Improvements in ChimeraTK.

Martin Killenberg

6th December 2018

7th MicroTCA Workshop for Industry and Research
DESY, Hamburg

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

ChimeraTK — A tool kit for control application development.

What does a control application do?

Task 1

- Talk to the hardware
- ChimeraTK DeviceAccess

Task 2

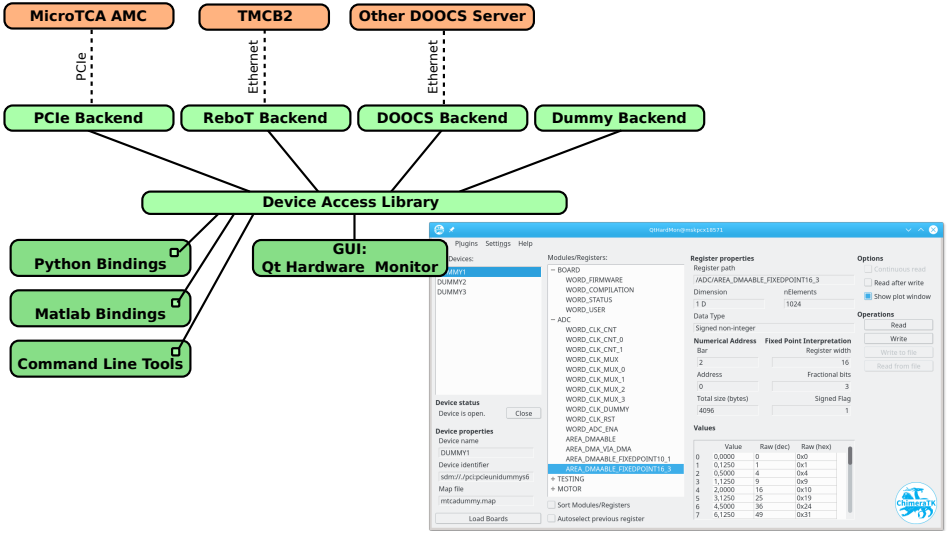
- Run the control algorithm
- ChimeraTK ApplicationCore

Task 3

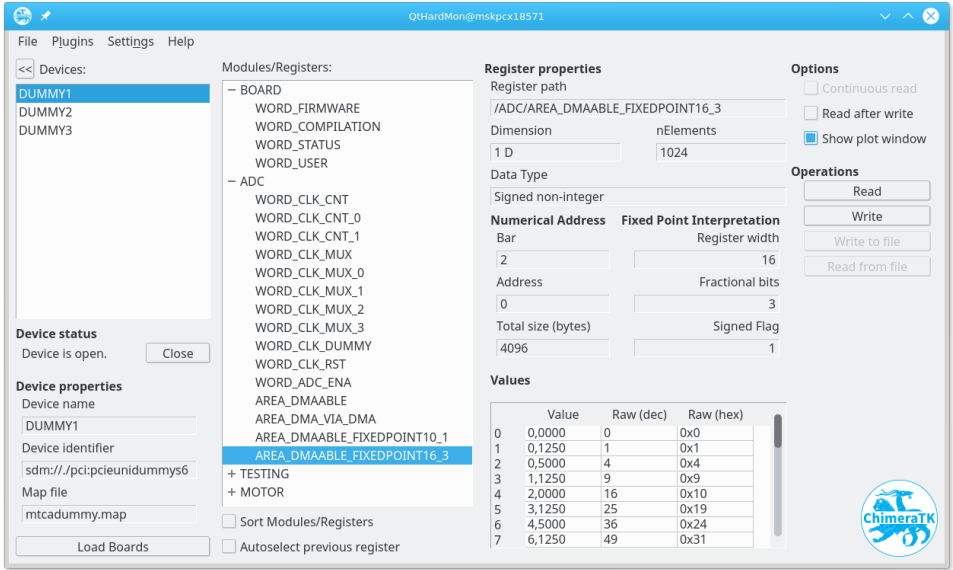
- Interface to the control system
- ChimeraTK ControlSystemAdapter

DeviceAccess.

