DESY is developing MTCA.4 boards for licensing by industry partners

- **Low latency 12.5 Gbit/s data processing unit**
- **8 channel 2.7 GHz ADC**
- **10 channel high frequency down converter**

DESY provides:
- Board support package (firmware development)
- Linux drivers
- C++ tools to facilitate software development
MTCA4U will provide

- Linux drivers
- C++ API

for all MTCA.4 boards developed at DESY.

- Board-specific classes for implementations used at DESY.

- Tools for easy integration into control systems.

Requirements

- Independent from the control system
- Well documented, intuitive API
- Base version open source (compile on many distributions)
- Board-specific classes can be closed source (protection of intellectual property)
- Universal and extendable
  - Avoid code duplication
  - Simplify development for new boards
Design Concept

- **Open Source**
  - Driver
  - Base API
  - Hardware Monitor
  - Control System Tools

- **Closed Source** (example)
  - Low Level Radio Frequency (LLRF) control library for the accelerator

- **Servers**
  - Control system dependent

---

**Board Support Package**

- **Firmware**
- **Register Map**
- **Driver**

**Mapping Library**

- **C++ Device API**
- **QT Hardware Monitor**
- **LLRF Library**
- **C++ LLRF API**

**Control System Tools**

- **EPICS**
  - LLRF EPICS Server
- **DOOCs**
  - LLRF DOOCs Server
- **TANGO**
  - LLRF TANGO Server
- **YOUR Control System**
  - YOUR LLRF Server

Martin Killenberg (DESY)
Basic firmware provides:
- PCIe endpoint with standard register set
- Support for DMA transfer
- Access to basic hardware features
- Mechanism for user projects in the FPGA

Register Mapping

Mapping file for the specific firmware:
- Automatically generated by the board support package
- Contains information about:
  - Register name
  - Address
  - Size
  - Data type

Advantages:
- Use descriptive names instead of hex-addresses
- Better code readability
- User code becomes independent from firmware version
- Automated type conversion
Basic functionality is the same for all boards

- Read and write access to registers on the FPGA
- DMA transfer of large memory areas (needs firmware support)
- Atomic read–modify–write of registers via ioctl

Generic base driver

- Implements basic functionality
- Only board-specific ioctlS have to be written
- Easy, short and straightforward implementation of dedicated drivers for individual boards

See talk by Ludwig Petrosyan (Session 8, 16:45h)
Higher Level C++ Software

MTCA4U is intended for applications running on front end CPUs in the MTCA.4 crate with direct hardware access.

Basic C++ API
- C++ classes for convenient read/write, incl. DMA (no need to bother with driver implementation details)
- Implements the register name mapping
- Hot-plug support (devices can become available/unavailable at run time)
- ...

Generic Tools
- GUI for direct hardware access in the setup/maintenance phase

Control System Tools
- Keep the application code control system independent
- Enable easy integration into various control systems
- Control system dependent software is a thin adapter layer
Modern, object oriented design
- Easy to use interfaces
- Multiple abstraction layers, adapted to the different use cases
  - Normal operation
  - Calibration/setup
  - Expert

Unit testing framework
- Well tested code
- Facilitates refactoring
- Dummy devices for software development without hardware access

Doxygen documentation
- Complete, browsable API documentation
Control System Tools

Two contradicting requirements

- Keep application code control system independent
- Do not reimplement functionality provided by the control system
  ⇒ Keep the layer as thin as possible

Process Variables

- MTCA4U is not event driven, but has to provide support for event driven control systems
- Each process variable has a
  - Set function
  - Get function
  - Callback function on change

State Machines

- Devices are usually implemented as state machines
- Some control systems require a device to be a state machine
  ⇒ MTCA4U will provide a state machine
- Callback functions on state change to synchronise with the control system
GUI for the basic API

- Display devices and registers by name
- Show and modify register content
- Basic plotting functionality
Status

Engineering versions for
- Board support package and firmware
- Generic modular driver
- I/O class
- Register mapping
  - Based on plain text files
- Hardware monitor GUI

All code needs
- Cleanup of the API
- Quality control (fully automated unit tests)
- Documentation

MTCA4U subversion repository on the DESY svn server:
https://svnsrv.desy.de/public/mtca4u/
Outlook and Plans

Next steps

- Clean up and improve basic API
- Complete the unit test suite
- Switch to XML mapping files
  - Support for data types and multiple projects
- Improve performance using generic DMA transfer (collaboration with Cosylab)
- Implement run time hot-plug support (collaboration with Cosylab)
- Design and implement Control System Tools
  - Process variables
  - State machine
- Python bindings

To be addressed

- Thread safety / concurrency
- Real time requirements