MTCA4U — The DESY MTCA.4 User Tool Kit
A generic interface for MTCA software development.

Martin Killenberg

Follow-up Meeting on MTCA.4 Board Development
25. June 2013

The Task

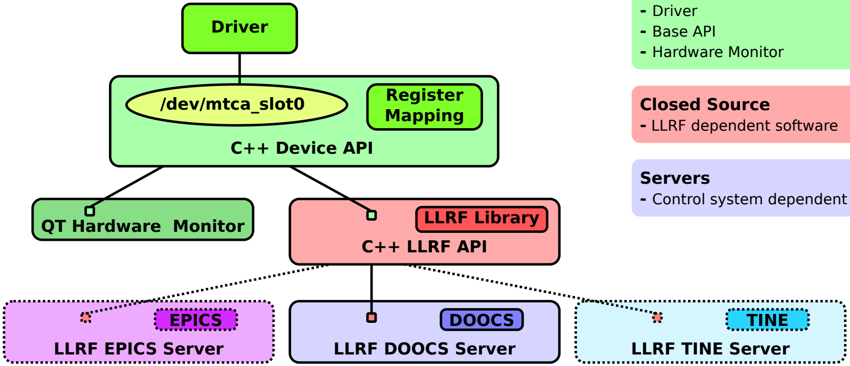
In the context of the Helmholtz Validation Fund:
Develop drivers and a software package to be delivered with every MTCA.4 board.

- Base version which only ships with basic tools
- LLRF version with full FPGA firmware installed. Ships with LLRF tools and server.

Requirements

- Independent from DOOCS
- Well documented, intuitive C++ API
- Base version open source (compile on many distributions)
- Universal and extendable
  - Avoid code duplication
  - Simplify development for new boards

Current Design Idea



Open Source
- Driver
- Base API
- Hardware Monitor

Closed Source
- LLRF dependent software

Servers
- Control system dependent

The Driver

Requirements

- Universal version for basic read and write
- DMA transfer of large memory areas (needs firmware support)
- Extendable for special features of individual boards (ioclt)

Two versions available (Adam Piotrowski, Ludwig Petrosyan).

Ludwig's version features a base driver with functions exported to kernel space:

- High reusability of code
- Easy, short and straight forward implementation of dedicated drivers for individual boards

My ideas:

- Read/write is done using a struct (to provide atomic operation)
  Can this be changed to normal register addressing?
- Special treatment of DMA transfer
  Should be transparent for users: DMA address space is automatically transferred via DMA with the same read call as normal registers.

Needs further discussion!

The Basic API / Device API

Modular C++ API by Adam Piotrowski

- Class for convenient read/write, incl. DMA
- Register name mapping, loadable from config file
- List of boards read from config file
- ...

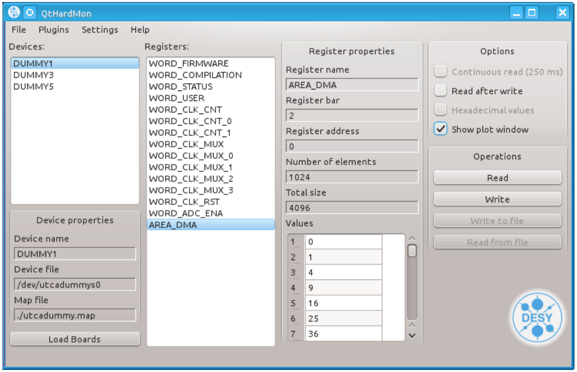
Improvements

- Consistent and intuitive class and method names
- Couple mapping file and firmware (version number, check sum)
- List of boards automatic at run time (hot-plug)

Needs further discussion!

QtHardMon

GUI for the basic API


QtHardMon

File Plugins Settings Help

Devices:
- DUMMY1
- DUMMY3
- DUMMY5

Registers:
- WORD_FIRMWARE
- WORD_COMPILATION
- WORD_STATUS
- WORD_USER
- WORD_CLK_CNT
- WORD_CLK_CNT_0
- WORD_CLK_CNT_1
- WORD_CLK_MUX
- WORD_CLK_MUX_0
- WORD_CLK_MUX_1
- WORD_CLK_MUX_2
- WORD_CLK_MUX_3
- WORD_CLK_RST
- WORD_ADC_ENA
- AREA_DMA

Device properties

Device name: DUMMY1
Device file: /dev/utcadummys0
Map file: ./utcadummy.map

Load Boards

Register properties

Register name: AREA_DMA
Register bar: 2
Register address: 0
Number of elements: 1024
Total size: 4096

Values

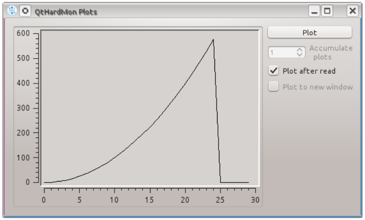
1	0
2	1
3	4
4	9
5	16
6	25
7	36

Options
- [ ] Continuous read (250 ms)
- [ ] Read after write
- [ ] Hexadecimal values
- [✓] Show plot window

Operations

Read | Write | Write to file | Read from file

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


QtHardMon Plots

Plot

Accumulate plots: 1
- [✓] Plot after read
- [ ] Plot to new window

Hardware monitor was Matlab-based

+ Directly export data to Matlab
– Matlab license required to run it

New version based on Qt

+ Standard open source software available on any Linux
– No scripting functionality like Matlab

The Advanced API / LLRF API

- Interface to the LLRF library
- Provides as much server functionality as possible, without duplicating functionality from DOOCS
- How to interface to different control systems (DOOCS, EPICS, TINE)?

The Challenge

- Decouple LLRF steering functionality from DOOCS
- Define a good interface

I have to study the server code.
Needs further discussion!

Build System and Repository

Build system: CMake

- Powerful tool to configure and create Makefile
- Used by many projects (MySQL, KDE, Qt5, ROOT (CERN), iLCSoft (DESY), . . .)
- Platform independent
- Easier to maintain than manually written Makefiles
- Knows package dependencies, can install required software

Interface to DOOCS build system: Pre-installed Ubuntu packages as dependency.

Repository: Subversion

- We want to decouple from DOOCS anyway ⇒ Use different repository
- Subversion is much more powerful than CVS
- Use the server hosted by DESY: svnsrv.desy.de
  - Authentication with DESY account or simple password
  - Public read access
  - Full backup and maintenence

Status and Outlook

Current Status

- CMake-based stand alone build system
- Hardware monitor implemented in Qt, using the original basic API
- New: Subversion repository on the DESY svn server: `https://svnsrv.desy.de/public/mtca4u/`

Next Steps

- Improve basic API
  - Consistent, intuitive class and function names
  - Prepare hot-plug capability
- Implement hot-plugging
  - Check driver (should work, but does not)
  - Make C++ classes hot-plug capable
- Change QtHardMon and LLRF server to use new basic API
- Separate LLRF server into LLRF functionality (plain C++) and DOOCS server part
- Guinea pig project: Single cavity control