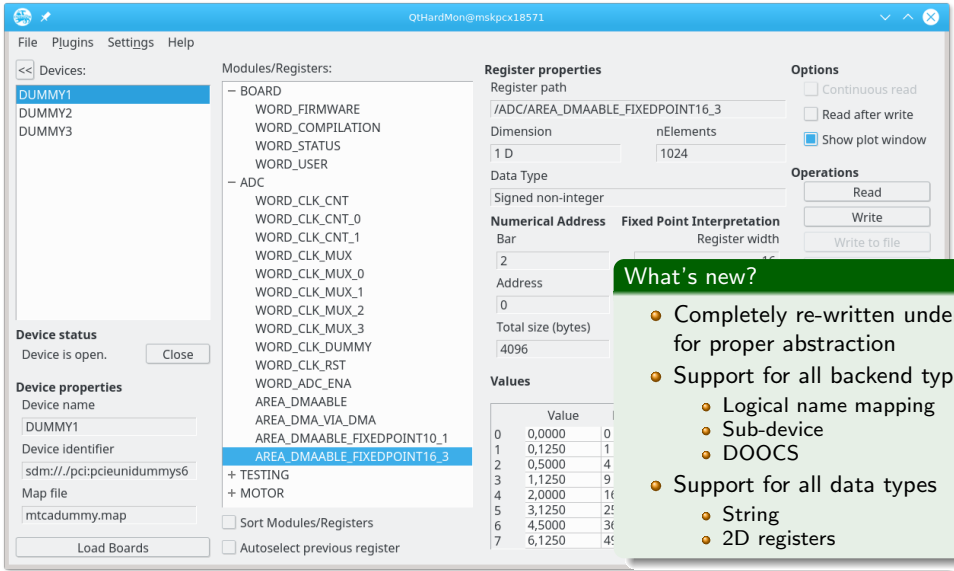
DeviceAccess Tools



QtHardMon v01.00



QtHardMon v01.00



What's new?

- Completely re-written under the hood for proper abstraction
- Support for all backend types
  - Logical name mapping
  - Sub-device
  - DOOCS
- Support for all data types
  - String
  - 2D registers

DeviceAccess: Device Mapping

Example device map file

```
#alias_name     device_descriptor
ADC_SLOT1       (pci:pcieunis1?map=my_adc_firmware.map)
ADC_SLOT2       (pci:pcieunis2?map=my_adc_firmware.map)
CONTROLLER      (pci:pcieunis3?map=my_controller_firmware.map)

@LOAD_LIB       /usr/lib/libChimeraTK-DeviceAccess-DoocsBackend.so
TIMER           (doocs:XFEL.RF/TIMER/LLA0M)
```

New: ChimeraTK Device Descriptor (CDD)

(backend_type:address?key1=value1&key2=value2)

Syntax

- Surrounded by parentheses – CDDs can be nested
- backend_type – Name of the backend, e.g. "pci", "dummy"
- address – Address of the device. The interpretation depends on the backend.
- keyX=valueX – List of key-value pairs. The interpretation depends on the backend.

Available Backends

Build-in

pci	PCI-Express (µTCA AMC)
rebot	***R**egister **b**ased **o**ver **T**CP*, lightweight, TPC/IP based inhouse protocol
subdevice	Show part of address space as own logical device (µTCA RTM)(new)
logicalNameMap	Rename and re-organise registers
dummy	Simulate register space in RAM
sharedMemoryDummy	Dummy with address space in shared memory (new)

Loadable plugins

doocs	DOOCS client interface[1]
modbus	Modbus client interface (under development)

