MTCA4U — The DESY MicroTCA.4 User Tool Kit.

Update and similarities to the PICMC Standard Device Model

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*

HELMHOLTZ ASSOCIATION

24th February 2015

µTCA®

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

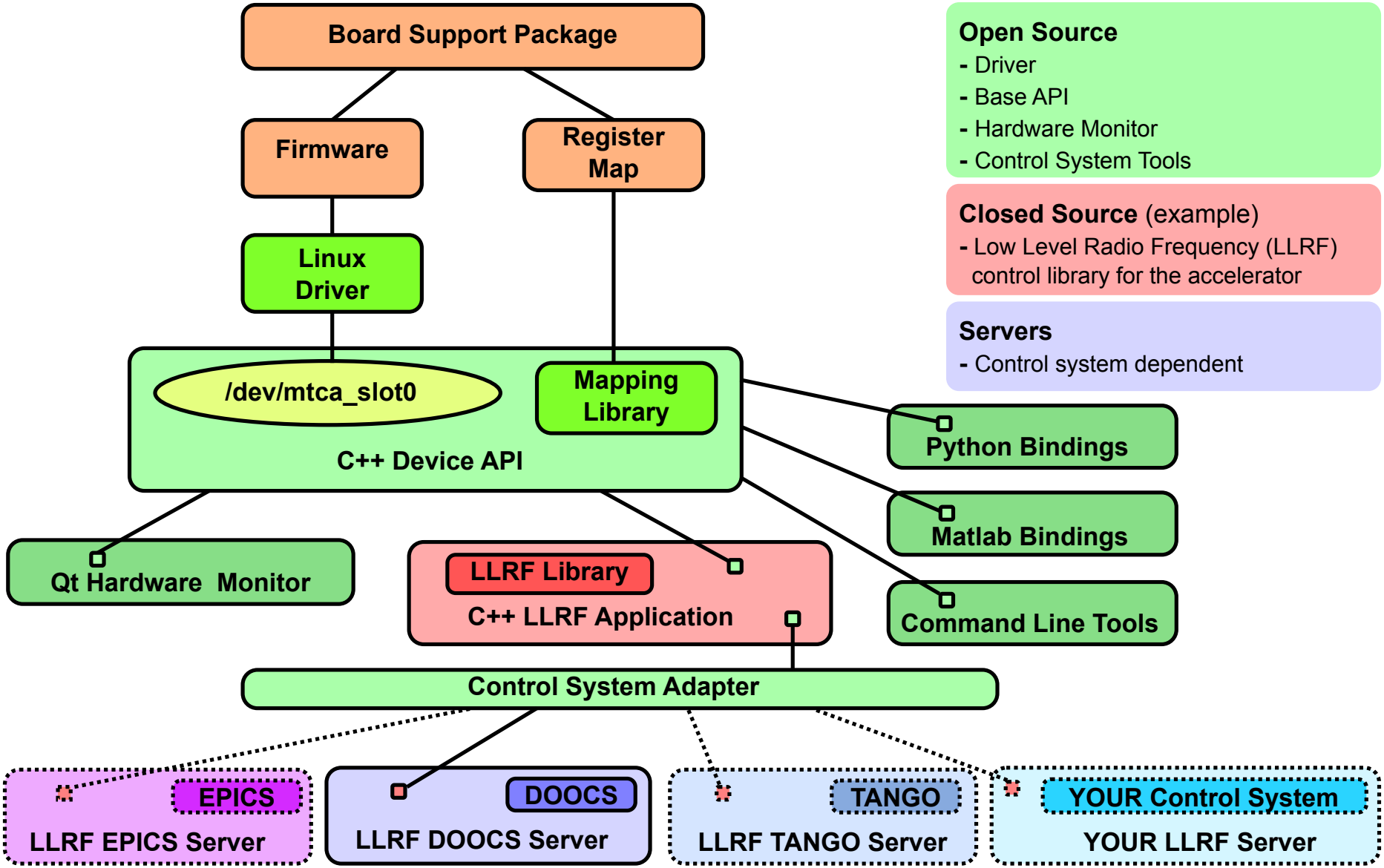
- Linux drivers for PCIexpress
- Intuitive C++ API
- Tools for easy integration into control systems

