multiple applications
- XML parser supported by most languages

```
<spw_pdl>
  <spw_packet name="generic_spw_packet">
    >> <address>
    >> >> <logical_address>
    >> >> >> 238
    >> >> </logical_address>
    >> >> </address>
    >> >> <payload>
    >> >> >> <field size="32" name="packet_counter" />
    >> >> >> <field size="8" name="1_byte_field" numOccurs="4" />
    >> >> >> <field size="16" name="2_bytes_field" numOccurs="2" />
    >> >> >> <field size="32" name="4_bytes_field" numOccurs="2" />
    >> >> >> <field size="8" numOccurs="100" name="dataarray" />
    >> >> >> <spareByte numOccurs="20"/>
    >> >> >> <EOP/>
    >> >> </payload>
    >> </spw_packet>
  </spw_pdl>
```

# SpW PDL – Format

```
<spw_pdl>
  <spw_packet name="generic_spw_packet">
    >> <address>
    >> >> <logical_address>
    >> >> >> 238
    >> >> >> </logical_address>
    >> >> </address>
    >> >> <payload>
    >> >> >> <field size="32" name="packet_counter"> 25 </field>
    >> >> >> <field size="8" name="1_byte_field" numOccurs="4">
    >> >> >> 255
    >> >> >> </field>
    >> >> >> <field size="16" name="2_bytes_field" numOccurs="2">
    >> >> >> 256
    >> >> >> </field>
    >> >> >> <field size="32" name="4_bytes_field" numOccurs="2">
    >> >> >> 65536
    >> >> >> </field>
    >> >> >> <EOP/>
    >> >> </payload>
    >> </spw_packet>
  </spw_pdl>
```



Generic SpW Packet	
	0xEE
	0x00
	0x00
	0x00
	0x19
	0xFF
	...
	0x01
	0x00
	...
	0x01
	0x00
	0x00
	...
	EOP

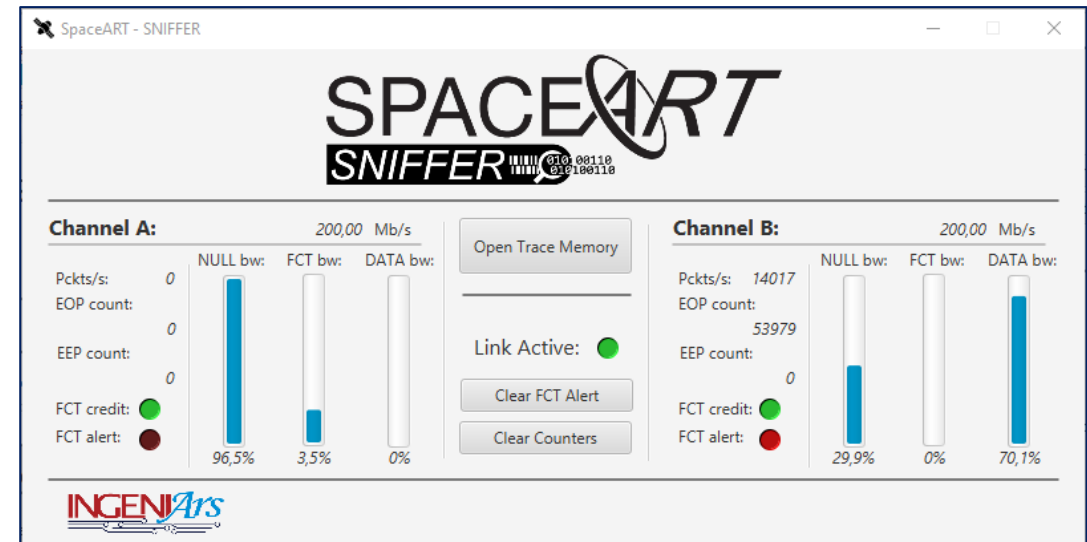
# Outline

- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion



# SpW SpaceART Sniffer

- SpW Link Analyser
- 2 SpW Interfaces
- Ethernet connection to Host-PC
- Graphical User Interface (GUI)
- SpW packets analysis
- SpW character level analysis
- Real-Time traffic analysis



# SpW SpaceART Sniffer – Data Analysis

- Database file
- Packet visualization
- Trigger
- Time visualization
- Time synchronization

