Modular Software for MicroTCA.4 Based Control Applications

N. Shehzad, M. Killenberg, M. Heuer, M. Hierholzer, L. Petrosyan, C. Schmidt, T. Kozak, G. Varghese, M. Viti, S. Marsching, M. Mehle, T. Sušnik, K. Žagar, A. Piotrowski, P. Prędki, J. Wychowaniak, K. Czuba, A. Dworzanski

***Abstract*—The MicroTCA.4 crate standard provides a powerful electronic platform for digital and analogue signal processing. Besides excellent hardware modularity, it is the software reliability and flexibility as well as the easy integration into existing software infrastructures that will drive the widespread adoption of the standard. The DESY MicroTCA.4 User Tool Kit (MTCA4U) is a collection of C++ libraries which facilitate the development of control applications. The device access library allows convenient access to hardware with an extensible register based interface. Starting from PCI Express, which is used inside a MicroTCA.4 crate, the introduction of new, network based protocols extends its reach beyond a single crate and even MircoTCA itself. Features like register name mapping and automatic type conversion provide a level of abstraction which makes the software robust against firmware and even hardware changes. Bindings to widely used scripting tools like Matlab and Python as well as a graphical user interface complete the portfolio needed for fast prototyping and firmware development. We give an update on the project status and present new features which have recently been introduced or are currently being implemented.**

I. Introduction

MICROTCA is a modular open source standard for switched fabric computer systems. The MicroTCA standards comprises basic system specifications including backplane, cooling, power management and different Advanced Mezzanine Card(AMC) to build a modular system. A MicroTCA.4 crate provides high-speed synchronous data acquisition in time critical systems and helps creating a robust and fail safe system by allowing hot swapping of the modules. [1][2]

II. Desy MicroTCA.4 User Tool Kit - MTCA4U

The Desy MicroTCA.4 User Tool Kit (MTCA4U) comes with a range of tools to provide easy access to the MircoTCA.4 devices with minimal effort. It contains a C++ library which can be used to create or extend the software to access such devices. The tool kit provides a testing framework for the software. This makes it easy to design and test a piece of software without the hardware. It also provides an interface between the control systems and the middle layer software to maximize the use of coherent software. The tool kit architecture is shown in Fig.1.

Manuscript received June 16th, 2016. This work was supported by the Helmholtz Validation Fund HVF-0016 MTCA.4 for Industry.

N. Shehzad, M Killenberg, M. Heuer, M. Hierholzer, T. Kozak, L. Petrosyan, C. Schmidt, G. Varghese, M. Viti are with Deutsche Elektronen-Synchrotron, Hamburg, 22607 Germany.

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

M. Mehle, T. Sušnik, K. Žagar are with Cosylab d.d., Ljubljana, Slovenia.

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

P. Prědki, J. Wychowaniak are with Łódź University of Technology, Łódź, Poland.

K. Czuba and A. Dworzanski are with the Warsaw University of Technology, Warsaw, Poland

III. Linux PCI Express Driver

In MicroTCA.4, devices usually connect to a CPU in the crate via PCI Express. The MTCA4U tool kit comes with an open source Linux kernel driver which provides an interface to the PCI Express bus. We use the Linux Device Driver Model to split and modularized the driver into two layers. The first layer is device independent and implements the PCI Express I/O address space. The second layer consists of device dependent features. This allows the first layer to be universal and reusable as this component is common to all PCI Express devices. Other device features like Direct Memory Access (DMA) are separated and stacked into the second layer. The devices developed at DESY use a standard register set and DMA mechanism provided by the firmware. This way a common driver can be used for most of the DESY devices. For devices coming from other vendors, the off the shelf universal driver provides quick access to all the basic features. Only device specific issues needed to be addressed. This way new devices could be added to the driver family without changing the interface. MTCA4U also supports Hot Plug on MicroTCA.4 crates as provided by the Linux kernel.