- Board-specific classes for implementations used at DESY

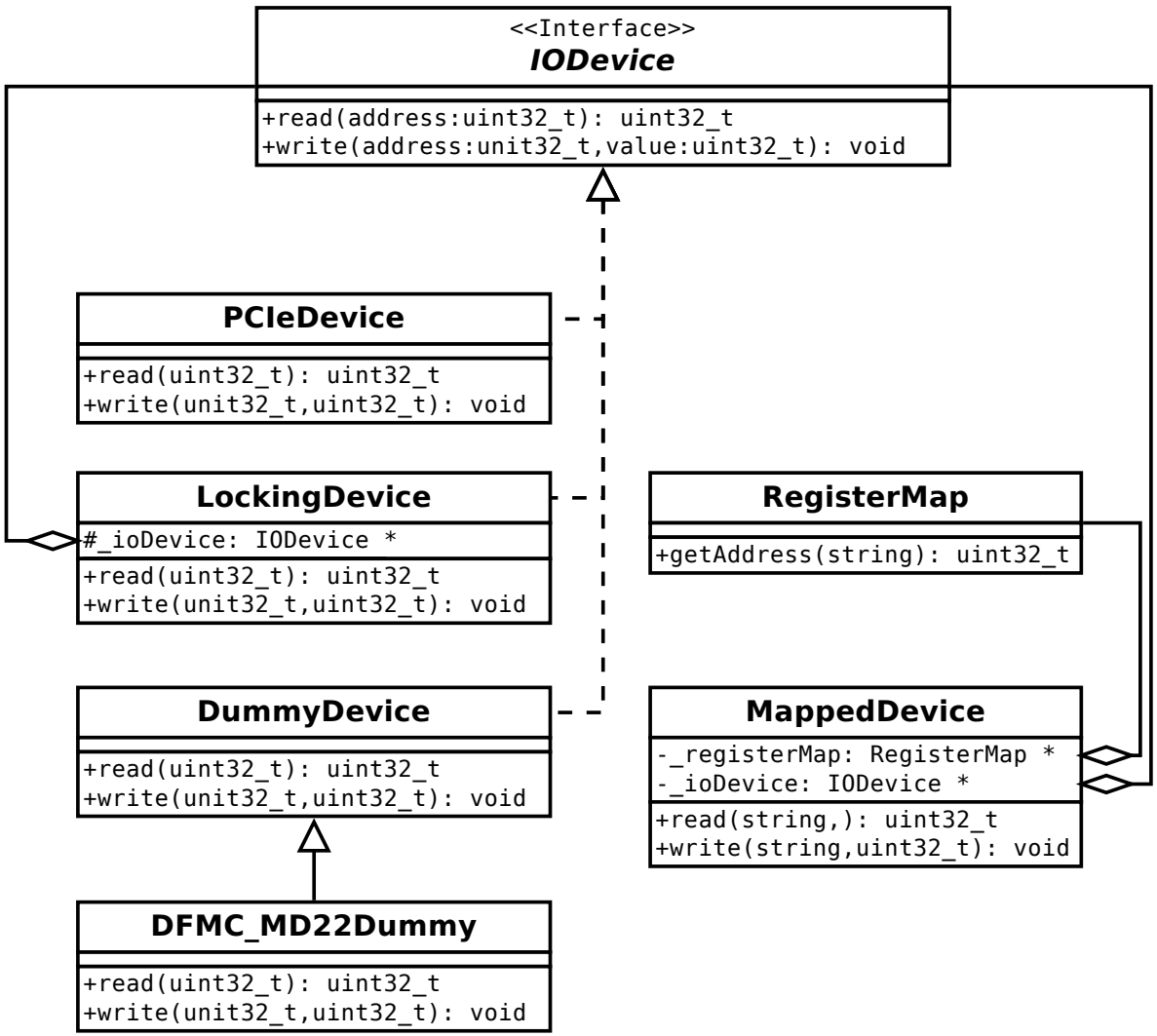
Requirements

- 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



C++ Device API



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

The C++ Interface

Basic C++ API

- Classes for convenient read/write through a common interface (address based)
- Different Base Address Ranges (there are 6 PCIexpress BARs)
- Interface for Direct Memory Access (no need to bother with driver implementation details)
- Register name mapping