Analysar Viewer

Configuration Content XML parser

Search: Filter Insert value SEARCH

☒ Align Tables

Go to Trigger

Time Unit: ☒ ms ☐ us ☐ ns

Show: ☐ FCT ☐ NULL ☐ ESC errors ☐ TIMECODE ☒ SYNC

Time Diff. [ms] First Time Lock Time Second Time Lock Time Compute Diff Difference

SpaceArt_EGSE			SpaceWire_DUT		
Pckt	Data	Time [ms]	Pckt	Data	
-	-	100	-	EOP	
8	HDR: 0xD0	101	-	-	
-	SIZE: 76	-	-	-	
-	EOP	101	-	-	
-	SYNC	110	-	SYNC	
-	-	110	9	HDR: 0xEE	
-	-	-	-	SIZE: 141	
-	-	110	-	EOP	
9	HDR: 0xD0	111	-	-	
-	SIZE: 76	-	-	-	
-	EOP	111	-	-	
-	SYNC	120	-	SYNC	
-	-	120	10	HDR: 0xEE	
-	-	-	-	SIZE: 141	
-	-	120	-	EOP	
10	HDR: 0xD0	121	-	-	
-	SIZE: 76	-	-	-	
-	EOP	121	-	-	
-	SYNC	130	-	SYNC	

<< < > >>

Previous Packet Next Packet Previous Packet Next Packet

DB loaded: database\_2020-10-28-11-34-48.db LOAD DB EXPORT DB

# SpW SpaceART Sniffer – Data Analysis

- Packet content
- Character level
- Time between characters
- Presence of special characters
- Single data value

Packet 2 content

Packet ID: 2

HEADER: 238

SIZE: 141

TERM: EOP

SpaceWire\_DUT

Time Scale: ☐ ms ☐ us ☒ ns

Show Data: ☒ HEX ☐ DEC ☐ BIN

SPEC CHAR: ☒ FCT ☐ NULL ☐ SYNC ☐ TIMECODE

Id	Time	Data
1	40714160	0xEE
2	40714240	0x4C
3	40714320	0x04
4	40714440	0x00
5	40714560	0x00
6	40714640	0x0A
7	40714720	0x09
8	40714840	0x00
9	40714960	0x00
10	40715040	0xDC
11	40715120	0x05
12	40715240	0x00
13	40715320	0x00
14	40715440	0x00



# SpW PDL – SpaceART Sniffer Integration

```
<spw pdl>
> <spw packet name="Packet_Type1">
>   <address>
>     <logical address>206</logical address>
>     <logical address>207</logical address>
>     <logical address>208</logical address>
>   </address>
>   <payload>
>     <field size="16">FIRST_16BIT_FIELD</field>
>     <field maxValue="2" minValue="0" size="16">16BIT_FIELD_MAX2</field>
>     <field maxValue="12000000" minValue="400" size="32">32BIT_FIELD</field>
>     <spareByte/>
>     <field size="8" numOccurs="28">ARRAY_OF_DATA</field>
>     <spareByte numOccurs="37"/>
>     <field size="8">CHECKSUM</field>
>     <EOP/>
>   </payload>
> </spw packet>
>
> <spw packet name="Packet_Type2">
>   <address>
>     <logical address>222</logical address>
>     <logical address>238</logical address>
>     <logical address>254</logical address>
>   </address>
>   <payload size="141">
>     <EOP/>
>   </payload>
> </spw packet>
>
> <spw packet name="Packet_Type3" size="16">
>   <address>
>     <logical address> 100</logical address>
>   </address>
>   <payload>
>     <spareByte />
>     <EOP />
>   </payload>
> </spw packet>
>
</spw pdl>
```

# SpW PDL – SpaceART Sniffer Integration

```
<spw pdl>
> <spw packet name="Packet_Type1">
>   <address>
>     <logical address>206</logical address>
>     <logical address>207</logical address>
>     <logical address>208</logical address>
>   </address>
>   <payload>
>     <field size="16">FIRST_16BIT_FIELD</field>
>     <field maxValue="2" minValue="0" size="16">16BIT_FIELD_MAX2</field>
>     <field maxValue="12000000" minValue="400" size="32">32BIT_FIELD</field>
>     <spareByte/>
>     <field size="8" numOccurs="28">ARRAY_OF_DATA</field>
>     <spareByte numOccurs="37"/>
>     <field size="8">CHECKSUM</field>
>     <EOP/>
>   </payload>
> </spw packet>
>
> <spw packet name="Packet_Type2">
>   <address>
>     <logical address>222</logical address>
>     <logical address>238</logical address>
>     <logical address>254</logical address>
>   </address>
>   <payload size="141">
>     <EOP/>
>   </payload>
> </spw packet>
>
> <spw packet name="Packet_Type3" size="16">
>   <address>
>     <logical address> 100</logical address>
>   </address>
>   <payload>
>     <spareByte />
>     <EOP />
>   </payload>
> </spw packet>
>
</spw pdl>
```

