Abstracted Hardware and Middleware Access in Control Applications

M. Killenberg, M. Heuer, M. Hierholzer, L. Petrosyan, C. Schmidt, N. Shehzad, T. Kozak, G. Varghese, M. Viti (DESY, Germany), S. Marsching (aquenos GmbH, Germany), A. Piotrowski (FastLogic, Poland), R. Steinbrück, M. Kuntzsch (HZDR, Germany), P. Prędki (Rapid Development, Poland), C. P. Iatrou, J. Rahm (TU Dresden, Germany), K. Czuba, A. Dworzanski (Warsaw Univ. of Technology, Poland)

The task

> Accelerator controls need complex devices servers

> Requires communication to FPGAs, micro-controllers, frontend and middelware PCs with different protocols via PCIe, Ethernet, etc.

> Devices should be used in various other facilities with different control systems
- XFEL and FLASH at DESY using DOOCS
- ELBE at HZDR using OPC UA
- FLUTE at KIT using EPICS 3
- TARLA in Ankara using EPICS 4

ChimeraTK

> Framework to abstract applications from the details of hardware and control system protocols

> Write device servers which are intrinsically control system independent

> Using modern C++ 11

> Open source software, (L)GPL

The DeviceAccess library

> Access to register-based devices

> Common interface to backends which implement different communication protocols

> RegisterAccessor objects represent registers as process variables (common interface with ControlSystemAdapter)

> Register name mapping: Identify registers by name instead of numerical address

> Device name mapping: Identify devices by functional name (independent from backend)

Available backends

> PCI Express

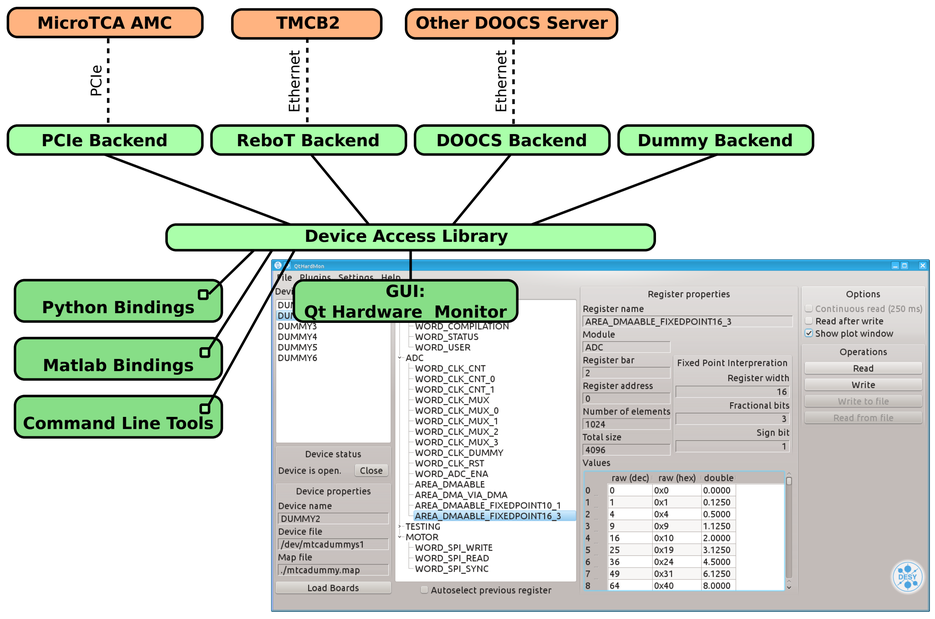
> Simple network protocol for FPGAs: RebоT (Register-based over TCP)

> DOOCS backend

> Dummy backend / VirtualLab

> LogicalNameMapping backend for more abstraction from implementation details of the firmware

Custom backends can be loaded at run time.



Software for interactive access and scripting

> Language bindings for Python and Matlab

> Linux command line tool

> Graphical user interface

Convenient tools for firmware development: Direct hardware access without having to write code.

