Contribution #18

IMPLEMENTATION OF LLRF CONTROL SOFTWARE OUTSIDE OF DESY USING EPICS

Patrick Nonn†, Julien Branlard, Martin Hierholzer, Martin Killenberg
Deutsches Elektronen-Synchrotron DESY, Germany

Abstract

Originally, the LLRF control software, developed for the accelerators at DESY (XFEL, FLASH,...), was exclusively based on DESYs in-house developed Distributed Object-Oriented Control System (DOOCS).

As DOOCS is rarely used outside of DESY, the ChimeraTK framework has been developed, to enable the LLRF control applications to use other control systems, like EPICS or OPC-UA. Especially EPICS-based LLRF control applications have been implemented outside of DESY, lately.

On this Poster we introduce some of the basic concepts of EPICS and ChimeraTK, and show how they come togeher, to form an EPICS-based ChimeraTK application. Then we will present some instances, where such an application was implemented.

ChimeraTK Overview

ChimeraTK is a software framework, developed by DESY, to separate the development of controller applications from the development of the hardware-related backend and the integration into a control system frontend. Hence it is named after the figure of greek mythology with multiple heads and tails. Some of its noteworthy features are:

- Open source code, available on github: https://github.com/ChimeraTK
- Control system adapters (frontends) available for EPICS, DOOCS and OPC-UA
- Device Access (backends) available for PCIe, Modbus, Python, DOOCS, EPICS, OPC-UA and others

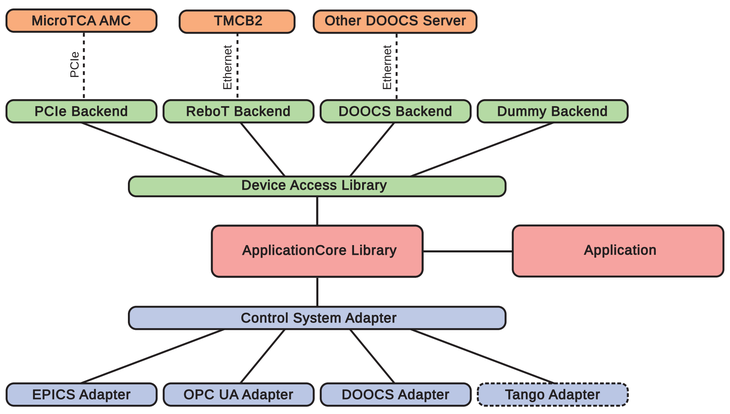

Figure 1: Modular concept of a ChimeraTK application

EPICS Concepts

- EPICS is a distributed control system, consisting of:
  - Servers called Input-Output Controllers (IOCs), which are usually installed close to the devices
  - Clients, i.e. Control System Studio, Matlab, Command Line Tools, usually running on remote locations, i.e. a control room
- Device Support(s) handle the communication to the device(s)
- Central part of the IOC is the database
  - Data, which is read from/send to devices, is managed by Process Variables (PVs)
  - Each PV is defined by a record in the database
  - The properties of a record are defined by its record type
  - The record types and their connection to the device support(s) are defined in the database description
- Various modules and extensions are available, which can add functionality to the IOC, i.e. persistence
- The operation of the IOC is controlled from a shell environment
- The communication between IOC and client is handled by EPICS own Channel Access (EPICS 3) or pvAccess (EPICS 7) protocol

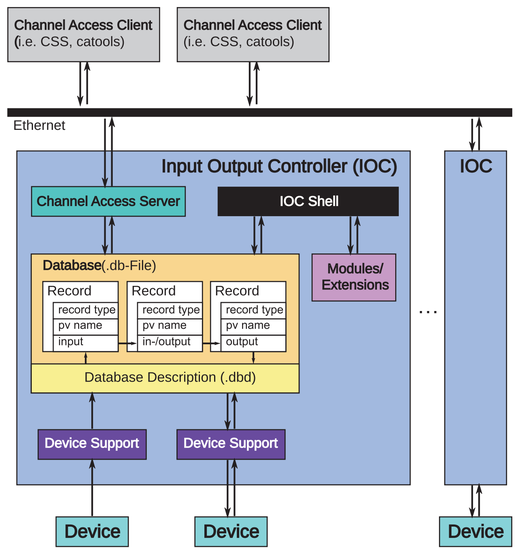

Figure 2: Schematic of general EPICS application

EPICS-based ChimeraTK LLRF Control Server

The LLRF Control Server controls the algorithms running on the FPGA. It has been implemented as a ChimeraTK application and is open source, available on DESYs Gitlab: https://gitlab.desy.de/msk-sw/low-level-radio-frequency/llrfctrl

Some of its features are:

- Pulsed and CW operation
- Pulse shape generation with multiple beam regions
- Pulse-to-pulse amplitude and phase controller
- Learning feed forward
- Beam loading compensation

Building a ChimeraTK application against an EPICS Control System Adapter results in an IOC. The ChimeraTK Application itself, containing the server logic, including the backend(s), acts, in EPICS terms, as a Device Support.