IV. The C++ Device API

The core function of the C++ API is to provide an easy interface to access the device without any hardware knowledge or the implementation details of the device specific input/output operations. The C++ library comes with a class called `Device`, which can be used to create an easy and quick access to the device's address space. The library provides different types of back-ends. For example the `PcieBackend` is used to access the aforementioned MicroTCA.4 boards which are connected via PCI Express or special back-end called `DummyBackend` can be used to simulate a device and its firmware functionality which is very useful for testing. The C++ API has been extended to access devices other than MicroTCA.4 based boards. The `RebotBackend` uses the Register-based over TCP (ReboT) protocol to access the devices via Ethernet and a `DoocsBackend` can

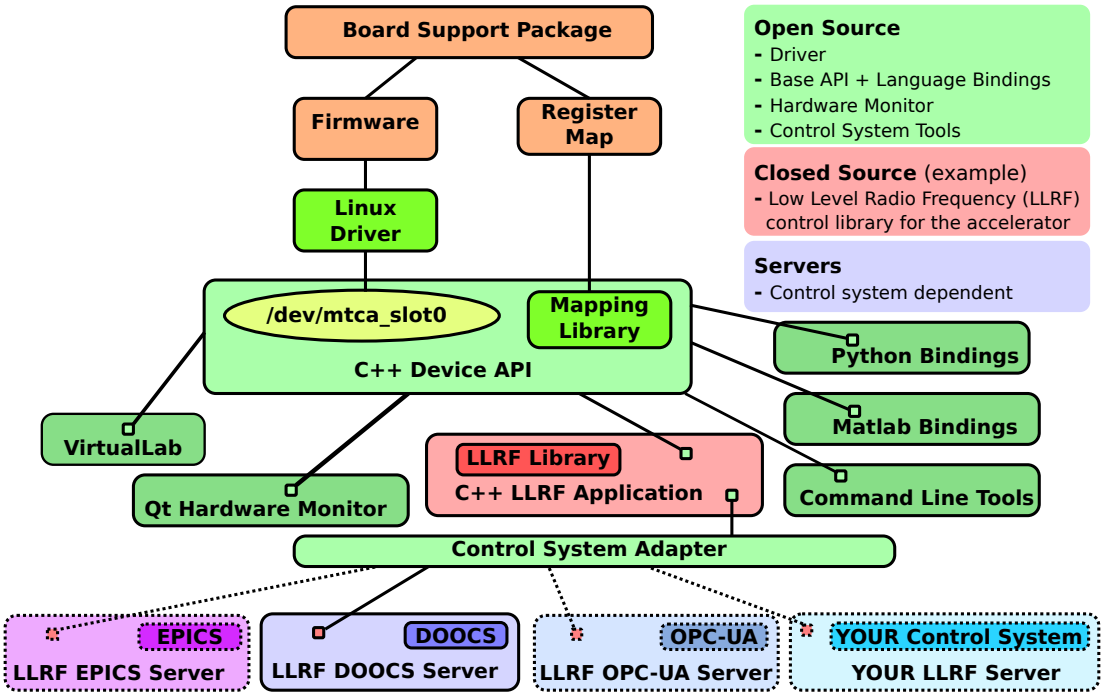

Fig. 1. The design concept of the MicroTCA.4 User Tool Kit MTCA4U.