Analyser Viewer

Configuration Content XML parser

☐ Align Tables

SpaceArt_EGSE			SpaceWire_DUT		
Pckt	Time [ns]	Data	Pckt	Time [ns]	Data
1	31699880	Packet_Type1	1	30721720	Packet_Type2
-	-	SIZE: 76	-	-	SIZE: 141
-	31707400	EOP	-	30735760	EOP
2	41696520	Packet_Type1	2	40714160	Packet_Type2
-	-	SIZE: 76	-	-	SIZE: 141
-	41704040	EOP	-	40728200	EOP
3	51696520	Packet_Type1	3	50712280	Packet_Type2
-	-	SIZE: 76	-	-	SIZE: 141
-	51704080	EOP	-	50726320	EOP
4	61695960	Packet_Type1	4	60712240	Packet_Type2
-	-	SIZE: 76	-	-	SIZE: 141
-	61703520	EOP	-	60726280	EOP
5	71699040	Packet_Type1	5	70712280	Packet_Type2
-	-	SIZE: 76	-	-	SIZE: 141
-	71706600	EOP	-	70726320	EOP
6	81696120	Packet_Type1	6	80711960	Packet_Type2

SpaceArt\_EGSE - XML parsing info:

10/10 packets of type Packet\_Type1  
0/10 packets of type Packet\_Type2  
0/10 packets of type Packet\_Type3

SpaceWire\_DUT - XML parsing info:

0/56 packets of type Packet\_Type1  
56/56 packets of type Packet\_Type2  
0/56 packets of type Packet\_Type3

