

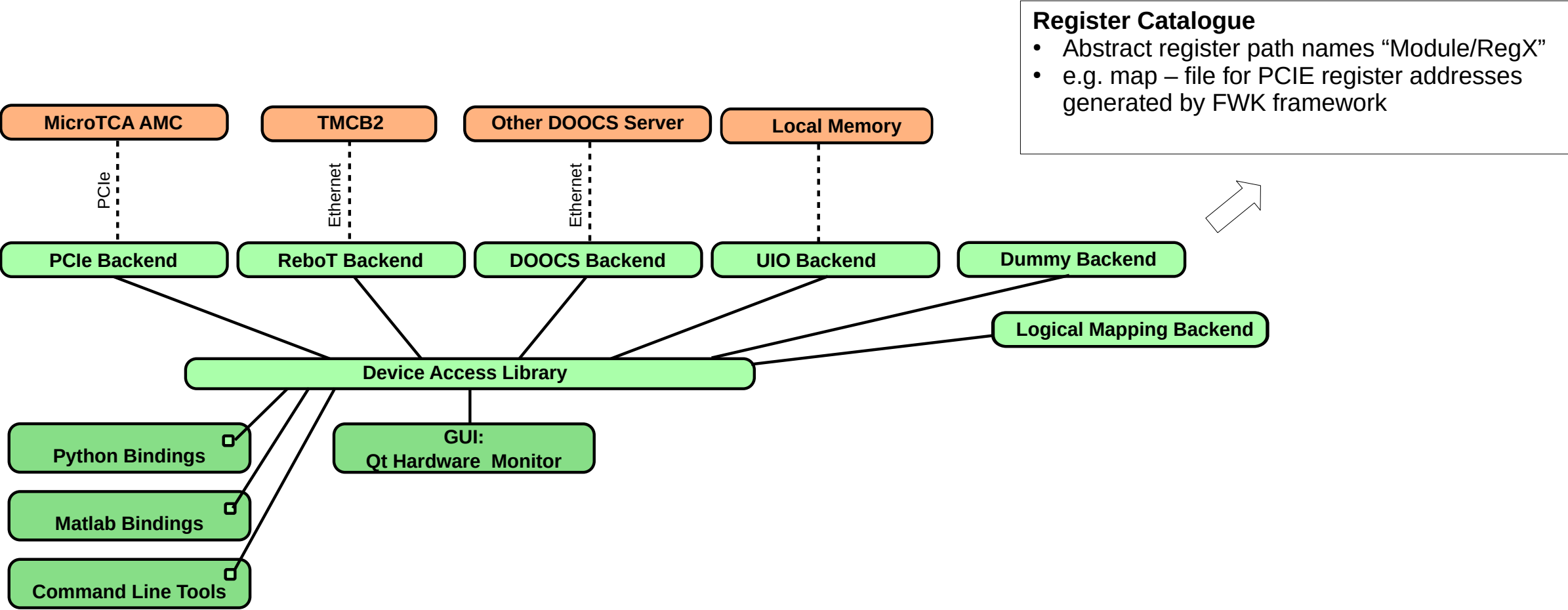
ChimeraTK Software Framework

Overview and OPC UA server hands-on

Dietrich Rothe, DESY - MSK
Dresden, Jul 5 2023

