ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO104

DRIVERS AND SOFTWARE FOR MicroTCA.4*

Martin Killenberg†, Lyudvig Petrosyan, Christian Schmidt, Deutsches Elektronen-Synchrotron DESY, 22607 Hamburg, Germany
Sebastian Marsching, aquenos GmbH, 76532 Baden-Baden, Germany
Adam Piotrowski, FastLogic Sp. z o.o., 90-441 Łódź, Poland

Abstract

The MicroTCA.4 crate standard is a powerful electronic platform for digital and analog signal processing. Besides its 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 this new standard.

The DESY MicroTCA.4 User tool kit (MTCA4U) provides drivers, and a C++ API for accessing the MicroTCA.4 devices and interfacing to the control system. The PCIexpress driver is universal for basic access to all devices developed at DESY. Modularity and expandability allow to generate device-specific drivers with a minimum of code, inheriting the functionality of the base driver. A C++ API allows convenient access to all device registers by name, using mapping information which is automatically generated when building the firmware. A graphical user interface allows direct read and write access to the device, including plotting functionality for recorded raw data. Higher level applications will provide callback functions for easy integration into control systems, while keeping the application code independent from the actual control system in use.

INTRODUCTION

The MicroTCA.4 crate standard [1, 2] provides a platform for digital and analog data processing in one crate. It is geared towards data acquisition and control applications, providing a backplane with high-speed point to point serial links, a common high-speed data bus (PCIexpress in this case) as well as clock and trigger lines. In typical control applications large amounts of data have to be digitized and processed in real-time on the front end CPU of the MicroTCA.4 crate.

MTCA4U—The DESY MicroTCA.4 User Tool Kit

The main goal of the DESY MicroTCA.4 User Tool Kit (MTCA4U) [3] is to provide a library which allows efficient, yet easy to use access to the MicroTCA.4 hardware in C++. The design layout of the tool kit is depicted in Fig. 1.

THE LINUX KERNEL MODULE

The Linux kernel module (driver) provides access to the MicroTCA.4 devices via the PCIexpress bus. As the basic access to the PCIexpress address space is not device dependent, we follow the concept of a universal driver for all MicroTCA.4 boards. The kernel module uses the Linux Device Driver Model which allows module stacking, so that the driver can be split into two layers: A universal part provides all common structures and implements access to the PCIexpress I/O address space. The device specific part implements only firmware-dependent features like Direct Memory Access (DMA), and uses all basic functionality of the universal part. For all devices developed at DESY the firmware will provide a standard register set and the same DMA mechanism, which permits to use a common driver for all boards. For devices from other vendors the universal part enables out-of-the-box access to the basic features, which can be complemented by writing a driver module based on the universal driver part. Like this the interface in MTCA4U does not change and the new device is easy to integrate into existing software.

THE C++ DEVICE API

The basic high level API provides C++ classes which allow access to all the functionality provided by the kernel module without requiring the user to have knowledge about implementation details like IOCTL sequences. In addition, it will have a callback mechanism for hot-plug events, which allows the application to go into a safe state.

A main component of the C++ library is the register name mapping. With evolving firmware the address of a register can change in the PCIexpress I/O address space. To make the user code robust against these changes, the registers can be accessed by their name instead of using the address directly, which also improves the code readability. The required mapping file is automatically generated by the Board Support Package together with the firmware. Performance overhead due to repeated table look-up is avoided by the use of register accessor objects, which cache the address and provide fast access to the hardware. Currently the mapping file implementation is being changed from plain text files to XML. This allows for more flexibility and enables new features like automatic data type conversion, for instance fixed point to floating point or signed 24 bit integer to system types.

GRAPHICAL USER INTERFACE

The mapping file, containing information of all the PCIexpress registers implemented in the firmware, allows to display this information in a graphical user interface. The Qt Hardware Monitor lists all registers and their properties, and permits the user to interactively display and modify their content. As the mapping file is automatically generated together with the firmware, this tool can be used for debugging

* This work is supported by the Helmholtz Validation Fund HVF-0016 "MTCA.4 for Industry".
† martin.killenberg@desy.de