XML file loaded: xml\_file\_example.xml

DB loaded: database\_2020-10-28-11-34-48.db

☒ Export Binary

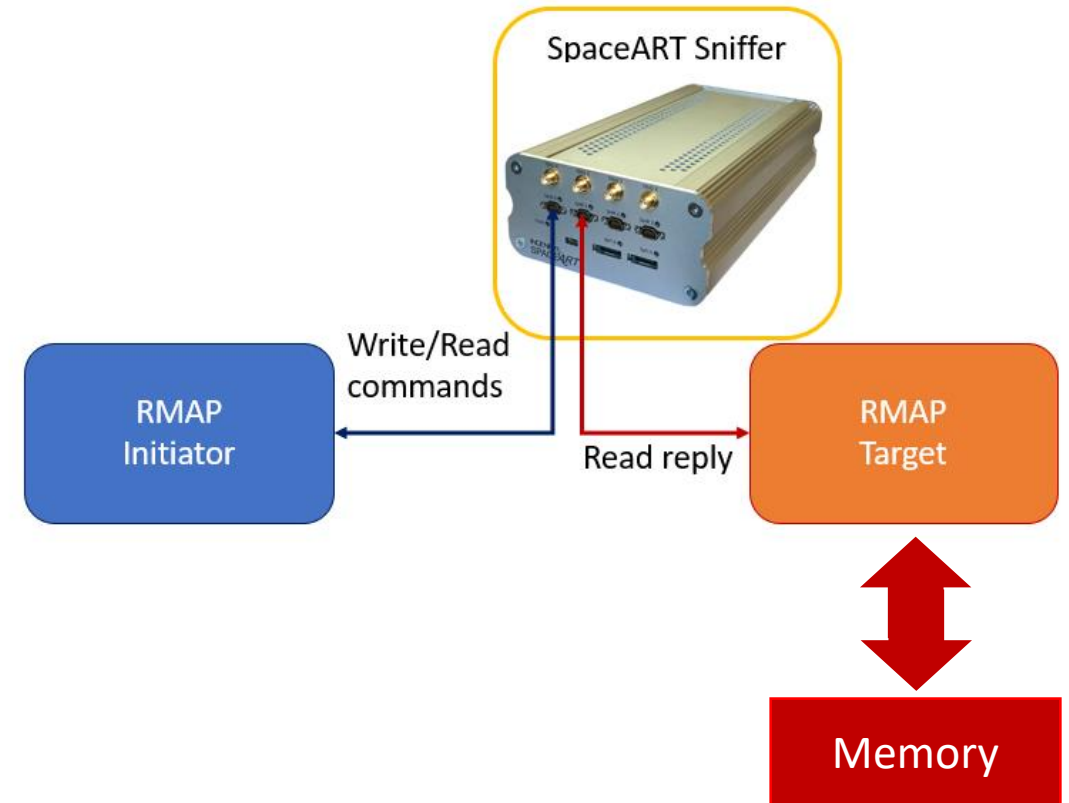
# Outline

- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion



# Test Case

- RMAP Communication
- RMAP Initiator sends a set of Write requests, each followed by a Read requests
- No replies for Write requests
- Dummy packets inbetween RMAP packets
- Data traffic analysed through SpaceART Sniffer (SpW PDL)



# Test Results

- Sniffer nominal data parser
- Write and Read packet can be recognized
- Read replies can be recognized on the other side

The screenshot shows the 'Analyser Viewer' application window. It has tabs for 'Configuration', 'Content', and 'XML parser'. Below the tabs is a search bar with a 'Filter' dropdown, an 'Insert value' text box, and a 'SEARCH' button. On the left side, there are checkboxes for 'Align Tables' and 'Go to Trigger'. Below these are radio buttons for 'Time Unit' (ms, us, ns) and a 'Show:' section with checkboxes for 'FCT', 'NULL', 'ESC errors', 'TIMECODE', and 'SYNC'. Further down are buttons for 'Time Diff. [ns]' (First Time, Lock Time, Second Time, Lock Time, Compute Diff, Difference). The main area displays two tables: 'RMAP\_Initiator' and 'RMAP\_Target'. The 'RMAP\_Initiator' table has columns 'Pckt', 'Data', and 'Time [ns]'. It shows packets 5, 6, 7, and 8, each with a yellow header row (HDR: 0xC8) and two blue data rows (SIZE: 26, EOP). The 'RMAP\_Target' table has columns 'Pckt' and 'Data'. It shows packets 6, 7, and 8, each with a yellow header row (HDR: 0xC9) and two blue data rows (SIZE: 22, EOP). At the bottom, there are navigation buttons (Prev, Packet, Next) and a status bar showing 'DB loaded: database\_2020-10-28-11-34-48.db' and buttons for 'LOAD DB', 'EXPORT DB', and 'Export Binary'.

RMAP_Initiator			RMAP_Target	
Pckt	Data	Time [ns]	Pckt	Data
-	-	70726320	-	EOP
5	HDR: 0xC8	71699040	-	-
-	SIZE: 26	-	-	-
-	EOP	71706600	-	-
6	HDR: 0xC8	72706000	-	-
-	SIZE: 17	-	-	-
-	EOP	72710500	-	-
-	-	75301010	6	HDR: 0xC9
-	-	-	-	SIZE: 22
-	-	75415000	-	EOP
7	HDR: 0xC8	91696520	-	-
-	SIZE: 26	-	-	-
-	EOP	91704040	-	-
8	HDR: 0xC8	92706000	-	-
-	SIZE: 17	-	-	-
-	EOP	92710500	-	-
-	-	100711640	7	HDR: 0xC9
-	-	-	-	SIZE: 22
-	-	100725680	-	EOP

# Test Results - SpW PDL for RMAP packets

```
<spw_pdl>
  <spw_packet name="RMAP_read_command">
>    <address>
>      <logical_address> 200 </logical_address>
>    </address>
    <protocol_id> 01 </protocol_id>
>    <payload>
      <field size="8" name="config_byte"> 76 </field>
      <field size="8" name="destination_key"> 00 </field>
      <field size="8" name="src_logical_addr"> 201 </field>
      <field size="16" name="transaction_id"> 01 </field>
      <field size="8" name="extended_addr"> 00 </field>
      <field size="32" name="read_address"> 00 </field>
      <field size="24" name="data_length"> 08 </field>
      <field size="8" name="header_crc"> 205 </field>
      <EOP/>
>    </payload>
  </spw_packet>
</spw_pdl>
```