ChimeraTK Framework Overview

DeviceAccess



ChimeraTK Framework Overview

DeviceAccess

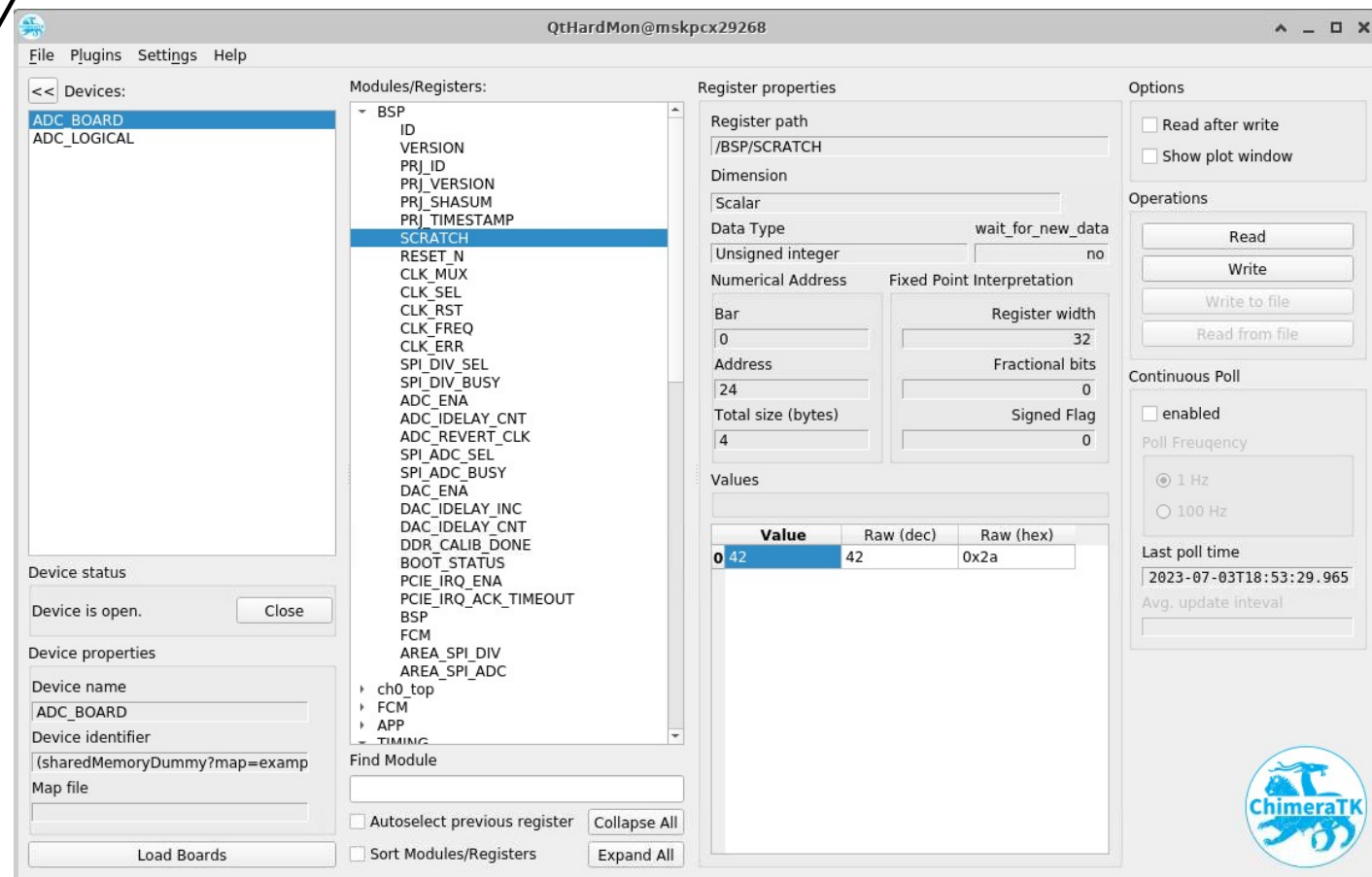
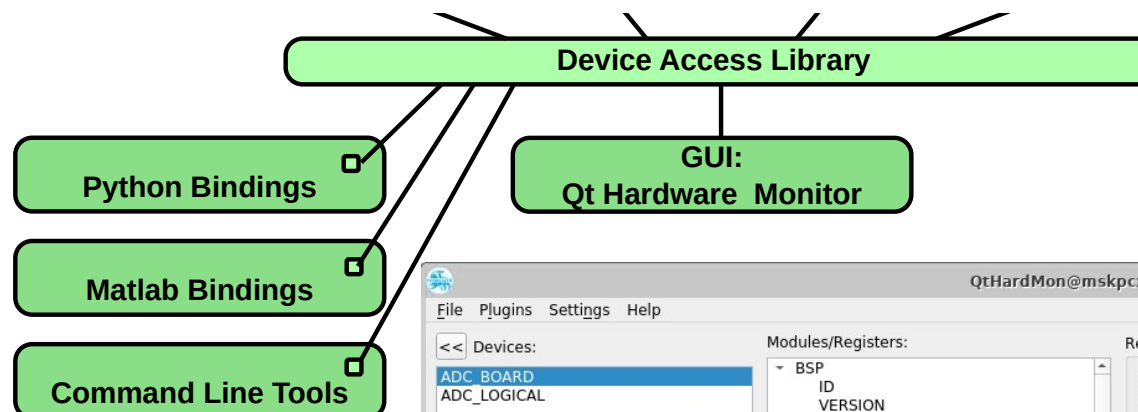
Accessor