Three implementations

- PCIexpress
- Memory (dummy device)
- File

Towards the Standard Device Model

+ Abstract interface with multiple implementations
+ Extensible and stackable
– Only address based, no stream I/O
– No support for sub-devices (can use more BARs)

Device Name Mapping

Device map file (dmap file)

Text file describes which boards are installed in the crate

- Alias (functional name to be used in the application)
- Device file node
- Mapping file (register name mapping)

Towards the Standard Device Model

+ Open a device by functional name
– No URI syntax
– Only works with PCIe (and partly with dummies)
– Only works per computer, no database mechanism

Register Name Mapping

Mapping file for specific firmware

- Automatically generated by the firmware board support package
- Contains information about
  - Register name
  - Address
  - Size
  - Data type

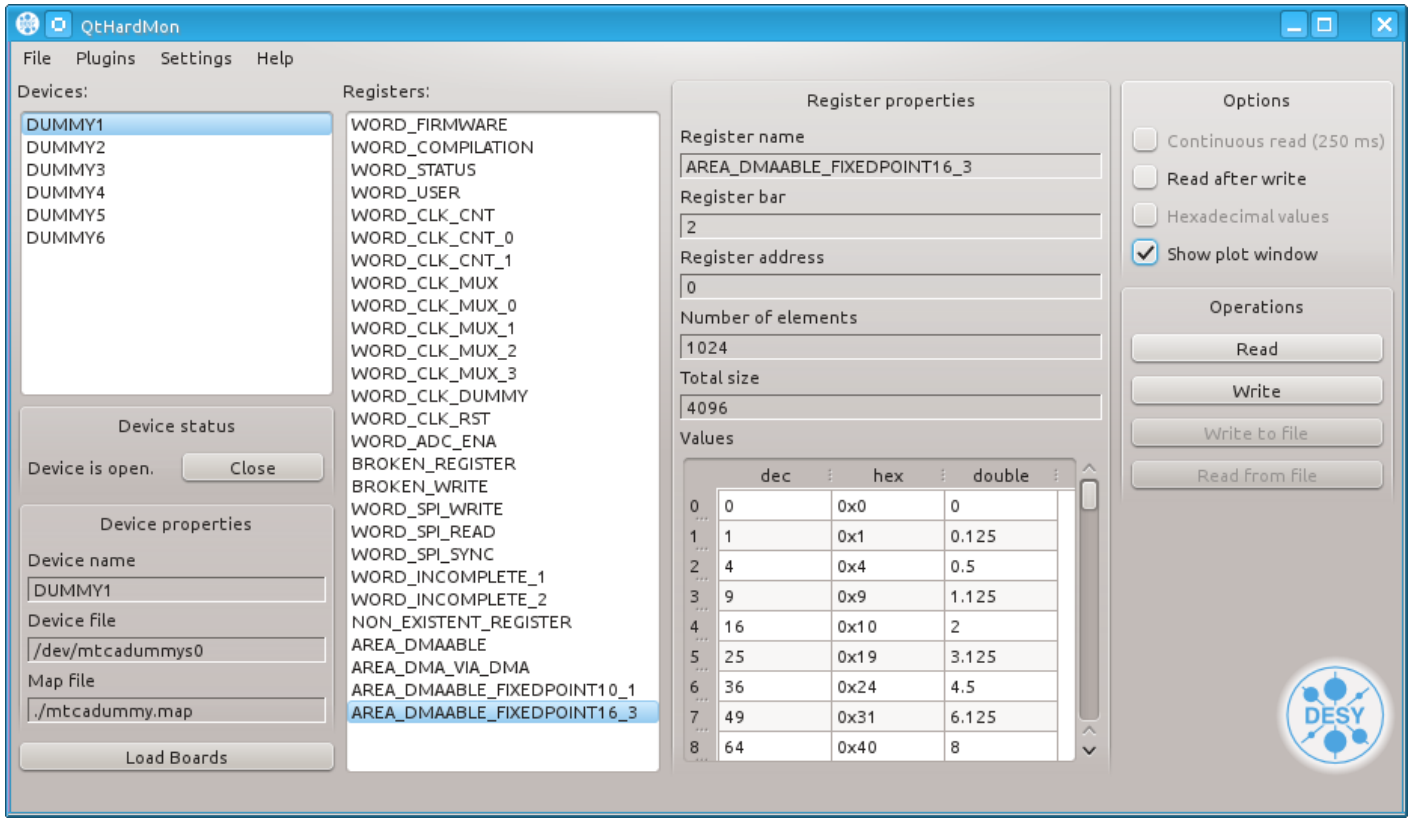
Advantages:

