MTCA4U — The DESY MicroTCA.4 User Tool Kit.
Generic software and drivers for MicroTCA.4 based controls

Martin Killenberg

M. Heuer, L. Petrosyan, C. Schmidt, 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

11th December 2014
MTCA4U — The DESY MicroTCA.4 User Tool Kit

**Goal**

Provide a tool kit to facilitate the development for MicroTCA.4 based control applications.

MTCA4U comprises

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

**Requirements**

- Independent from the control system
- Universal and extensible
- Base version open source (compile on many distributions)
- Board-specific classes can be closed source (protection of intellectual property)
Design Concept

Board Support Package

Firmware

Register Map

Linux Driver

/dev/mtca_slot0

Mapping Library

C++ Device API

Qt Hardware Monitor

LLRF Library

C++ LLRF Application

Control System Adapter

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

Python Bindings

Matlab Bindings

Command Line Tools

EPICS

LLRF EPICS Server

DOOCS

LLRF DOOCS Server

TANGO

LLRF TANGO Server

YOUR Control System

YOUR LLRF Server

Martin Killenberg (DESY)
Firmware and PCIeexpress Driver

Firmware

- Standard Register Set at DESY
  ⇒ All boards can use the same driver
- PICMG is working on Standard Hardware API
  (see talk by Till Straumann, Wednesday 11:15 h)
- Firmware board support package automatically generates a register map

Goal

Make all firmware compliant to the PICMG Standard Hardware API.

PCleexpress driver

- Common driver for all boards
  (see talk by Ludwig Petrosyan, Thursday 9:30 h)
- The drivers are part of MTCA4U
The C++ Interface

Basic C++ API

- Classes for convenient read/write via PCIexpress
- Interface for Direct Memory Access (no need to bother with driver implementation details)
- Register name mapping

Goal

Make MTCA4U a reference implementation of the PICMG Standard Device Model (see talk by Till Straumann, Wednesday 11:15 h).

Register Name Mapping

Map is automatically generated by the firmware board support package

- 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
Display devices and registers by name
Show and modify register content
Basic plotting functionality
Control System Adapter

Task
Complex control algorithms should be used with different control systems.

Contradicting requirements
- Keep application code control system independent
- The algorithm must interact with the control system
- Do not re-implement functionality provided by the control system
⇒ Keep the layer as thin as possible

Additional requirements:
- Thread-safety
- Real-time capability
- Must not copy large data objects (arrays)

Control System Adapter
- Process variables to transfer data to/from the control system
- Callback mechanism to perform actions
Wrapper classes for
- simple data types
- arrays of simple data types
- Contains instance of control system variable

Only basic interface
- Get function
- Set function
- Callback function on change
- No control system functionality!

Adapters
- DOOCS (in preparation)
- EPICS (in preparation)
- OPC-UA (planned)
New Tools

Command line tools
- Query devices (list registers)
- Read/write incl. register mapping
- First version is released

Matlab bindings
- Directly use MicroTCA.4 devices inside of Matlab
- Uses the C++ library when running on the front end CPU
- Can tunnel to a remote host via ssh, using the command line tools

Python bindings
- Use the C++ library from python
- Work has just started
Test suite

- Unit tests with very high code coverage (99 %)
- Dummy driver to test the I/O classes
  - Simulates PCIe registers in the Linux kernel memory
- Dummy device for writing mock classes
  - Loads the mapping file
  - Simulates all registers in user space memory
  - Register callback functions to inject functionality
- Planned: Reference firmware to unit-test the driver

Continuous integration tests

- Check out every subversion commit
- Compile, install and run tests
- Send email in case of errors
Project mtca4u_MtcaMappedDevice

Project disk usage information + trend graph

- Disk Usage: Workspace 309 MB (On slaves 309 MB, Non slave workspaces -1), Builds 58 MB (Locked -1), Job directory 61 MB

Code Coverage

- Packages: 100%
- Files: 98%
- Classes: 98%
- Methods: 100%
- Lines: 99%
- Conditionals: 44%

Cpptest Check Trend
Status and Outlook

Engineering versions for

- Board support package and firmware
- Universal PCleexpress driver
- C++ I/O class
- Register name mapping and type conversion
- Hardware monitor GUI
- Command line tools and Matlab bindings
- Quality control (unit tests and continuous integration)

Under development

- Control system adapter
- Python bindings

MTCA4U is published under the GNU General Public License.

Subversion repository on the DESY svn server:
https://svnsrv.desy.de/public/mtca4u/
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
- Code coverage

Doxygen documentation

- Complete, browsable API documentation
Three types of actions in the control system adapter

1. **Synchronous actions on set/get**
   - Only one process variable is updated
   - Callback function registered with each process variable

2. **Update functions triggered by the control system**
   - Periodic updates
   - Control system triggers
   - All process variables are updated
   - Callback function registered with the control system adapter

3. **Synchronisation triggered by the business logic**
   - Control loop running in its own thread
   - Business logic determines when to synchronise
   - All process variables are updated
   - Callback function registered with the control system adapter

- The control system adapter assures thread safety for all process variables inside the callback functions.
- All callback functions have a control system independent signature.
**mtca4u::MotorDriverCard Class Reference**

A class to access the DFMC-MD22 motor driver card, which provides two MotorControllers. More...

```cpp
#include <MotorDriverCard.h>
```

Inheritance diagram for mtca4u::MotorDriverCard:

```
mtca4u::MotorDriverCard

mtca4u::MotorDriverCardExpert

mtca4u::MotorDriverCardImpl
```

List of all members.

**Public Member Functions**

```cpp
virtual unsigned int getControllerChipVersion ()=0
```

Get access to one of the two motor controllers on this board.

```cpp
virtual MotorController & getMotorController (unsigned int motorControllerID)=0
```

```cpp
virtual PowerMonitor & getPowerMonitor ()=0
```

Get a reference to the power monitor.

```cpp
virtual ReferenceSwitchData getReferenceSwitchData ()=0
```

**Detailed Description**
Unit Tests

- Tests written using the boost::test library
- Fully integrated into the CMake build system
  - Automatically run when packaging, e.g.
- Used to create code coverage report
  - Goal: Test every single line of code
# LCOV - code coverage report

**Current view:** top level - /home/killenb/MotorDriverCard_trunk/src

**Test:** coverage.info  
**Date:** 2014-03-22  
**Lines:** 555 / 564  
**Functions:** 150 / 151  
**Coverage:** 98.4%  

<table>
<thead>
<tr>
<th>Filename</th>
<th>Line Coverage</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFMC_MD22Dummy.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MotorController.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MotorControllerConfig.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MotorDriverCardConfig.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MotorDriverCardConfigXML.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>MotorDriverCardImpl.cc</td>
<td>92.1%</td>
<td>96.7%</td>
</tr>
<tr>
<td>MultiVariableWord.cc</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Generated by: LCOV version 1.10