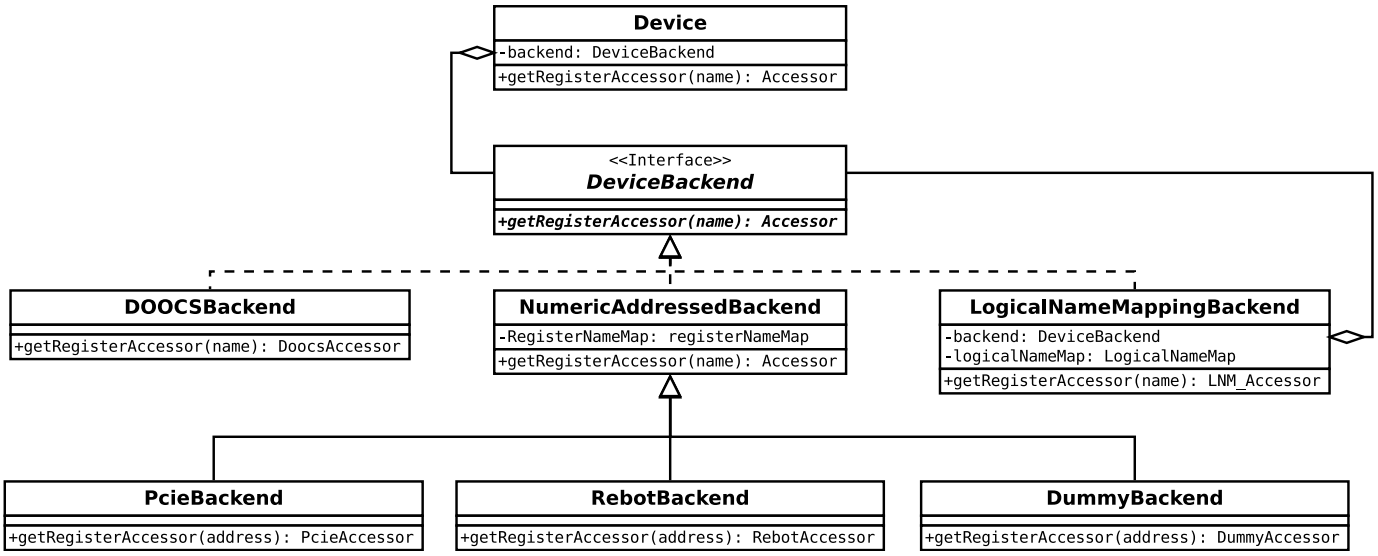

Fig. 2. High level architecture of MTCA4U devicessaccess library. Device Class interfacing with different back-ends through DeviceBackend Class.

be used to access control system middleware servers. A `LogicalNameMappingBackend` that provides mapping of logical register names onto real hardware registers is also available. This evolution of the library to use devices beyond MicroTCA.4 was the motivation to change of tool kit's name from MTCA4U to ChimeraTK[Control system and Hardware Interface with Mapped and Extensible Register-based device Abstraction Tool Kit]. The library can be easily extended by adding a new back-end.

An important feature of the library is the register name mapping of the `NumericAddressedBackend`. The I/O address space can change over time due to firmware updates. To make the software independent from these changes, an address mapping is used where the registers can be accessed by name rather than the address. This abstracts the register addressing scheme and enable users to write clean and robust code. The mapping file is created automatically with the board support package and is shipped together with the firmware. The table look up is minimized by creating accessor objects to cache the register addresses and enhance the overall access performance. The register name mapping is an important abstraction step because some back-ends (like the `DoocsBackend`) do not have numeric addresses.[3]

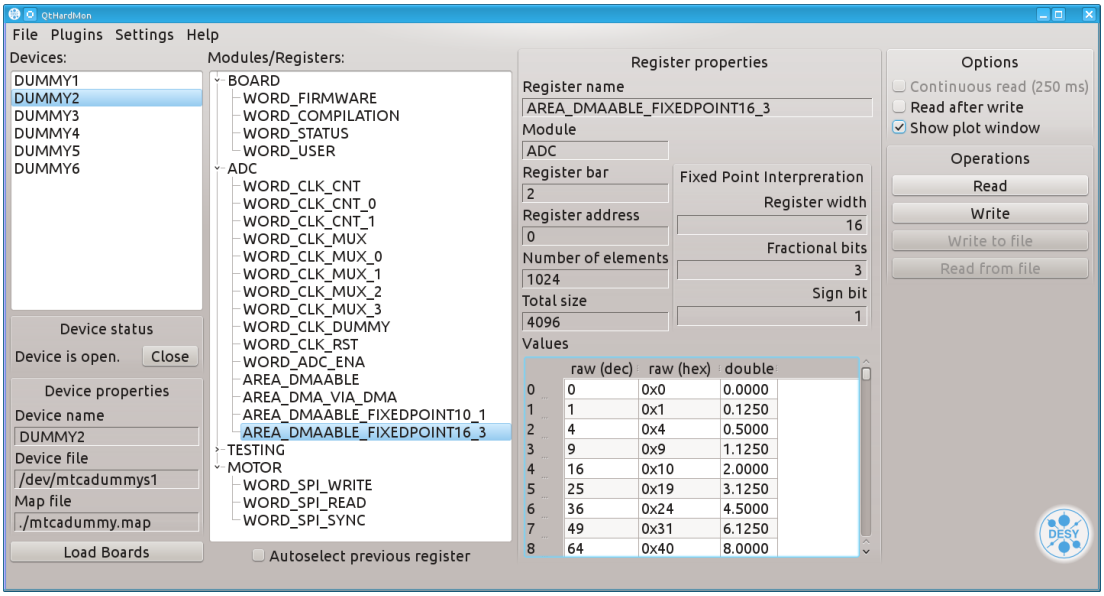

Fig. 3. The Qt Hardware Monitor makes the functionality of the device access library available in a graphical user interface.