# Test Results – RMAP packets

- Identifies the RMAP packets
- Shows their occurrence in the traffic
- Shows statistics about the packets
- The data inspection is simplified

The screenshot displays the 'Analyser Viewer' application window. It features two main panels: 'RMAP\_Initiator' on the left and 'RMAP\_Target' on the right. Each panel contains a table with columns for 'Pckt', 'Time [ns]', and 'Data'. Below the tables are navigation buttons (left arrow, right arrow, double left arrow, double right arrow) and a text box for 'XML parsing info:'. At the bottom, there are buttons for 'LOAD XML', 'LOAD DB', 'EXPORT DB', and a checkbox for 'Export Binary'.

**RMAP\_Initiator**

Pckt	Time [ns]	Data
5	71699040	RMAP_write_command
-	-	SIZE: 26
-	71706600	EOP
6	72706000	RMAP_read_command
-	-	SIZE: 17
-	72710500	EOP
7	91696520	RMAP_write_command
-	-	SIZE: 26
-	91704040	EOP
8	92706000	RMAP_read_command
-	-	SIZE: 17
-	92710500	EOP
9	111696520	RMAP_write_command
-	-	SIZE: 26
-	111704080	EOP
10	112706040	RMAP_read_command

**RMAP\_Target**

Pckt	Time [ns]	Data
6	75301010	RMAP_read_reply
-	-	SIZE: 22
-	75415000	EOP
7	100711640	RMAP_read_reply
-	-	SIZE: 22
-	100825630	EOP
8	126122270	RMAP_read_reply
-	-	SIZE: 22
-	126236260	EOP
9	151532900	RMAP_read_reply
-	-	SIZE: 22
-	151646800	EOP
10	176943530	RMAP_read_reply
-	-	SIZE: 22
-	177057430	EOP
11	202354160	RMAP_read_reply

**RMAP\_Initiator - XML parsing info:**

518/1482 packets of type RMAP\_read\_command  
518/1482 packets of type RMAP\_write\_command  
0/1482 packets of type RMAP\_read\_reply

**RMAP\_Target - XML parsing info:**

0/865 packets of type RMAP\_read\_command  
0/865 packets of type RMAP\_write\_command  
518/865 packets of type RMAP\_read\_reply

XML file loaded: rmap\_packets.xml

DB loaded: database\_2020-10-28-11-34-48.db

# Test Results – RMAP packets occurrences

- Identifies RMAP packets
- Shows statistics about the packets
- The data inspection is simplified

## RMAP\_Initiator - XML parsing info:

518/1482 packets of type RMAP\_read\_command  
518/1482 packets of type RMAP\_write\_command  
0/1482 packets of type RMAP\_read\_reply

## RMAP\_Target - XML parsing info:

0/865 packets of type RMAP\_read\_command  
0/865 packets of type RMAP\_write\_command  
518/865 packets of type RMAP\_read\_reply

XML file loaded: rmap\_packets.xml

LOAD XML



# Nominal Data Analysis

- Packet content
- Character level
- Time between characters
- Presence of special characters
- Single data value