ApplicationCore

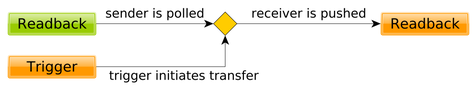
> Application modules implement the algorithms

> Connection code combines modules and creates an application

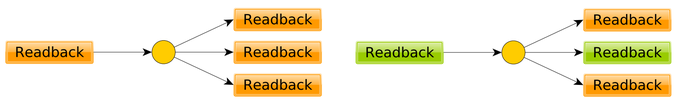
Connections

Requirement: Intuitive syntax that minimises number of code lines

> Use any pushing sender as trigger to connect a polled sender to a pushed receiver



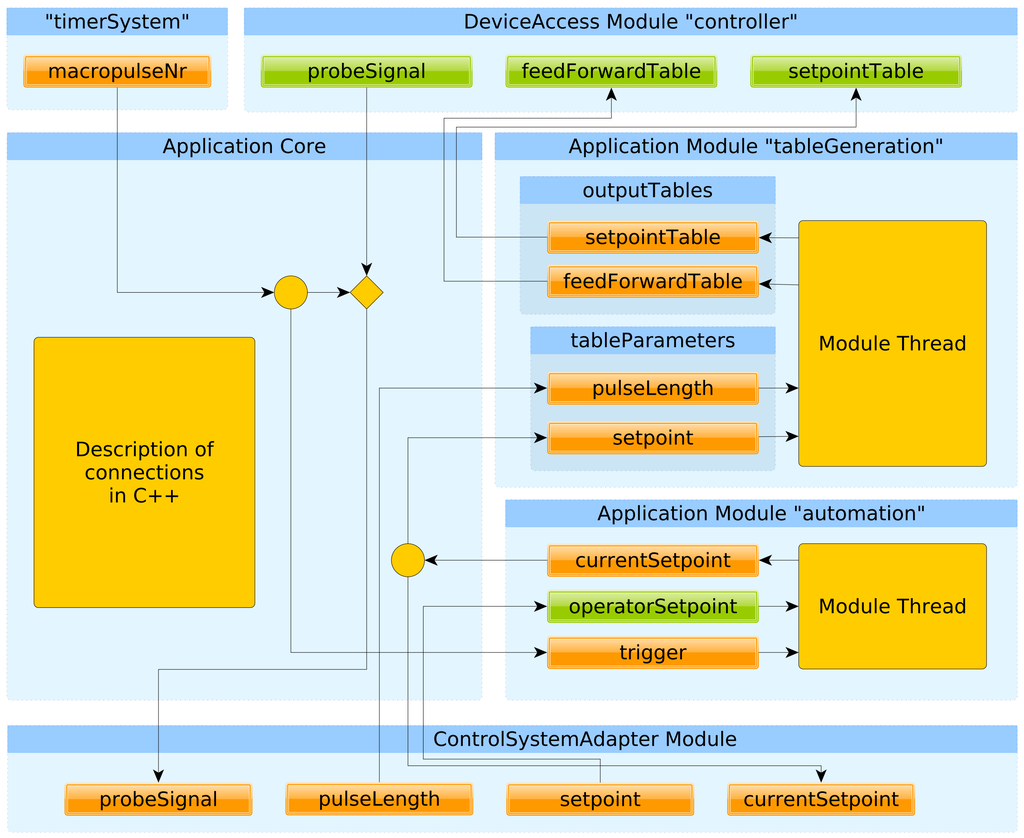
> "Fan out" to distribute variables



> Connect all variables with the same name in a single command

> Group variables and modules to structure the code

> Plot tree with variable content of an application



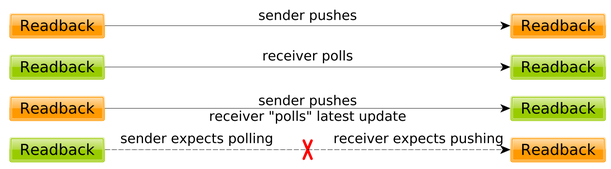
Application modules

Abstraction: If a module does not know if a process variable is coming from the hardware, the control system or another software module, it will not be sensitive to specifics of a particular middleware.