- buffered in mem: use like a variable
- read(), write(): sync with hardware

```
import deviceaccess as da
import numpy as np
```

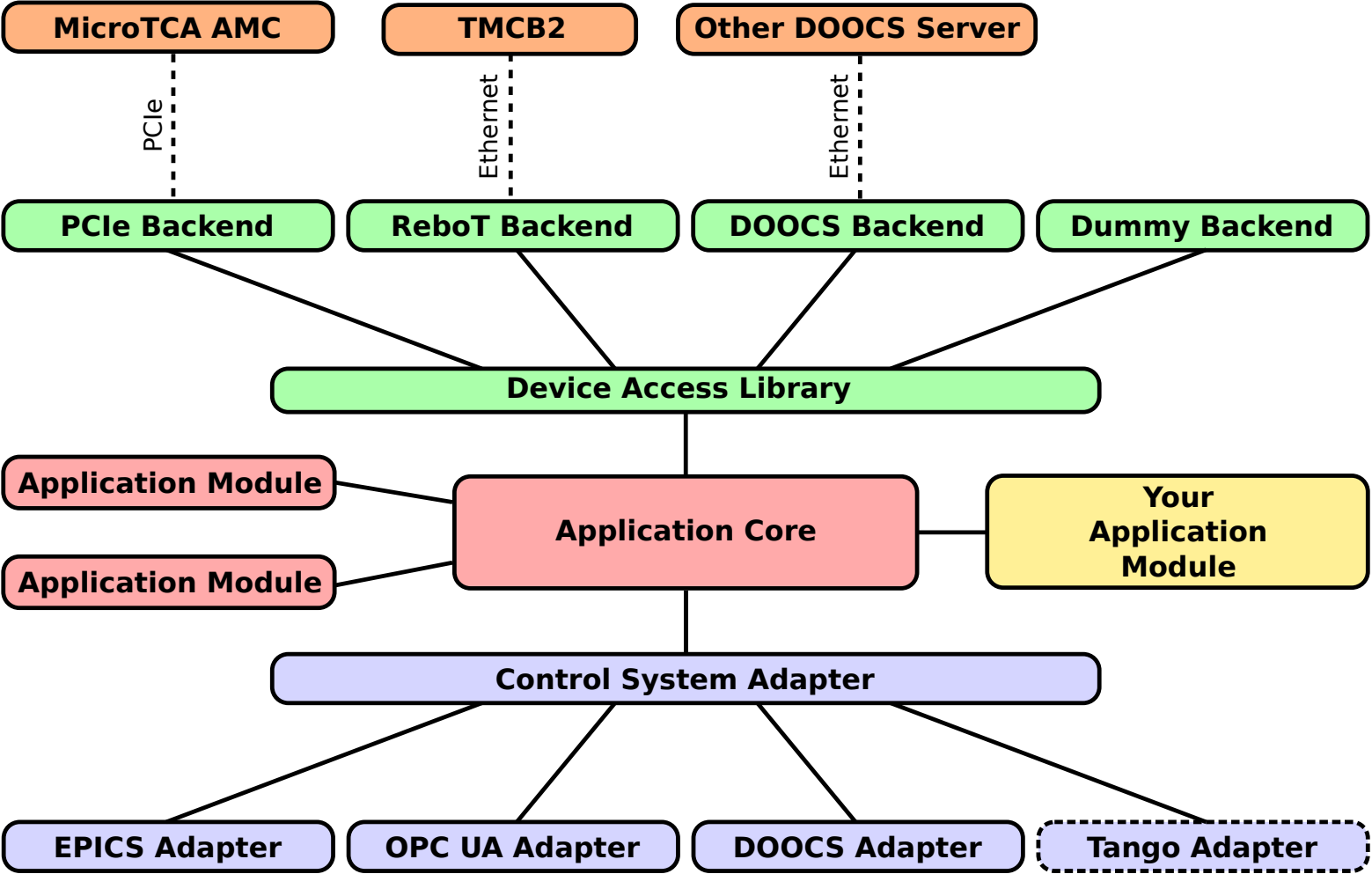
```
da.setDMapFilePath('devices.dmap')
d = da.Device('ADC_BOARD')
d.open()
```

```
acc = d.getOneDRegisterAccessor(np.uint32,
                                'BSP/SCRATCH')
acc[0] = 42
acc.write()
```