V. The Back-end Factory

The C++ API provides an easy way to add new back-ends using a plug-in mechanism. It allows the new back-ends to be added at link time or at run-time. This way external plug-ins can be added whenever they are required, without needing to compile them together with the core library. Work is being done to provide a configuration file where shared objects can be defined, they are loaded at the runtime without having to recompile the existing binaries.

VI. Graphical User Interface

The API comes equipped with a graphical user interface tool called QtHardMon (Qt Hardware Monitor). Each back-end provides a list of available registers. For numerical addressed back-ends the mapping file which is shipped together with the firmware contains this information. Using QtHardMon, registers and their properties can be listed and the user can read from or write to the registers and modify their contents. This tool is typically used to debug and prototype by accessing the devices without having to write any code.

VII. Language Bindings

For quick and easy access to devices MTCA4U includes Python and Matlab binding as well as a command line tool. These bindings allows easy monitoring, configuration, testing and evaluation of the devices. It also provides a starting point to create special tool-sets, such as automated scripts.

VIII. Control System Adapter

Creating control algorithms is a complex and a tedious job which requires lots of time and energy. Different control system middle-wares provide different sets of features. Most of the time an application is strongly coupled to a control system. In order to re-use existing complex algorithms MTCA4U provides an adapter layer to write the control applications. When an application is written against the adapter rather then the concrete control system middle-ware it can be easily integrated into different control systems. The decoupling also removes the dependency on control system locks, which facilitates the implementation of real-time capable controls. A more detailed description of the control system adapter can be found in [5].

IX. Virtual Lab Framework

In addition to the device access library and the control system adapter, the C++ API provides a framework called 'VirtualLab' to test the software and help creating software simulations. The main focus is to understand the behavior of different components of the software when combined together. The VirtualLab provides signal sources and sinks to connect different components and a virtual timer to avoid situations like race conditions in multi-threaded applications. This framework ensures quality control for the software developers working with ChimeraTK and helps writing reliable and robust software. A more detailed description of the ChimeraTK virtual lab can be found in [6].

X. Conclusion

The DESY MicroTCA.4 User Tool Kit MTCA4U recently renamed to ChimeraTK, is an open source library published under the GNU Lesser General Public License and GNU General Public License. A robust C++ API provides a convenient way to write new software for hardware access. It has become flexible and universal and is not limited to MircoTCA.4 anymore. It comes with many different built-in back-ends to access PCI Express and ReboT devices or control system middle-ware. A plug-in mechanism allows to add new back-ends in the library itself or at run time. Rich features like register name mapping makes the life easier for new device back-end implementations by allowing it to re-use the API functionality. The library also includes a set of tools designed

to access the register based devices without having to write much code and comes with language bindings for Python and Matlab. A command line based tool as well as graphical user interface tool are included which allow quick prototyping and testing. The control system adapter provides a way to write control applications that could be used across all kinds of control systems. This allows a wider field of application for software written using MTCA4U/ChimeraTK. A testing framework VirtualLab comes with the tool kit. This ensures software quality, reliable software development and testing.

REFERENCES

[1] PICMG, *Telecommunications Computing Architecture, MicroTCA.0*, R1.0, 2006.

[2] PICMG, *Enhancements for Rear I/O and Precision Timing, MicroTCA.4 R1.0*, 2011/2012.

[3] DESY, *ChimeraTK - The DESY MicroTCA.4 User Tool Kit, Git Repository* Available: https://github.com/ChimeraTK

[4] Qt, *The Qt Project,* Available http://qt-project.org/

[5] M. Killenberg et al., *Integrating control applications into different control systems*, These Proceedings, 20th Real Time Conference, Padua Italy. 2016

[6] M. Hierholzer et al., *Software tests and simulations for realtime applications based on virtual time*, These Proceedings, 20th Real Time Conference, Padua Italy. 2016