> Interface consists of input and output variables

> One thread per module

> Two types of variables:
- active sender pushes updates to passive receiver
- passive sender is polled by active receiver



> Use *blocking read* to synchronise to other threads and to hardware
- on a single variable
- on *all* variables or *any* variable in a group

> Hierarchical variable names

> Advanced modelling with tags on variables

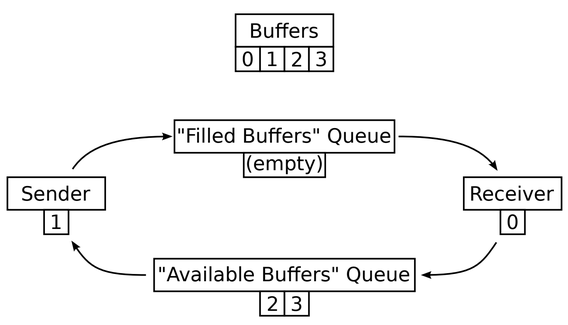
ControlSystemAdapter library

Process variables are implemented as lock-free sender/receiver pairs

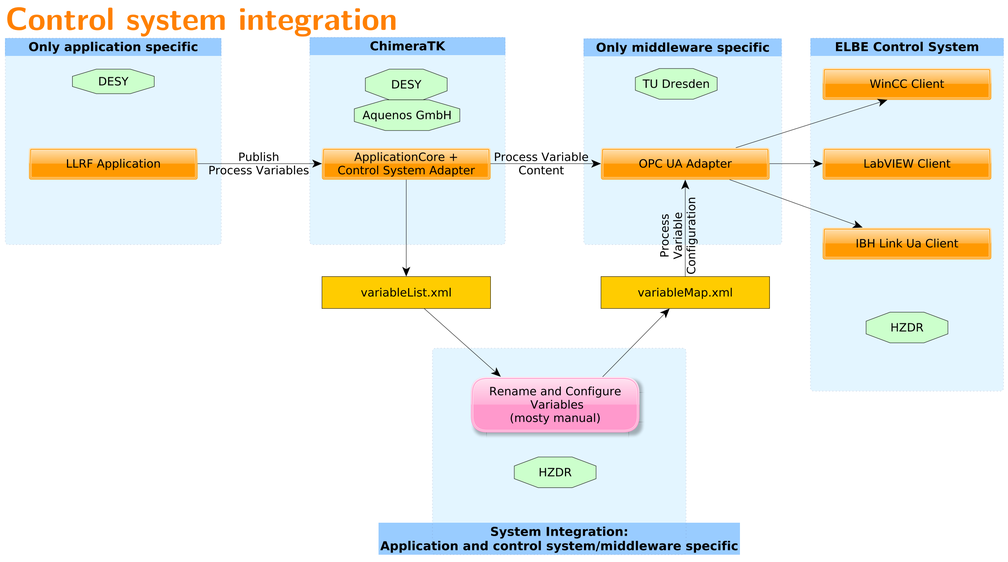
> Avoid locking problems with middleware

> Lock-free queues allow different read-modes
- *non-blocking read*: return last received value if queue is empty
- *read latest*: empty the queue and return last received value
- *blocking read*: wait for new data if queue is empty

> Basis for inter-thread communication, also in ApplicationCore



Control system integration



The ChimeraTK ControlSystemAdapter is complemented by a middleware-specific part (DOOCS Adapter, OPC UA Adapter, EPICS Adapter)

> Publish process variables via middleware

> Define variable name visible in control system

> Define middleware dependent features/data types (server-side histories, display properties)

> Application independent, configured via config file

Software repositories

> https://github.com/ChimeraTK/DeviceAccess

> https://github.com/ChimeraTK/ControlSystemAdapter

> https://github.com/ChimeraTK/ApplicationCore

> https://github.com/ChimeraTK/ControlSystemAdapter-DoocsAdapter

> https://github.com/ChimeraTK/ ControlSystemAdapter-OPC-UA-Adapter

> http://oss.aquenos.com/svnroot/epics-mtca4u

Presenter

Martin Killenberg
martin.killenberg@desy.de

ICALEPCS2017, Barcelona, Spain, 10th October 2017, PosterID TUPHA178