ChimeraTK.

Hardware Access and Control System Integration (not only) for MicroTCA

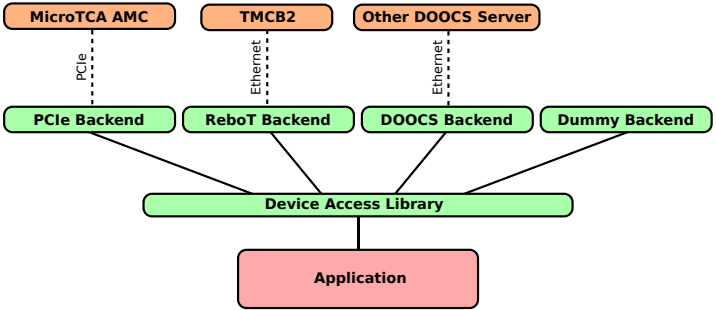
Martin Killenberg

M. Heuer, M. Hierholzer, T. Kozak, L. Petrosyan, C. Schmidt, N. Shehzad,
G. Varghese, M. Viti, *DESY, Hamburg, Germany*
S. Marsching, *aquenos GmbH, Baden-Baden, Germany*
A. Piotrowski, *FastLogic Sp. z o.o., Łódź, Poland*
C. P. Iatrou, J. Rahm, *Technische Universität Dresden, Dresden, Germany*
R. Steinbrück, M. Kuntzsch, *HZDR, Dresden-Rossendorf, Germany*
P. Prędki *Łódź University of Technology, Łódź, Poland*
A. Dworzanski, K. Czuba, *Warsaw University of Technology, Warsaw, Poland*

6th MicroTCA Workshop
DESY, Hamburg
7. December 2017

ChimeraTK

Control system and **H**ardware **I**nterface with **M**apped and **E**xtensible **R**egister-based device **A**bstraction **T**ool **K**it



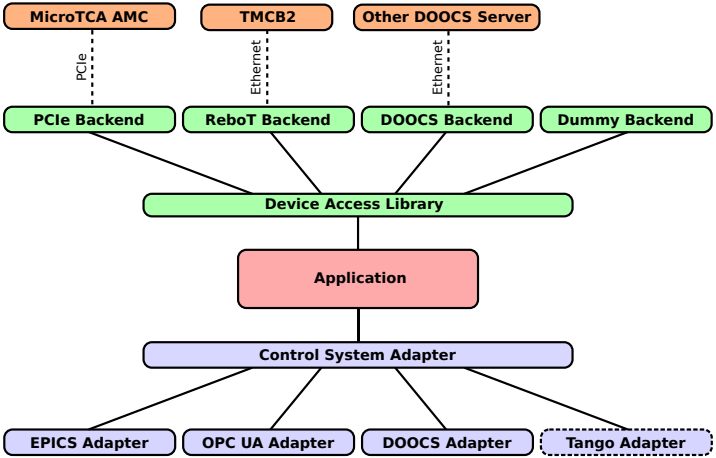
DeviceAccess Library

Last year's presentation

- Register based devices
- Abstracted interface with backends

ChimeraTK

Control system and **H**ardware **I**nterface with **M**apped and **E**xtensible **R**egister-based device **A**bstraction **T**ool **K**it



DeviceAccess Library

Last year's presentation

- Register based devices
- Abstracted interface with backends

ControlSystemAdapter

- Making application implementations independent from the middleware

ApplicationCore

- Unify concepts of DeviceAccess and ControlSystemAdapter
- Enhance functionality

ControlSystemAdapter

Task

Complex control algorithms should be used with different control systems.

LLRF server

- $\mathcal{O}(400)$ process variables
- iterative learning algorithms
- feed forward table calculations

$\Rightarrow$

Facilities

- DOOCS at XFEL/FLASH (DESY)
- WinCC/OPC UA at ELBE (HZDR)
- EPICS 3 at FLUTE (KIT)
- EPICS 4 at TARLA (Ankara)

ControlSystemAdapter

Task

Complex control algorithms should be used with different control systems.