- Use descriptive names instead of hex-addresses
- Better code readability
- User code becomes independent from firmware version
- Automated type conversion

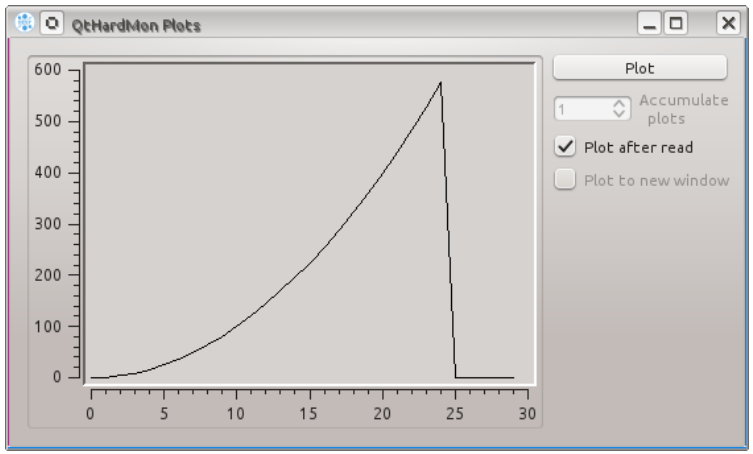
Towards the Standard Device Model

- Not foreseen in the Standard Device Model yet

QtHardMon — A GUI for the Basic API



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



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

Status and Outlook

C++ Device API

- Abstract API for address based I/O
- Device name mapping
- Register name mapping
- Language bindings for Matlab and Python
- Command line tools
- Hardware monitor GUI

– No stream I/O
– No support for sub-devices

Goal

- Turn MTCA4U into a PICMG Standard Device Model reference implementation

  or

- Write an indepedent reference implementation and use it in MTCA4U

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

Backup

Doxygen Documentation

DFMC-MD22 Motor Controller: mtca4u::MotorDriverCard Class Reference – Konqueror

File Edit View Go Bookmarks Settings Window Help

/space/killenb/MotorDriverCard_trunk/doc/html/classmtca4u_1_1_motor_driver_card.html

mtca4u | MotorDriverCard

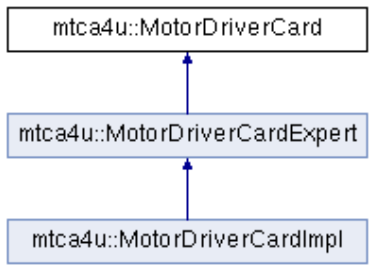
Public Member Functions

mtca4u::MotorDriverCard Class Reference

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

#include <MotorDriverCard.h>

Inheritance diagram for mtca4u::MotorDriverCard:



List of all members.