Packet 2 content

Packet ID: 2

SpaceWire\_DUT

HEADER: 238

SIZE: 141

TERM: EOP

Time Scale: ☐ ms ☐ us ☒ ns

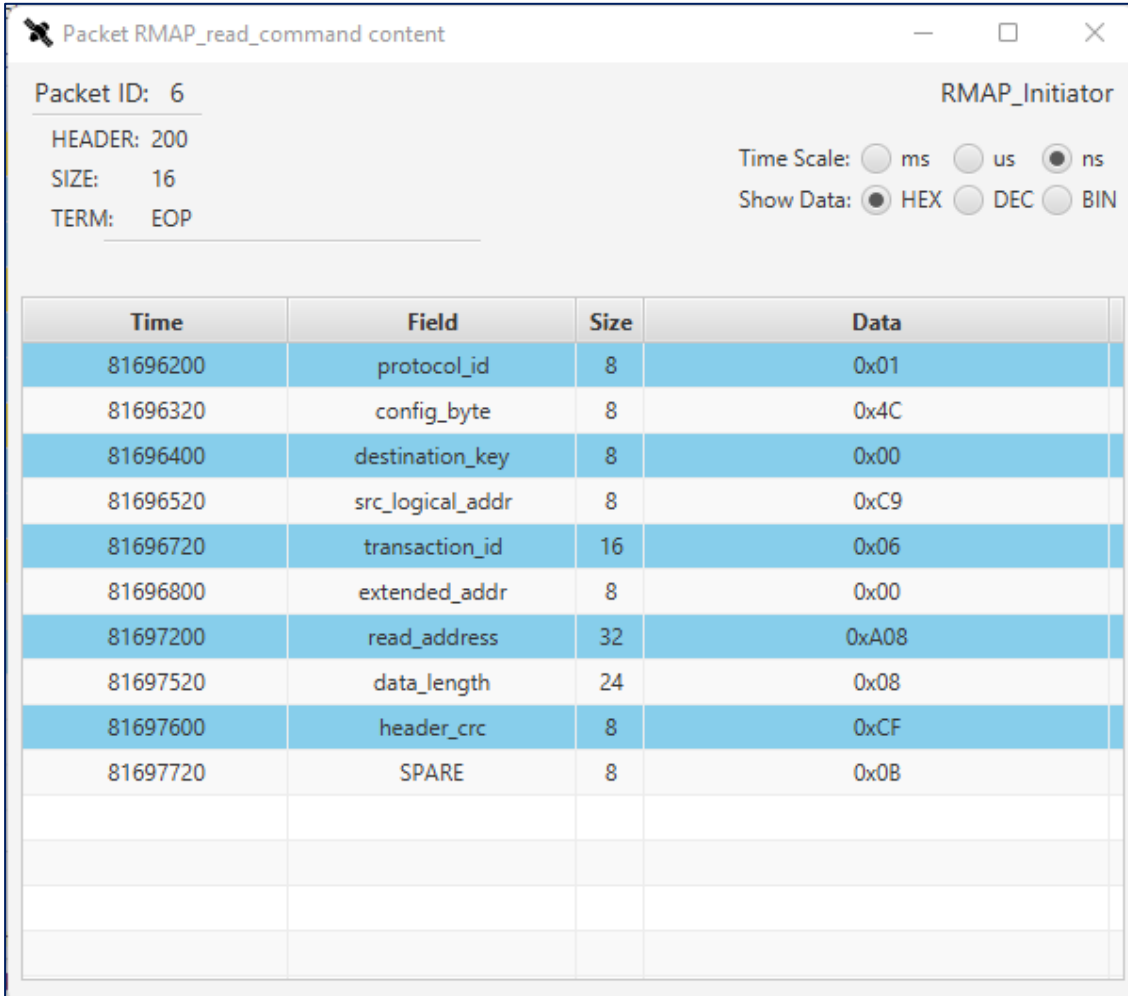
Show Data: ☒ HEX ☐ DEC ☐ BIN

SPEC CHAR: ☒ FCT ☐ NULL ☐ SYNC ☐ TIMECODE

Id	Time	Data
1	40714160	0xEE
2	40714240	0x4C
3	40714320	0x04
4	40714440	0x00
5	40714560	0x00
6	40714640	0x0A
7	40714720	0x09
8	40714840	0x00
9	40714960	0x00
10	40715040	0xDC
11	40715120	0x05
12	40715240	0x00
13	40715320	0x00
14	40715440	0x00

# Test Results – RMAP packets content

- Data organized as in SpW PDL
- Data bytes are grouped and named as indicated by the user
- More readable
- Inspect packet content is easier



Packet ID: 6

HEADER: 200  
SIZE: 16  
TERM: EOP

RMAP\_Initiator

Time Scale: ☐ ms ☐ us ☒ ns  
Show Data: ☒ HEX ☐ DEC ☐ BIN

Time	Field	Size	Data
81696200	protocol_id	8	0x01
81696320	config_byte	8	0x4C
81696400	destination_key	8	0x00
81696520	src_logical_addr	8	0xC9
81696720	transaction_id	16	0x06
81696800	extended_addr	8	0x00
81697200	read_address	32	0xA08
81697520	data_length	24	0x08
81697600	header_crc	8	0xCF
81697720	SPARE	8	0x0B