LLRF server

- $\mathcal{O}(400)$ process variables
- iterative learning algorithms
- feed forward table calculations

$\Rightarrow$

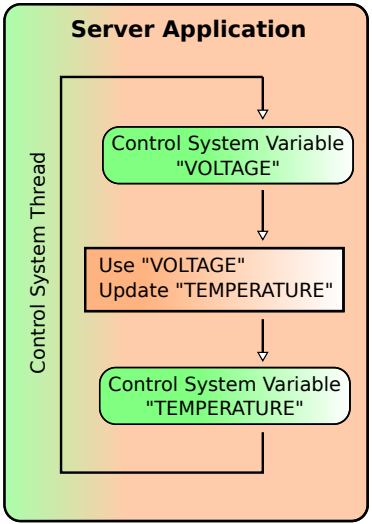
Facilities

- DOOCS at XFEL/FLASH (DESY)
- WinCC/OPC UA at ELBE (HZDR)
- EPICS 3 at FLUTE (KIT)
- EPICS 4 at TARLA (Ankara)

Requirements For Abstraction

- Keep application code control system independent
- The algorithm must interact with the control system
- Use functionality provided by the control system
- No device-dependent code on the control system side

A Typical Device Server (Without Adapter)



- Control system data types used inside the algorithm
- Control system variables might not be thread safe
- Control system variables can be locking/blocking
- Threading often handled by control system

Abstraction is needed

Required abstraction for the ControlSystemAdapter:
Separate device logic and control system integration

Application code

- Define process variables
- Implement algorithms
- Talk to hardware

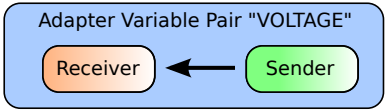
Control system "code"

- Publish process variables via middleware
- Define variable name visible in control system
- Define middleware dependent features/data types
  - Histories
  - Display properties
- Application independent, configured via config file

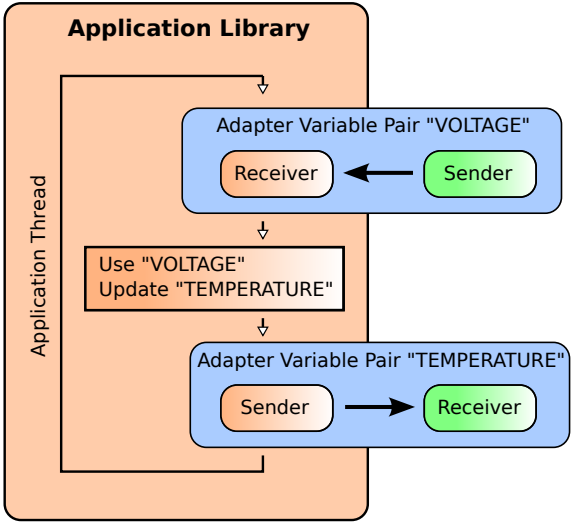
Application and control system dependent code

- Avoid it!

