Nadeem Shehzad

M. Hierholzer, M. Heuer, L. Petrosyan, C. Schmidt, M. Killenberg, M. Vitti, G. Varghese

DESY, Hamburg, Germany

S. Marsching, aquenos GmbH, Baden-Baden, Germany

J. Krašna, M. Mehle, T. Sušnik, K. Žagar, Cosylab d.d., Ljubljana, Slovenia

A. Piotrowski, FastLogic Sp. z o.o., Łódź, Poland

T. Kozak, P. Prędki, J. Wychowaniak, Łódź University of Technology, Łódź, Poland

MTCA Workshop 10th December 2015, DESY
**MTCA4U — The DESY MicroTCA.4 User Tool Kit**

<table>
<thead>
<tr>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide a tool kit to facilitate the development for MicroTCA.4 based control applications.</td>
</tr>
</tbody>
</table>

MTCA4U comprises
- Linux drivers for PCIeexpress
- Intuitive C++ API
- Tools for easy integration into control systems
- Board-specific classes for implementations used at DESY

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent from the control system</td>
</tr>
<tr>
<td>Universal and extensible</td>
</tr>
<tr>
<td>Base version open source (compile on many distributions)</td>
</tr>
<tr>
<td>Board-specific classes can be closed source (protection of intellectual property)</td>
</tr>
</tbody>
</table>
Device Backends for Hardware Access

Features

- C++ API has an abstract DeviceBackend interface
  - Different implementation for specific hardware
- Easy extensibility by adding new Backends
- Device backends can be added at run time (factory with plug-in mechanism)

Rebot = Register-based over TCP
The Backend Factory

Features

- Abstract the creation of device backends.
- Devices are created using alias names.
- New backends can be added through plug-in mechanism.
- Factory remains unchanged.
- Factory uses the URI format based on the Standard Device Model suggested by Augustus P. Lowell.
  (sdm work still in progress in the PICMG SW working group)

URI Scheme To Describe Backend Types

- Device Name Mapping
- Common format for portability across different platforms
- Specify interfaces, instances and parameters to access the hardware
Backend Factory in Action

Application → Create 'SIS8300' → Backend Factory

- mtca4u::PcieBackend

Device Map:
- SIS8300 sd://./pci:pcieunis6
- TMCB sd://./rebot=192.168.1.20
- VirtualSIS8300 sd://./dummy=llrf.map
QtHardMon — A GUI for the Basic API

- Display devices and registers by name
- Show and modify register content
- Basic plotting functionality
Command Line Tools

Features

- Uses new Device Factory.
- Easy monitoring of device from command prompt.
- Fast application development through scripting.
- All mtca4u device functions including reading and writing to registers

```
mskpcx18356% mtca4u help

mtca4u command line tools, version 00.07.00

Available commands are:

help                  Prints the help text
version               Prints the tools version
info                  Prints all devices
device_info           Board Module
register_info         Board Module Register
register_size         Board Module Register
read                  Board Module Register [offset] [elements] [raw | hex]  Read data from Board
write                 Board Module Register Value [offset]  Write data to Board
```
Matlab Tools and Python Bindings

**Matlab**

```matlab
>> struck_adc = mtca4u('SIS8300');
>> struck_adc.read('BOARD','WORD_COMPILATION')
ans =
         9

>> struck_adc.write('ADC', 'WORD_ADC_ENA',1)
```

**Python**

```python
In [1]: import mtca4u
In [2]: struck_adc = mtca4u.Device("SIS8300")
In [3]: struck_adc.read('BOARD', 'WORD_COMPILATION')
Out[3]: array([ 9.], dtype=float32)
In [4]: struck_adc.write('ADC', 'WORD_ADC_ENA', 1)
In [5]:
```
Virtual Lab

Features

➢ Developed as part of MTCA4U package

➢ A virtual device can be implemented and inserted in place of the actual device.

➢ Automated test can be performed to ensure software quality.

➢ e.g, a simple cavity model can be connected for realistic tests.
Virtual Lab at Work

Virtual Device: SIS8300L

- Firmware State Machine
- Simulates control loop etc.
- Dummy Register Set

Signal Sink: I/Q

Timer

Signal Source: I/Q

Cavity Model

Calculations

Signal Source: I/Q

Timer

Signal Sink: I/Q

Test Routines: Simulate operator commands, check server and cavity behavior

Image Courtesy: Martin Hierholzer
## Goal

Make use of complex control algorithms with different control systems.

## Features

- Allows application code to access process variables independent of control system.
- Thread safe.
- Real Time compliant.
- Implementations for DOOCS and EPICS are ready to use.
- Planned: support for UPC-UA
Summary

MTCA4U

- Easy hardware access for MicroTCA.4
  - C++ API
  - GUI and bindings to scripting languages

- New BackendFactory
- Virtual Lab Framework
- Control System Adapters for DOOCS and EPICS are ready.

SVN repository:
https://svnsrv.desy.de/public/mtca4u