- The PCIe backend allows the LLRF control server to read/write registers on the FPGA, which is running the control algorithm. This requires a PCIe-endpoint, provided by either the pcieuni- or the xdma-driver, to be present in the system, as well as a proper mapping of the registers.
- The LLRF control server requires a connection to the x2timer-server, to receive an interrupt. Unfortunately, the x2timer-server was developed before ChimeraTK and would need to be redeveloped as an ChimeraTK application in order to be built as an EPICS server. Thus, DOOCS has still to be present in the system and is connected with the EPICS-llrfctrl-server via a DOOCS backend.

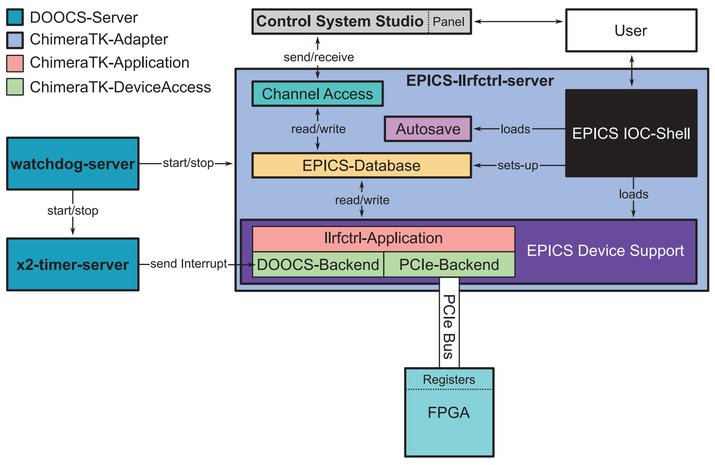

Figure 3: EPICS-based single cavity LLRF control software, including accompanying DOOCS servers

The EPICS Control System Adapter also provides the database definition (.dbd-file) and includes the Autosave module. But it does not generate the database itself. That, as well as the .req-files for autosave and the start script, among other files, has to be provided by the server configuration.

EPICS-based LLRF Control Systems at the Helmholtz Zentrum Berlin

DESY and the Helmholtz Zentrum Berlin (HZB) cooperated to set up two MicroTCA-based LLRF control systems, one for the booster of BESSY and one at SEALAB (formerly BerlinPRO). Both systems use firmware and software, developed at DESY. As the HZB prefers EPICS for a control system, we set up an EPICS software, as described above, including panels for the Control System Studio Phoebus.

If you would like to learn more about the user experience with those systems, please visit these posters:

- "BESSY-II new digital mTCA.4-based LLRF control for the booster upgrade" by Pablo Echevarria Fernandez
- "Status of the Helmholtz Zentrum Berlin Sealab LLRF infrastructure" by Andriy Ushakov

Coming Soon: Generic ChimeraTK Server

The LLRF Control Server requires specific firmware to work and vice versa. In order to make it easier to access firmware without existing software, we provide a simple, generic ChimeraTK server.

If set up correctly, it allows to access the registers, defined in the mapp file, without any additional logic. This is possible due to a feature of ChimeraTK that automatically publishes variables, provided by device access, to the control system adapter, if it is not used in the application. Thus the ChimeraTK Application of the generic server is actually the most simple application, possible.

The Generic ChimeraTK Server is open source. It can be found on GitHub:

https://github.com/ChimeraTK/GenericDeviceServer

Especially in combination with an EPICS Control System Adapter has various possible use-cases:

- As the Generic ChimeraTK Server is open source, it can be used as a starting point for your ChimeraTK application development.
- It can be used as a classical EPICS device support for MicroTCA devices. The application logic can than be implemented in the EPICS database, as usual.
- It can be used to provide access to distributed MicroTCA devices for some high level software, based on i.e. Python with PyEPICS

EPICS Server Configuration

An EPICS-based ChimeraTK server by itself only provides an IOC shell with the ChimeraTK application and the Autosave module implemented. In order to successfully start the server, the following components have to be provided.

EPICS Database

Traditionally, the logic of an EPICS sever is implemented in the database, by defining different kinds of records and linking them together. In the ChimeraTK LLRF control server the logic is located in the ChimeraTK Application. Thus the database simply passes data from the ChimeraTK Application aka Device Support to the Channel Access (or pvAccess) Server.