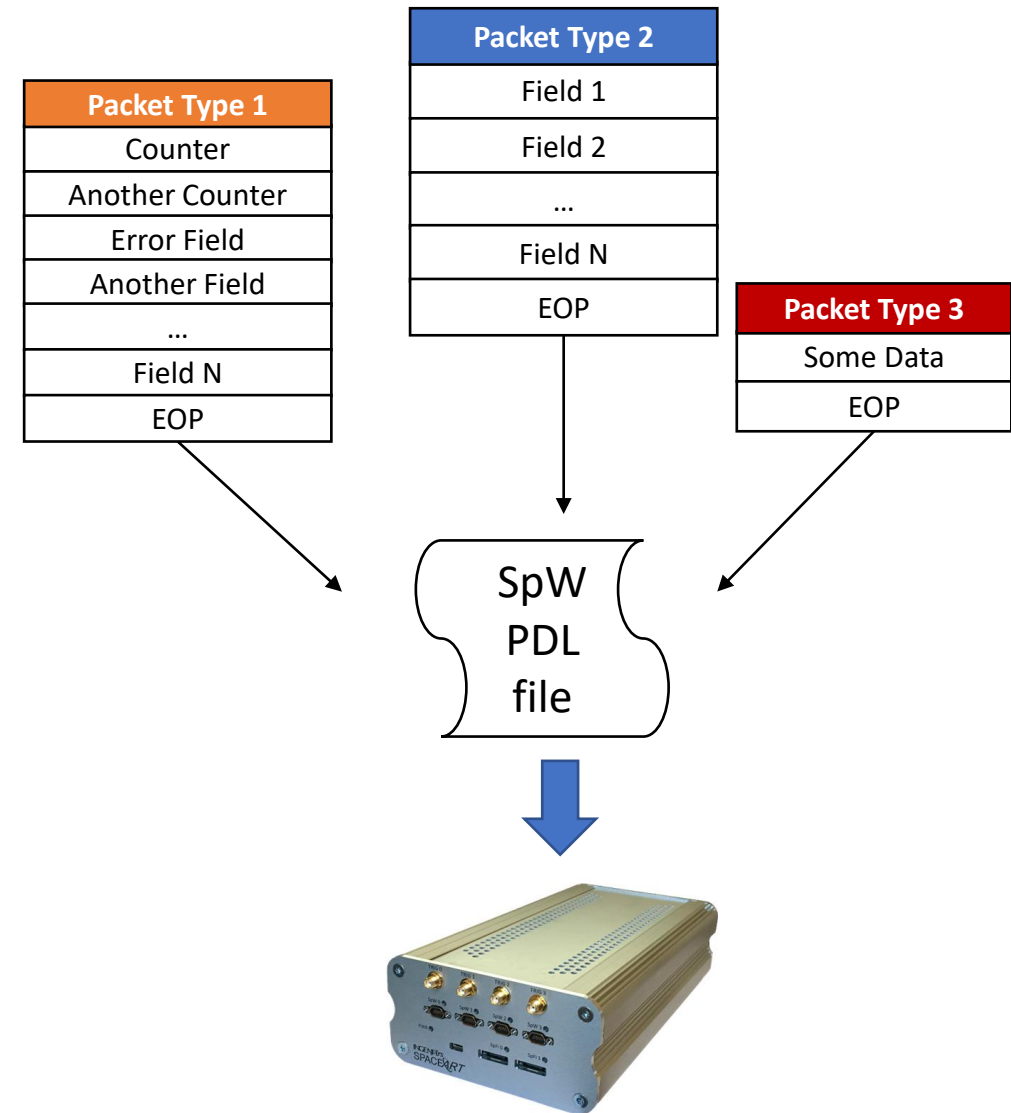
# Outline

- Introduction:
  - SpaceWire Packet Format
  - Protocol ID and RMAP
- SpW Packet Description Language:
  - Why?
  - Format
  - SpaceART SpW Sniffer Integration
- Use-Case
- Conclusion