Various dedicated dummies for tests

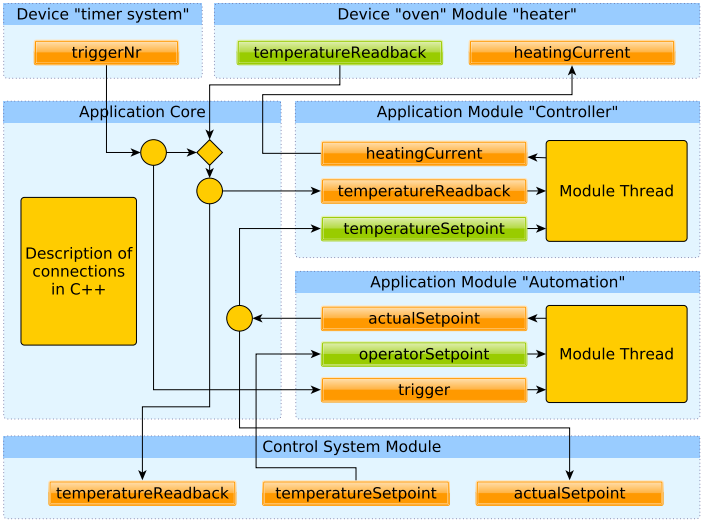
Planned

epics	Native EPICS client interface
opu-ua	OPC UA client interface

[1] Can also read EPICS channels

ApplicationCore.

ApplicationCore



Modules

- Input/output variables
- Application Modules
  - One thread per module
- Special modules
  - Device module
  - Control system module

Connection Code

- Connect application modules
- Triggering
  - Read multiple variables synchronously
  - Synchronise application modules to HW trigger

New Application Modules

Config Reader

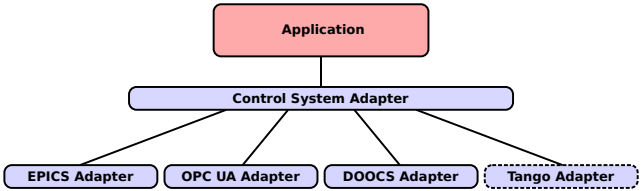
- Read config parameters from an XML config file
- Connect variables to other modules
- Already available when connecting
  - Instantiate modules depending on configuration

Periodic Trigger

- Needed in all applications that don't have hardware readout triggers
- Testable mode
  - Full control when a trigger is send
  - Ideal for automated testing and debugging

Control System Adapter.

Control System Adapter



Connect ChimeraTK applications to various control system middlewares

Available adapters

- DOOCS adapter
- OPC-UA adapter
- EPICS 3 device support adapter
- EPICS 3 IOC adapter **(new)**

New: EPICS 3 IOC adapter

- ChimeraTK application → complete EPICS IOC
- Can be configured via records file
- One IOC per application

Application Example: LLRF Control Server

LLRF control server used in production at

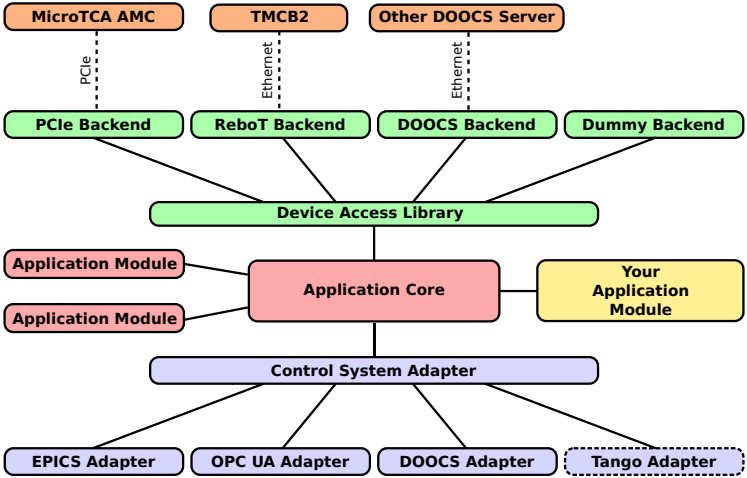
- REGAE at DESY (DOOCS adapter)
- CMTB at DESY (DOOCS adapter)
- SINBAD at DESY (DOOCS adapter)(in preparation)