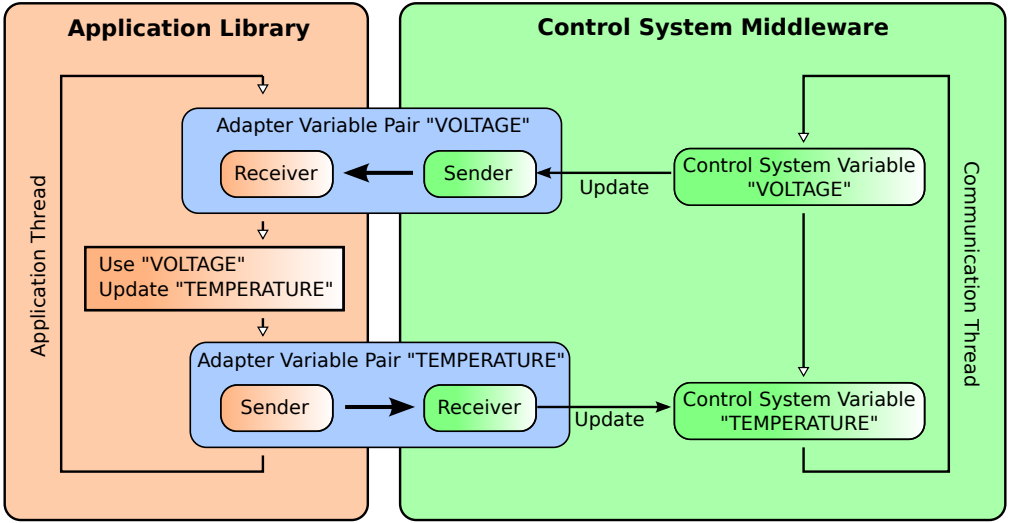
A Device Using The ControlSystemAdapter



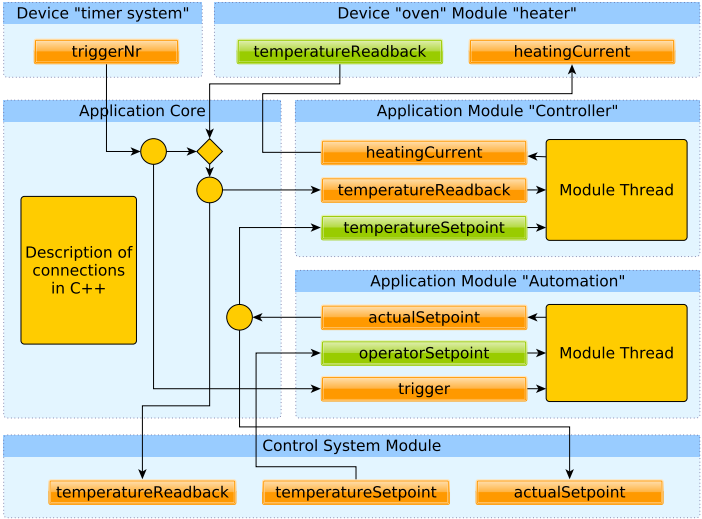
A Device Using The ControlSystemAdapter



A Device Using The ControlSystemAdapter



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

Details Matter

Threading system with efficient implementation:
Sender-receiver pairs based on lock-free queues and futures

- Message driven design: sender pushes data into the queue
- No dead-locks possible
- Threads are not unintentionally blocked
- Wait for data without using CPU time

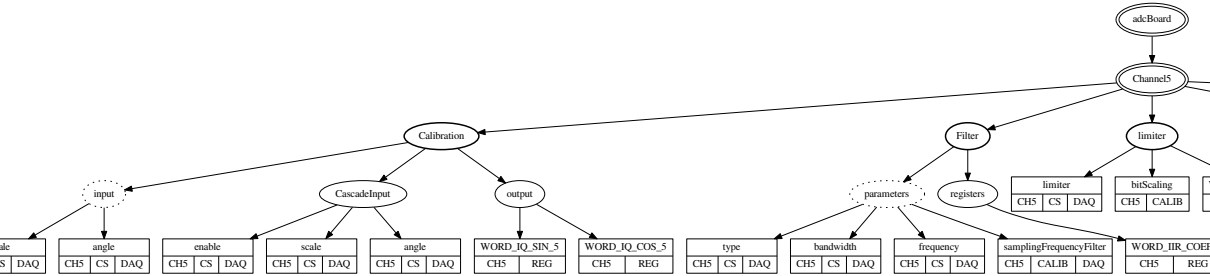
API fosters designs with small modules

- Clean design
- Improved performance on multi-core CPUs
- High scalability

Use C++11 to avoid boiler plate code where possible

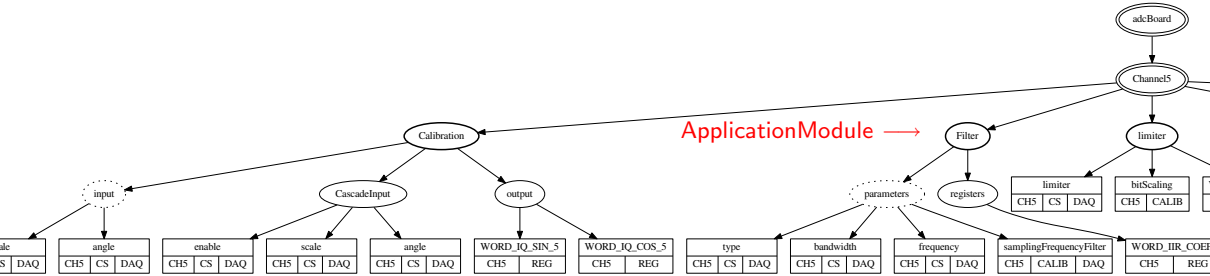
Information modelling

Automatically generated graph for LLRF server, module AdcBoard, channel 5 (part)