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO104

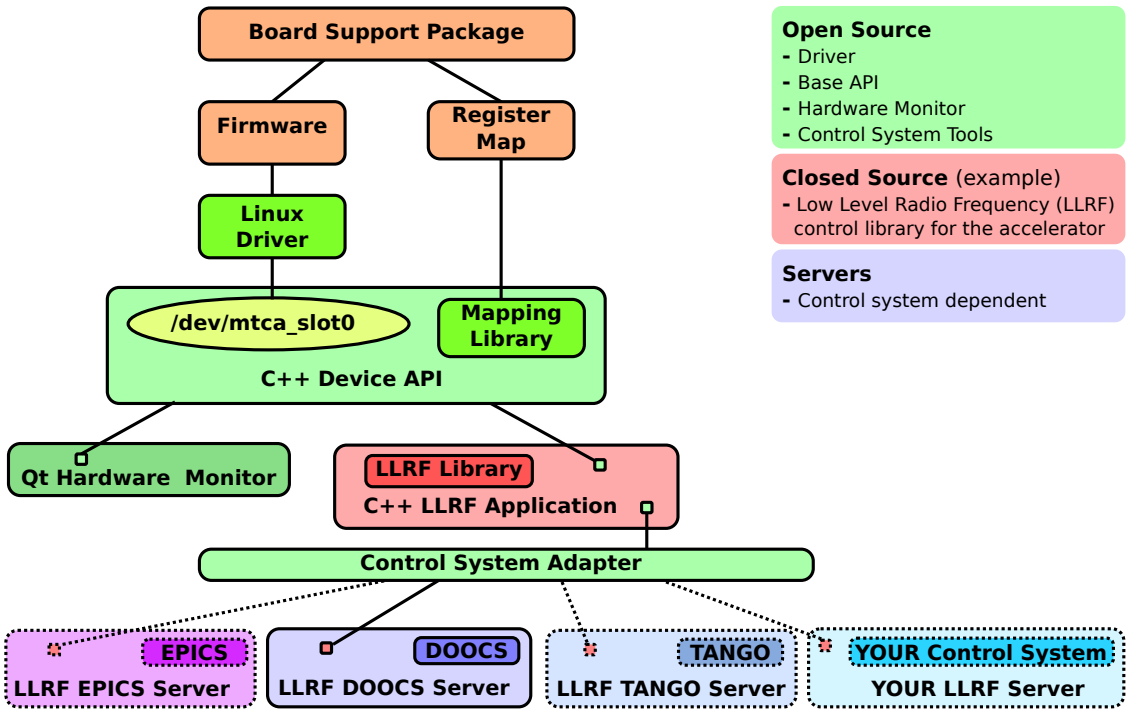

Figure 1: The design concept of the MicroTCA.4 User Tool Kit MTCA4U.

and prototyping immediately after the firmware has been deployed. It is written using Qt [4], an open source, cross-platform user interface framework which is available on all Linux platforms. A screen shot of the Qt Hardware Monitor is shown in Fig. 2.

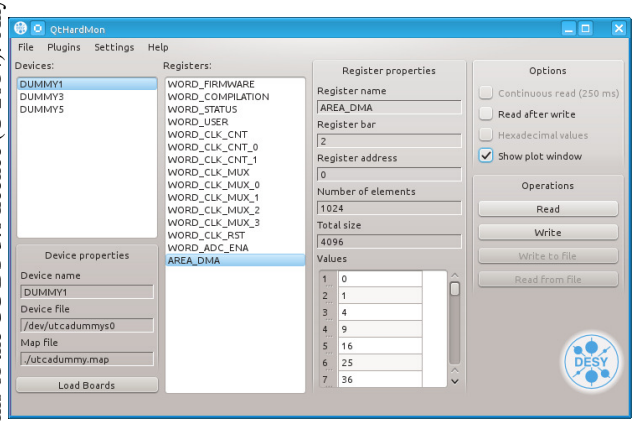

Figure 2: A screen shot of the Qt Hardware Monitor.

THE CONTROL SYSTEM ADAPTER

For larger control applications the different components use a control system or a middleware layer to communicate with each other. For this purpose the control system provides data structures which are used inside the user code. This causes a strong coupling of the code to the control system.

As it is expensive and time consuming to develop control algorithms, it would be desirable to have the core application portable between different control system. This brings contradicting requirements: On one hand the application has to communicate via the control system protocol and provide functionality like logging and data history, one the other hand it should be independent from the specific control system implementation without reimplementing the functionality.

The approach in MTCA4U is the introduction of a control system adapter layer, which should be as thin as possible. The business logic, which is independent from the control system, is only talking to the adapter and does not use the control system directly. The adapter comprises two components: A process variable adapter (Fig. 3) and a callback mechanism to react on control system events.