ChimeraTK Framework Overview

Ecosystem



Configure GenericDeviceServer

- Compile with OPC UA ControlSystemAdapter
- Add raw device
 - Protocol: *.dmap
 - Register set: *.map
 - Initialisation script!
Use DeviceAccess Python bindings
- Add Logical Device: *.xmap
- Configure simple software trigger, connect Device to trigger
- (Configure ControlSystemAdapter)
 - select, rename, convert, re-organize
 - History and filters

```
ADC_BOARD (xdma:xdma/slot3?map=example_dwc8vm1.mapp)
ADC_LOGICAL (logicalNameMap?map=example_dwc8vm1.xlmap)
```

```
@MAPFILE_REVISION 2.1.0-2-g815f78df-pcx29859
BSP.ID ..... 1 0x00000000
ch0_top.BSP ..... 19201 0x00000000
BSP.VERSION ..... 1 0x00000000
BSP.PRJ_ID ..... 1 0x00000000
BSP.PRJ_VERSION ..... 1 0x00000000
BSP.PRJ_SHASUM ..... 1 0x00000001
BSP.PRJ_TIMESTAMP ..... 1 0x00000001
BSP.SCRATCH ..... 1 0x00000001
RSP.RESET.M ..... 1 0x00000001
```

```
<configuration>
  <variable name="devices" type="string">
    <value i="0" v="ADC_LOGICAL"/>
  </variable>
  <module name="ADC_LOGICAL">
    <variable name="triggerPath" type="string" value="/msTimer/tick" />
    <variable name="pathInDevice" type="string" value="" />
    <variable name="initScript" type="string" value="./initscript.py" />
  </module>
  <variable name="periodicTimers" type="string">
    <value i="0" v="msTimer"/>
  </variable>
</configuration>
```

Logical Device

Logical name map

- Select & rename registers
- Apply simple formulas like rescaling, bit extraction
- **(Re-)define read/write access**
server initialises all writeable automatically!
- Define new variables
- Adjust data types
- Extract channels

⇒ Enables **better abstraction** in server logic!

Use with same DeviceAccess API as raw device

```
<logicalNameMap>
  <module name="board">
    <redirectedRegister name="version">
      <targetDevice>ADC_BOARD</targetDevice>
      <targetRegister>BSP/PRJ_VERSION</targetRegister>
    </redirectedRegister>
    <redirectedRegister name="scratch">
      <targetDevice>ADC_BOARD</targetDevice>
      <targetRegister>BSP/SCRATCH</targetRegister>
      <plugin name="forceReadOnly" />
    </redirectedRegister>
  </module>
  <module name="timing">
  <module name="channels">
    <redirectedChannel name="signal1">
      <targetDevice>ADC_BOARD</targetDevice>
      <targetRegister>DAQBUF/DAQ_CTRL_BUF0</targetRegister>
      <targetChannel>1</targetChannel>
      <plugin name="typeHintModifier"><parameter name="type">int32</parameter></plugin>
    </redirectedChannel>
  </module>
  <module name="attenuator">
    <redirectedRegister name="attSelectionMask">
    <variable name="attFactor">
      <type>float32</type>
      <value>0</value>
    </variable>
    <redirectedRegister name="attSetpoint">
      <targetDevice>ADC_BOARD</targetDevice>
      <targetRegister>RTM/ATT_VAL</targetRegister>
      <plugin name="math">
        <parameter name="enable_push_parameters" />
        <parameter name="formula"> f*x </parameter>
        <parameter name="f">/attenuator/attFactor</parameter>
      </plugin>
      <plugin name="typeHintModifier"><parameter name="type">int32</parameter></plugin>
    </redirectedRegister>
    <redirectedRegister name="attReadback">
      <targetDevice>ADC_BOARD</targetDevice>
      <targetRegister>RTM/ATT_VAL</targetRegister>
      <plugin name="forceReadOnly" />
      <plugin name="typeHintModifier"><parameter name="type">int32</parameter></plugin>
    </redirectedRegister>
  </module>
</logicalNameMap>
```