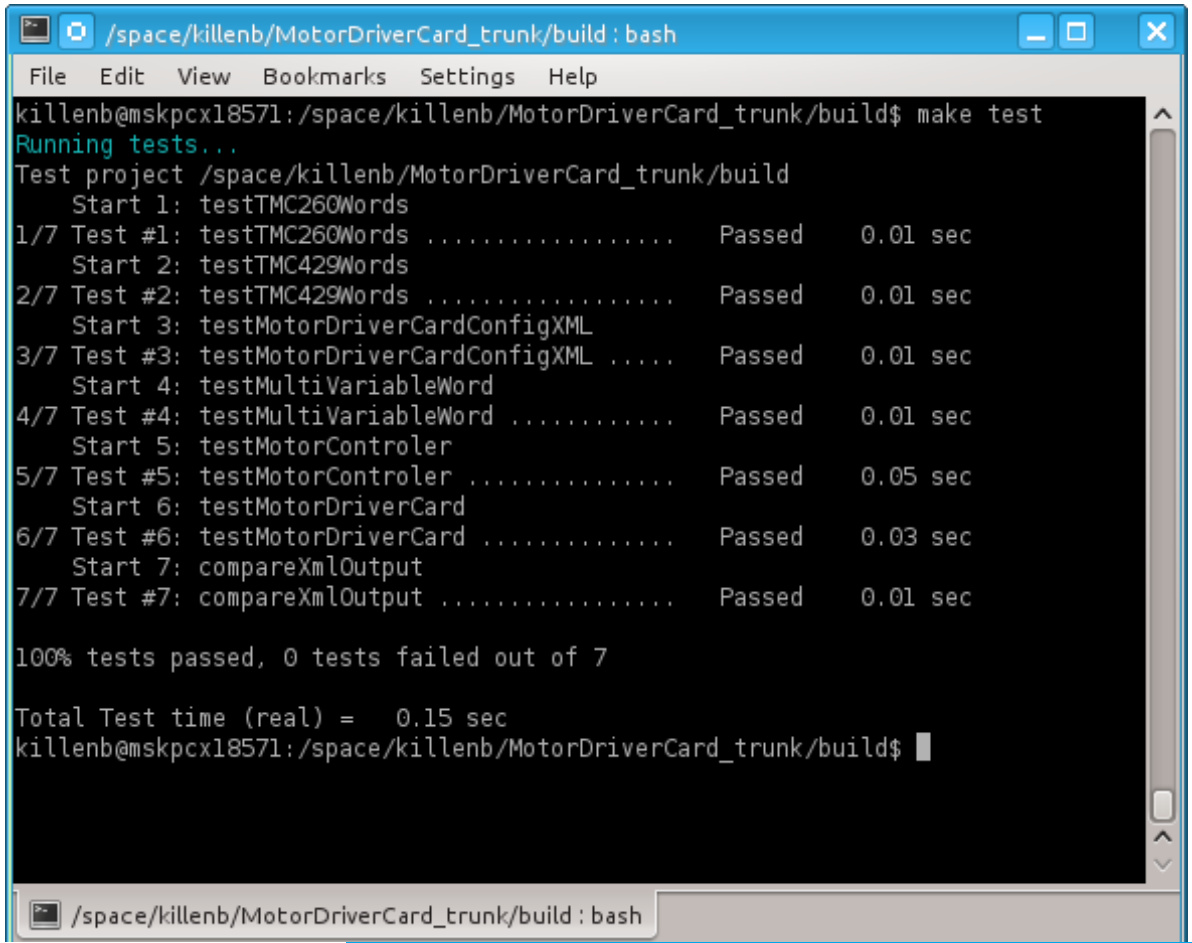
Public Member Functions

virtual unsigned int	**getControlerChipVersion** ()=0
virtual **MotorControler** &	**getMotorControler** (unsigned int motorControlerID)=0 Get acces to one of the two motor controlers on this board.
virtual **PowerMonitor** &	**getPowerMonitor** ()=0 Get a reference to the power monitor.
virtual **ReferenceSwitchData**	**getReferenceSwitchData** ()=0