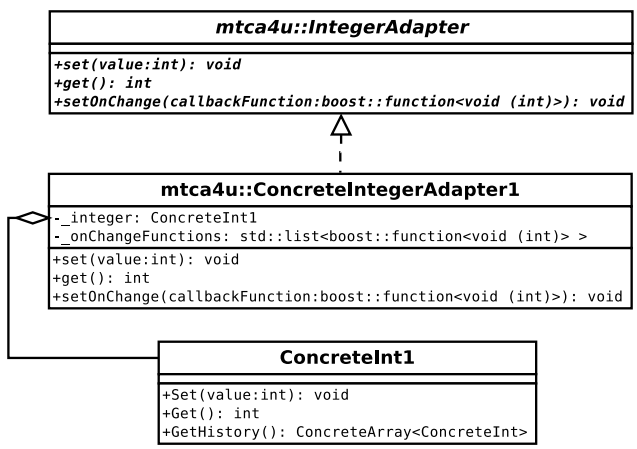

Figure 3: The process variable adapter for an integer type variable. An instance of the concrete control system data type is living within the adapter. Like this there is only a single copy of the data content, which avoids synchronization issues. The control system specific function *GetHistory()* is intentionally not reflected in the adapter interface.

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO104

The Process Variable Adapter

The general idea is to keep the application code agnostic from the fact that it is running in a control system. From the point of view of the business logic the process variables look like normal simple data types, or arrays of simple data types. Each variable has accessor functions (getters and setters), and a callback function can be registered which is executed when the variable content changes. Like this the process can react on changes coming from the control system. In addition to these methods, arrays will have iterators and random access operators. It is planned to have an interface similar to the C++ std::array. All other control system specific functionality is not reflected by the interface of the process variable adapter. This approach also has an impact on the design of the application code: In principle it has to be able to run without a control system attached. The advantage here is that one can run it for testing with a small, local script or GUI without the need to set up a control system environment. In addition it simplifies unit testing.

The copying of large data structures from the application to the control system has to be avoided for performance reasons. To realize this, an instance of the control system data type lives within the MTCA4U process variable and is accessed directly. This avoids unnecessary copying and synchronization overhead because it is the only copy of the variable content. For large data structures these process variables will be templated adapters to the control system classes, so the application is interfaced to the control system by recompilation without the need to modify the source code.

Control System Action Interface

The application will probably have to react on events coming from the control system, trigger information to synchronize different processes for instance. It is foreseen to provide an interface to register callback functions with the control system adapter with actions to be executed upon the arrival of certain events. The signatures of these functions again do not use control system specific data types and are connected to the control system in the concrete implementation of the adapter.

Implementations of the Control System Adapter

An application which is written against the control system adapter instead of a specific control system can be easily ported to a new control system by writing an adapter implementation. Once this new adapter exists, all other applications using the control system adapter will also be available for the new control system. Like this the availability of applications for a control system is increased, as well as the possible sites of operation for a specific program.

The two first adapters to be written are for DOOCS [5], which is the control system used for FLASH and the European XFEL [6] at DESY, and EPICS [7], as one of the most widely used control systems for particle accelerators.

CONCLUSIONS

The DESY MicroTCA.4 User Tool Kit MTCA4U is a C++ library which allows convenient access to MicroTCA.4 boards via PCIexpress. It comprises a modular, expandable Linux driver, an API with register name mapping and a graphical user interface for fast prototyping. A control system adapter is currently being developed, which allows the application code to be independent from the actual control system in use. This makes the business logic portable between control systems with minimal effort and allows a wider field of application for software written using MTCA4U.

REFERENCES

[1] PICMG®, "Micro Telecommunications Computing Architecture, MicroTCA.0 R1.0", 2006.

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

[3] MTCA4U—The DESY MicroTCA.4 User Tool Kit, Subversion Repository https://svnsrv.desy.de/public/mtca4u

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

[5] The Distributed Object Oriented Control System (DOOCS), http://doocs.desy.de

[6] M. Altarelli et al., "XFEL : The European X-Ray Free-Electron Laser : Technical Design Report", DESY-2006-097, DESY, Hamburg, 2007.

[7] Experimental Physics and Industrial Control System (EPICS), http://www.aps.anl.gov/epics/index.php