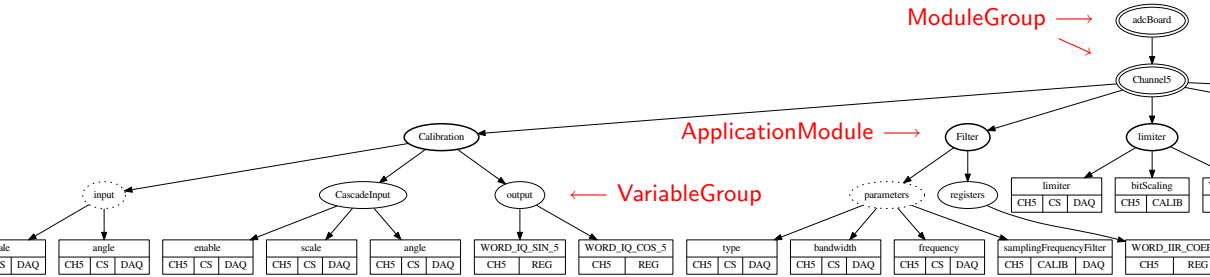
Information modelling

Automatically generated graph for LLRF server, module AdcBoard, channel 5 (part)



Information modelling

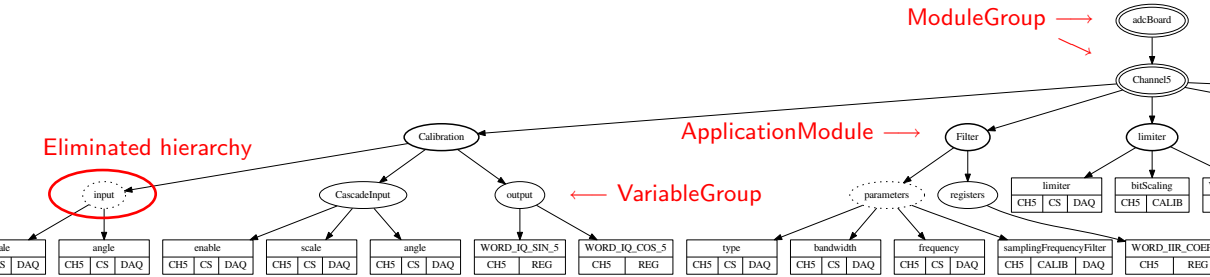
Automatically generated graph for LLRF server, module AdcBoard, channel 5 (part)



- Module groups and variable groups: Add hierarchies

Information modelling

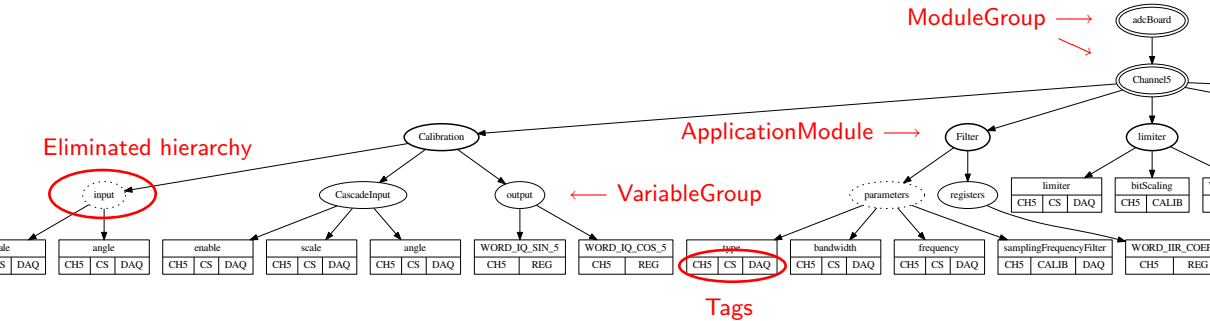
Automatically generated graph for LLRF server, module AdcBoard, channel 5 (part)