- Each ChimeraTK Application provides an "-xmlGenerator" executable
  - It generates a list of all variables including some metadata
  - The data is written into an xml-file as depicted in Figure 4
- The ChimeraTK Application organizes its variables in a tree structure, which can result in quite long variable names. As the name length for PVs is limited to 60 characters (in EPICS 3.16) the variable names can not be easily adapted as PV names.
- Some information, needed for an EPICS record (i.e. the record type) can not be provided by the application.

```
<application xmlns="https://github…" name="llrfctrl">
  <directory name="Controller">
    <directory name="Status">
      <variable name="overall">
        <value_type>int32</value_type>
        <direction>application_to_control_system</direction>
        <unit></unit>
        <description>Controller configuration…</description>
        <numberOfElements>1</numberOfElements>
      </variable>
    </directory>
  </directory>
</application>
```

Figure 4: Extract from ChimeraTK variable file

In order to make it easier to generate db files, I developed a tool, which uses a configuration file to store data, missing in the variable xml file. This tool, named "dbGenerator", is available as part of the EPICS-IOC-Adapter at Gitlab:

- https://github.com/ChimeraTK/ControlSystemAdapter-EPICS-IOC-Adapter/tree/master/tools
- Preliminary database configuration files can be generated from variable xml files.
- The configuration file is written in xml (see Figure 5).
- Records are organized in groups of the same type.
- Fields can be configured on every level.
  - The lower level overwrites the higher one.
- Field values can be read from the variable xml file.

A configuration file can than be processed by the dbGenerator, whose output includes:

- one or multiple database files
- ".req"-files for autosave
- text files, holding the descriptions for the variables, if provided by the ChimeraTK application

```
<EPICSdb xmlns="<namespace>" application="llrfctrl">
  <sourcefile type="xml-variables" path="llrfctrl.xml" label="llrfctrl">
    <alias handle="Ctrl" surrogate="Controller/" />
    <alias handle="CtrlFF" surrogate="+{Ctrl}+{FF}" />
    <alias handle="Ff" surrogate="FeedForward/" />
  </sourcefile>
  <outputfile path="<path>" autosavePath="<path>" macroReserve="8">
    <field type="DTYP" value="ChimeraTK" />
    <recordgroup type="aai" autosave="false">
      <field type="SCAN" value="1 second" />
      <field type="INP" value="@${APP} +{:address}" />
      <field type="EGU" value="+{:unit}" />
      <field type="FTVL" value="+{:value_type}" />
      <field type="NELM" value="+{:numberOfElements}" />
      <field type="PREC" value="6" />
      <record pvName="${Server}/CtrlFF/FTRatio" source="llrfctrl.+{CtrlFf}flattopRatio" />
      <record pvName="${Server}/B0Cfg/BCF" source="llrfctrl.+{Adcb0Cfg}BCF">
        <field type="EGU" value="MHz" />
      </record>
    </recordgroup>
  </outputfile>
```

Figure 5: Exemplar database configuration file

Mapping

Part of the server configuration is the interface between the firmware and the ChimeraTK application. This interface consists of three different kinds of files:

- ".mapp" files associate labels for the registers with their addresses and have to be provided by the firmware developer.
- ".xlmap" files mapp the registers from the mapp file to the ChimeraTK applications variables and should come with the software.
- The .dmap file associates a PCIe endpoint with a mapp file (see Figure 6). As this depends on the set up, this file has to be provided by the system integrator.

```
x2timer    (doocs:TEST.DOOCS/X2TIMER/NEUROMANCER.0?cacheFile=timer.cache)
Timer      (logicalNameMap?map=timer.xlmap)
CtrlBoard  (pci:pcieunis3?map=./llrf_scav_sis8300ku_regae_1.1.4-2-g078e9dfc.mapp)
Controller (logicalNameMap?map=llrfctrl_controller.xlmap)
```

Figure 6: Typical content of dmap file

Start Script

The EPICS IOC is configured and started by executing various commands in the IOC shell. On Linux systems this can be done with an executable script. This has the downside, that the process is attached to the shell, it is started from and will be terminated, if that shell is closed.

In order to demonize the process, the tool "procServ" (see https://linux.die.net/man/1/procserv) is often used. It features access to the IOC shell via telnet and a configurable logging option. The relatively unsecure network access via telnet is also a downside of "procServ".

An alternative to "procServ" is "screen" (see https://linux.die.net/man/1/screen). Screen is a window manager, which emulates a terminal. The screen terminal can be detached from the current shell, which allows it to run in the background. It can than be reattached to any local shell, owned by the same user, to allow access to the screen terminal. As this requires an SSH access to access the IOC shell remotely, it is inherently more secure, than procServ. It is possible to fork the standard input/output of the screen terminal into a file, allowing for logging.

Screen can be configured to use the IOC shell. This way, if the IOC crashes, the screen terminal is terminated, too, allowing daemon managers (i.e. watchdog, systemd) to automatically restart the EPICS server.

†patrick.nonn@desy.de

Low Level RF Workshop 2022
9-13 Oct 2022, Brugg-Windisch, Switzerland

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

PAUL SCHERRER INSTITUT PSI

CERN

ChimeraTK

EPICS