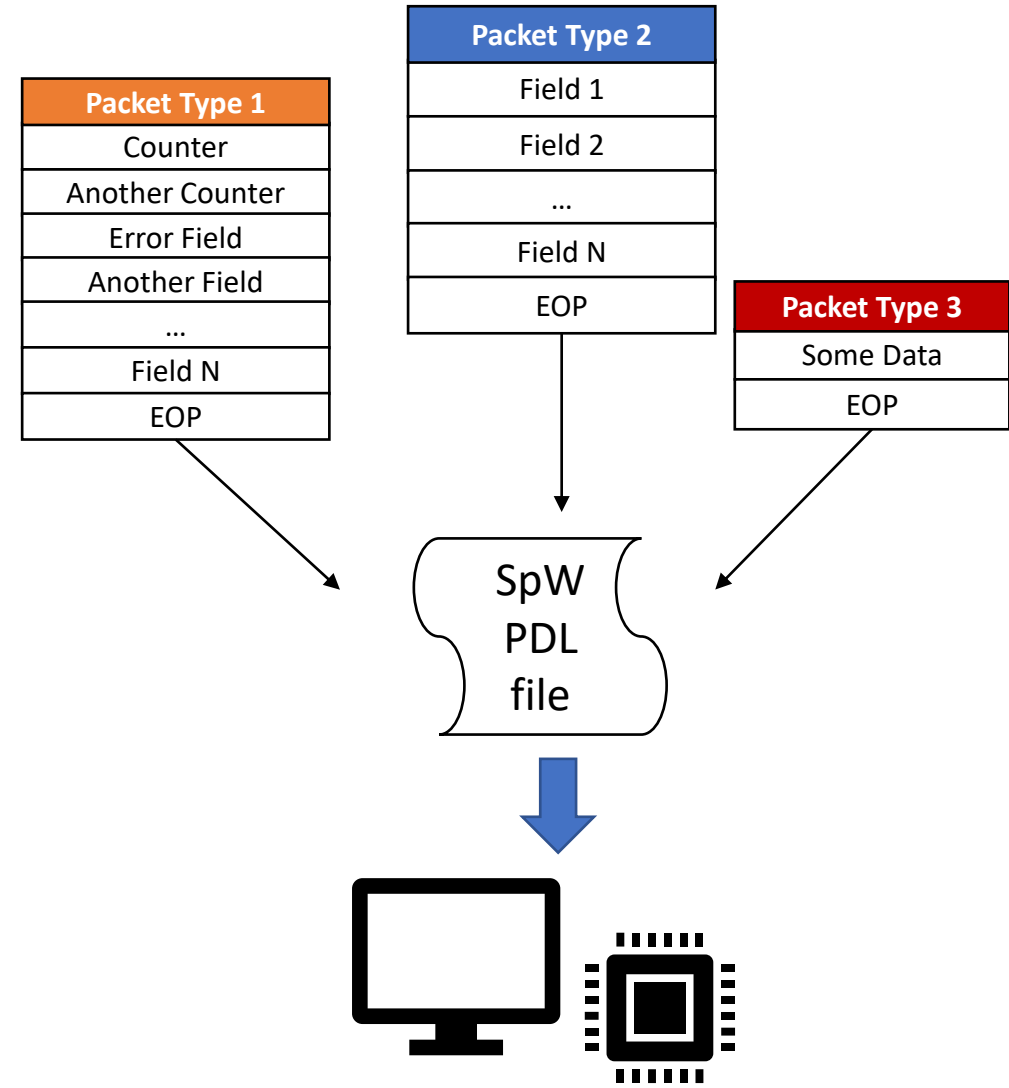
# Conclusion

- SpW PDL
  - SpW packet structure
  - XML flexibility
  - Human- and Machine-Readable
- SpaceART SpW Sniffer Integration
- Enhance user-friendly SpW data inspection



# Conclusion

- SpW PDL
  - SpW packet structure
  - XML flexibility
  - Human- and Machine-Readable
- SpaceART SpW Sniffer Integration
- Enhance user-friendly SpW data inspection
- Suitable for SpW-based applications



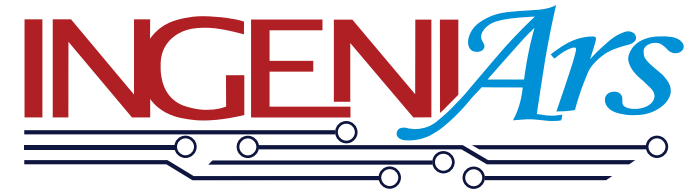
# Thank You



roberto.ciardi@phd.unipi.it  
roberto.ciardi@ingeniars.com



UNIVERSITÀ DI PISA



*The Art of Engineering*



Roberto Ciardi – University of Pisa

9th International SpaceWire and  
SpaceFibre Conference 2022