- FLUTE at KIT (DOOCS adapter)
  - Will change to EPICS

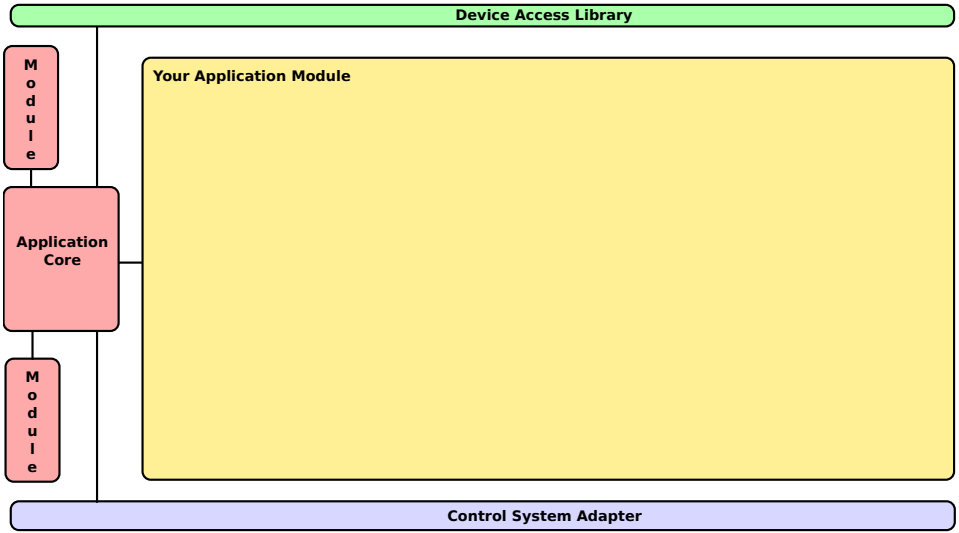
- TARLA in Ankara (EPICS IOC adapter)
- MESA in Mainz (EPICS IOC adapter)(in preparation)

- ELBE at HZDR (OPC UA adapter)

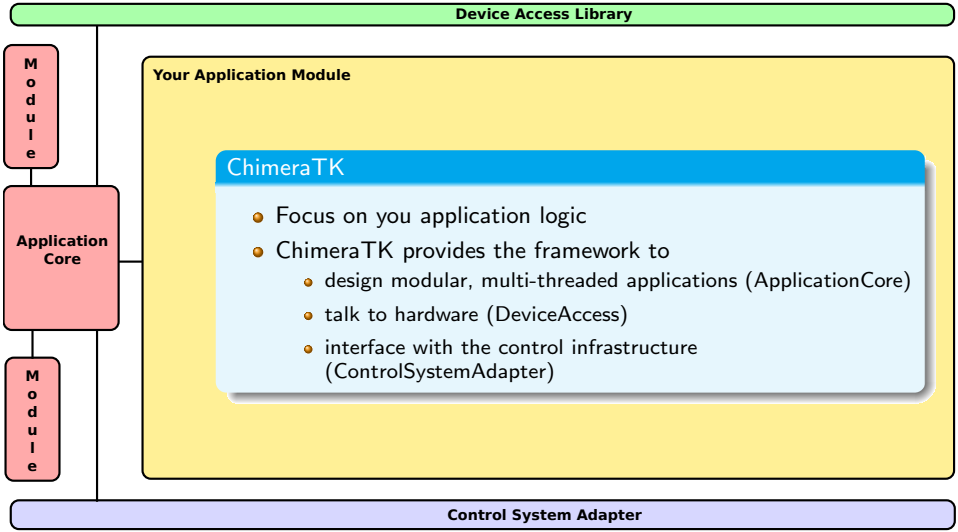
Summary


MicroTCA AMC
TMCB2
Other DOOCS Server
PCIe
Ethernet
Ethernet
PCIe Backend
RebоT Backend
DOOCS Backend
Dummy Backend
Device Access Library
Application Module
Application Module
Application Core
Your Application Module
Control System Adapter
EPICS Adapter
OPC UA Adapter
DOOCS Adapter
Tango Adapter

Summary


Device Access Library
Module
Your Application Module
Application Core
Module
Control System Adapter

Summary


Device Access Library
Module
Application Core
Module
Your Application Module
ChimeraTK
Focus on you application logic
ChimeraTK provides the framework to
design modular, multi-threaded applications (ApplicationCore)
talk to hardware (DeviceAccess)
interface with the control infrastructure (ControlSystemAdapter)
Control System Adapter

Software Repositories

All software is published under the GNU GPL or the GNU LGPL.