- Module groups and variable groups: Add hierarchies
- Eliminate hierarchies

Information modelling

Automatically generated graph for LLRF server, module AdcBoard, channel 5 (part)



- Module groups and variable groups: Add hierarchies
- Eliminate hierarchies
- Tags: Simplify the connection code

Control System Integration Example: Variable Names

Input tree

```
|-- oven1
|   |-- controller
|   |   |-- temperatureSetpoint
|   |   |-- temperatureReadback
|   |
|   |-- supplyVoltages
|
|-- oven2
    |-- controller
    |   |-- temperatureSetpoint
    |   |-- temperatureReadback
    |
    |-- supplyVoltages
```

System Integration

Control System Integration Example: Variable Names

Input tree

```
|-- oven1
|   |-- controller
|   |   |-- temperatureSetpoint
|   |   |-- temperatureReadback
|   |
|   |-- supplyVoltages
|
|-- oven2
    |-- controller
    |   |-- temperatureSetpoint
    |   |-- temperatureReadback
    |
    |-- supplyVoltages
```

Output tree

```
|-- breadOven
|   |-- temperature
|
|-- cookieOven
    |-- temperature
```

System Integration

- Names need to be adapted for the facility (manufacturer does not know if oven is used for bread or cookies)
- Configure middleware-dependent features (display limits, . . .)

⇒ Do it in system integration via config file

Control System Integration Example: Variable Names

Input tree

```
|-- oven1
|   |-- controller
|   |   |-- temperatureSetpoint
|   |   |-- temperatureReadback
|   |
|   |-- supplyVoltages
|
|-- oven2
    |-- controller
    |   |-- temperatureSetpoint
    |   |-- temperatureReadback
    |
    |-- supplyVoltages
```

Output tree

```
|-- breadOven
|   |-- temperature
|
|-- cookieOven
|   |-- temperature
|
|-- expert
|   |-- temperatureSetpoints
|       |-- breadOven
|       |-- cookieOven
|
|-- powerSupply
    |-- fuse1
    |   |-- breadOvenVoltages
    |
    |-- fuse2
        |-- cookieOvenVoltages
```

System Integration

- Names need to be adapted for the facility (manufacturer does not know if oven is used for bread or cookies)
- Configure middleware-dependent features (display limits, . . .)

⇒ Do it in system integration via config file

Status

What about the LLRF server...?

- LLRF server developed using the DOOCS adapter at DESY
- Re-linked with the OPC UA adapter and integrated at ELBE/HZDR

⇒ Works without changing a single line of C++ code!

- Server and OPC UA adapter need some refinement for production system

Presentation:
"ChimeraTK OPC UA Adapter for the Integration of MicroTCA.4 based digital LLRF" by Reinhard Steinbrück

Poster:
"ApplicationCore: A Framework for Modern Control Applications at the Example of a Facility Independent LLRF Server" by Martin Hierholzer

Introduction to ChimeraTK: Summary

ControlSystemAdapter

- Use device logic with different control systems
- Implementations for DOOCS and OPC UA
- Epics 3 adapter is currently being updated

DeviceAccess Library

- Abstracted, register based hardware access
- Use real and virtual hardware, device servers
- Scripting tools and GUI

ApplicationCore Library

- Unifies DeviceAccess and ControlSystemAdapter
- Application modules
  - Input/output variables
- Hierarchical data model
  - Module and variable groups
  - Tags

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

- ChimeraTK: https://github.com/ChimeraTK
- EPICS 3 Adapter: http://oss.aquenos.com/svnroot/epics-mtca4u/

Slides and Source Code from Tuesday's Tutorial

- https://github.com/killenb/DeviceAccess_live_demo
- https://github.com/mhier/ApplicationCoreLiveDemo