Extend server by business logic

```
struct LogicExample : public ctk::ApplicationModule {
    using ctk::ApplicationModule::ApplicationModule;
    ChimeraTK::ArrayPushInput<uint32_t> timingCnt{this, "/ADC_LOGICAL/timing/triggerCnt", "", 4, "hw trigger counters"};
    ChimeraTK::ScalarOutput<std::string> status{this, "status", "(no unit)", "fpga app status"};

    void mainLoop() override {
        // device is already initialized and current input values loaded
        uint32_t lastCnt = 0;
        while(true) {
            status = timingCnt[1] != lastCnt ? "fpga app running" : "fpga app not running";
            lastCnt = timingCnt[1];
            writeAll();

            readAll();
        }
    }
};
```

A
U
T
O

Connection Code

Error handling

Device Init

Setpoint persistency

History

ApplicationCore

DeviceAccess & Python scripts

Control system adapters

GenericDeviceServer with OPC UA Adapter

Unified Automation UaExpert - The OPC Unified Architecture Client - last*

FileViewServerDocumentSettingsHelp

Address Space

No Highlight

Root

Objects

OPCUA-Adapter

ADC_BOARD_RO

APP

AREA_SPI_ADC

AREA_SPI_DIV

BSP

DAQ

DAQBUF

FCM

RTM

TIMING

ch0_top

ch13_top

ADC_LOGICAL

attenuator

attFactor

attReadback

attSelectionMask

attSetpoint

board

scratch

version

channels

signal1

timing

triggerCnt

Config

Devices

ADC_BOARD_RO

ADC_LOGICAL

deviceBecameFunctional

initScriptOutput

status

status_message

logicExample

status

msTimer

period

tick

Server

Types

Views

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp
1	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/board/version	version	33685504	UInt32	15:16:41.497
2	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/board/scratch	scratch	42	UInt32	15:16:41.497
3	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/attenuator/attSetpoint	attSetpoint	1	Int32	15:17:16.135
4	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/attenuator/attFactor	attFactor	2	Float	15:17:21.110
5	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/attenuator/attReadback	attReadback	2	Int32	15:17:21.497
6	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/attenuator/attSelectionMask	attSelectionMask	3	UInt16	15:17:32.310
7	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/timing/triggerCnt	triggerCnt	{2349912457,871478886,2349912459,2...}	UInt32	15:27:40.497
8	localhost	NS1 String OPCUA-Adapter/logicExample/status	status	fpga app running	String	15:18:14.497
9	localhost	NS1 String OPCUA-Adapter/ADC_LOGICAL/channels/signal1	signal1	{-139,-130,-132,-144,-144,-144,-13...}	Int32	15:27:40.497
10	localhost	NS1 String OPCUA-Adapter/msTimer/period	period	1000	UInt32	15:24:38.120
11	localhost	NS1 String OPCUA-Adapter/msTimer/tick	tick	669	UInt64	15:27:40.497
12	localhost	NS1 String OPCUA-Adapter/Devices/ADC_LOGICAL/initScriptOutput	initScriptOutput	----- ADC_LOGICAL initialisation SUCCESS!	String	15:16:28.165

Address Space

Project

Thank you

References

Source repositories

- ChimeraTK <https://github.com/ChimeraTK>
- GenericDeviceServer <https://github.com/ChimeraTK/GenericDeviceServer.git>

Ubuntu 20.04 packages [DESY DOOCS repository](#)

API documentation <https://chimeratk.github.io/>

Build details

```
git clone https://github.com/ChimeraTK/GenericDeviceServer.git
git switch drothe/dresen23ard-demo

sudo apt install libchimeratk-applicationcore-dev libchimeratk-controlsystemadapter-opcuaadapter-dev libchimeratk-
deviceaccess-doocsbackend-dev python3-mtca4upy
cmake -DADAPTER="OPCUA" -S ../GenericDeviceServer/ .
./opcua-generic-chimeratk-server01
```

ChimeraTK Framework Overview

DeviceAccess

Accessor

- buffered in mem: use like a variable
- read(), write(): sync with hardware

```
import deviceaccess as da
import numpy as np
```

```
da.setDMapFilePath('devices.dmap')
d = da.Device('ADC_BOARD')
d.open()
```

```
triggerCounters = d.getOneDRegisterAccessor(np.uint32,
      'TIMING/TRIGGER_CNT/DUMMY_WRITEABLE')
```

```
while True :
    triggerCounters[0] += 1
    triggerCounters.write()
```