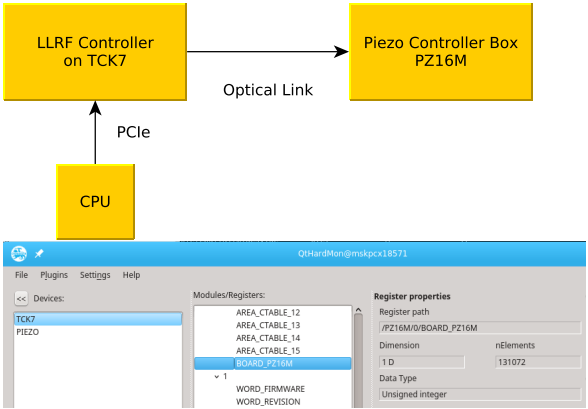
- ChimeraTK source code: https://github.com/ChimeraTK
- Ubuntu 16.04 packages are available in the DESY DOOCS repository.
- Launchpad PPA for Ubuntu is planned

Documentation and Tutorials

- API documentation https://chimeratk.github.io/
- Tuesday's tutorials on the MicroTCA Workshop Indico page
- DeviceAccess live demo https://github.com/killenb/DeviceAccess_live_demo
- e-mail support: chimeratk-support@desy.de

Backup.

Subdevice Backend



- Firmware maps the register space of a subdevice (PIEZO) into a 1D register of it's own address space (TCK7).
  (Usually offset address in a numerically addressed space).
- Both devices have firmwares which evolve separately.
- The Subdevice backend allows to access the subdevice through the parent device as a separate logical entity.

⇒ Separate map file to describe the subdevice.

```
#alias device_descriptor
TCK7   (pci:llrfutcs4?map=llrf_ctrl_tck7b_acc1_r2097.mapp)
PIEZO  (subdevice:area,TCK7,PZ16M.0.BOARD_PZ16M?map=piezo_pz16m_acc1_r2323.mapp)
```

Simulate address space of device in shared memory

- Loads map file used for real device
- Device stays active as long at least one client has opened it
- Last closing client clears the shared memory

Use cases

- Run on your desktop PC without hardware
- Debugging if faulty software can damage the hardware
- Prepare test cases on the fly with QtHardMon

What can I do with Logical Name Mapping?

Arrange the register content to logically match your application

- Rename registers
- Add constant registers or dummy registers
- Extract channels from 2D registers and give it a name
- Extract scalars from 1D registers and give it a name
- Extract bits from a scalar register

Logical Name Mapping Example

Abstract away cabling details

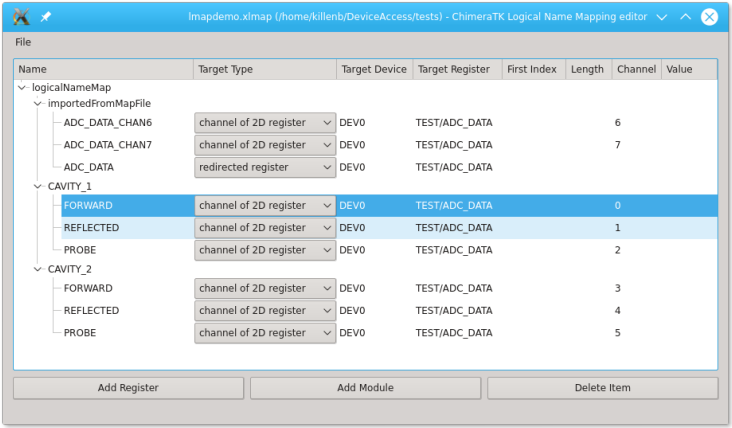
- Cavity with 3 signals: Forward, reflected, probe.
- Recorded on 8 channel ADC (2D array with data): adc_data[8][1024]
- Cabling:
  - Cavity 1 on channels 0..2
  - Cavity 2 on channels 3..5

Logical name mapping

adc_data[0]	→	cavity1/forward
adc_data[1]	→	cavity1/reflected
adc_data[2]	→	cavity1/probe
adc_data[3]	→	cavity2/forward
adc_data[4]	→	cavity2/reflected
adc_data[5]	→	cavity2/probe

You don't have to fiddle with channel numbers in you cavity module.
Use the logical names *forward, reflected, probe*.

LMAP Editor



- Import map file as starting point
- Modify the mapping
- Save and load logical name mapping

Tool under development.
Please give feedback or implement missing features.