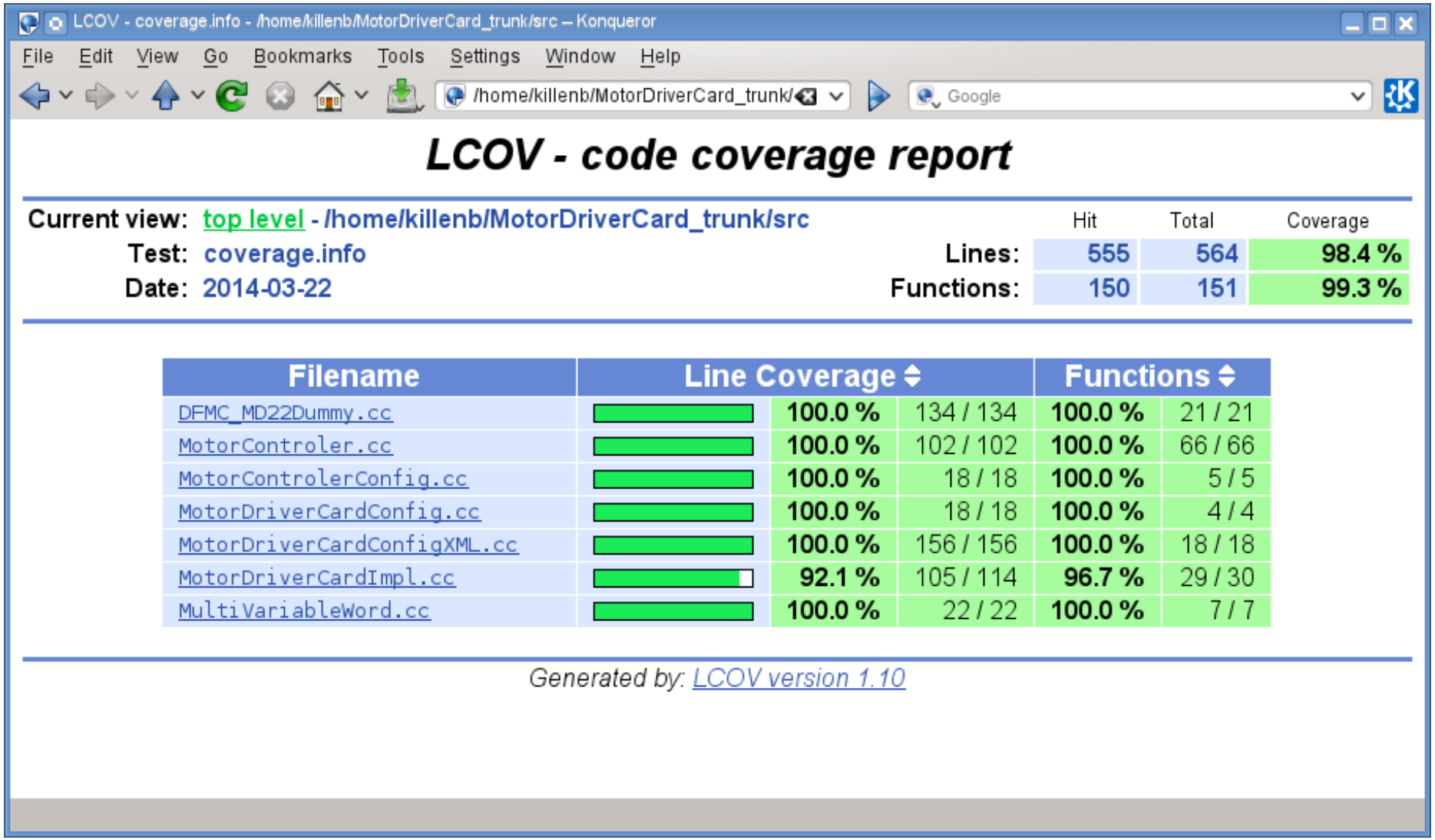
Detailed Description

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



Code Coverage Report



Quality Assurance

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 devices 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

- Checkout every subversion commit
- Compile, install and run tests
- Send email in case of errors

Jenkins Continuous Integration Server