Software design and implementation for LLRF modules in the European XFEL.

F. Makowski*, W. Cichalewski, A.Napieralski, DMCS, Łódź, Poland
M. Killenberg, J. Branlard, H. Schlarb, DESY, Hamburg, Germany
A. Piotrowski, FastLogic Sp. z o.o., Łódź, Poland

Abstract

LLRF modules in the European XFEL are a set of devices that need a common interface for management and control. A generic software structure was designed on the top of Distributed Object Oriented Control System (DOOCS) framework to allow easy and flexible implementation of new applications, that provide direct access to such devices as the Drift Compensation Module (DCM), Local Oscillator & Generator Module (LOGM) and Power Supply Module (PSM). The current design's structure allows decoupling from DOOCS framework without introducing much performance overhead, also allowing the addition of safety redundancy mechanisms, which are not available in the framework itself. The design has proven to work with register-based protocol access (SNMP, DMA and PCIE readout), but does not restrict usage for just device access, as it can be used for LLRF data computation and any other action-on-demand features. This contribution presents the software architecture, its implementation and some experimental results acquired on devices already installed in the XFEL accelerator.

Introduction

The European X-Ray Free Electron Laser (XFEL) is an accelerator consisting of 25 RF stations, each of them having both master and slave racks, that gather hardware modules fullfilling separate tasks. Some of those modules are:

- **Drift Compensation Module** (DCM, Fig. 1.) – module used for cavity channel attenuation control, as well as phase and amplitude drift compensation
- **Local Oscillator Generation Module** (LOGM, Fig. 2.) – module generating the 1.3 GHz reference signal and distributes several signals for the down converters and digitizers
- **Power Supply Module** (PSM, Fig. 3.) - power provider for other modules, designed as redundant and modular, with doubled units to allow hot swapping

Those modules use different **register-based protocols** (SNMP, direct PCI-E access or PCI-E over TCP/IP connection), which rises the necessity of unifying the structure of the applications to allow management and control over the devices.

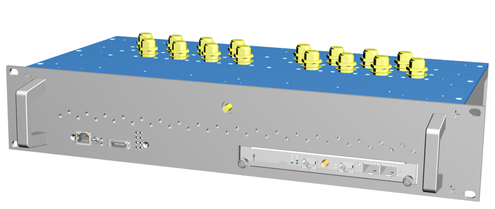

Figure 1:
Drift Compensation Module

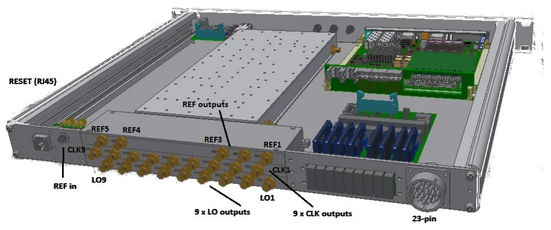

Figure 2:
Local Oscillator Generation Module

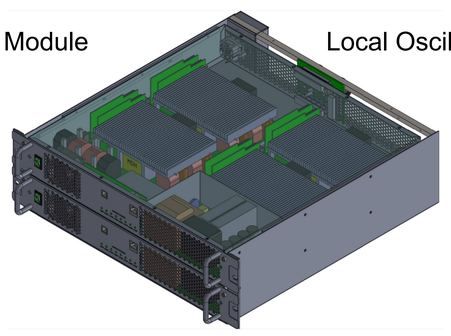

Figure 3:
Power Supply Module

Basic DOOCS Framework Data Access

DOOCS (Distributed Object-Oriented Control System) is a RPC-based network framework developed in DESY, and used in such facilities as FLASH or the European XFEL.

The basic principle of DOOCS framework is dynamic allocation as device-oriented instances called locations, that works within their own threads fulfiling the **update design pattern**. Each of these objects has their own set of data containers, called properties, that allows data manipulation over centralized RPC-based network.

Figure 4. shows the sequence diagram describing the simplified object control through calls performed by the user from the external DOOCS graphical interface panels. The diagram is divided in three sections, first showing actions that are done for each property during the init() process. Two other sections give an overview of how the writing and reading tasks are handled.

Early development has shown, that implementation of the sequence as shown on Fig. 4 can be performance limiting and complex, since it requires handling asynchronous calls synchronously and does not couple accessors against properties. After considering possible ways to simplify and unify the structure and communication flow, the idea of implementing a **component pattern design** arised, allowing injection of components called **Actions**, that give the possibility of calling methods during asynchronous get() and set() RPC calls.

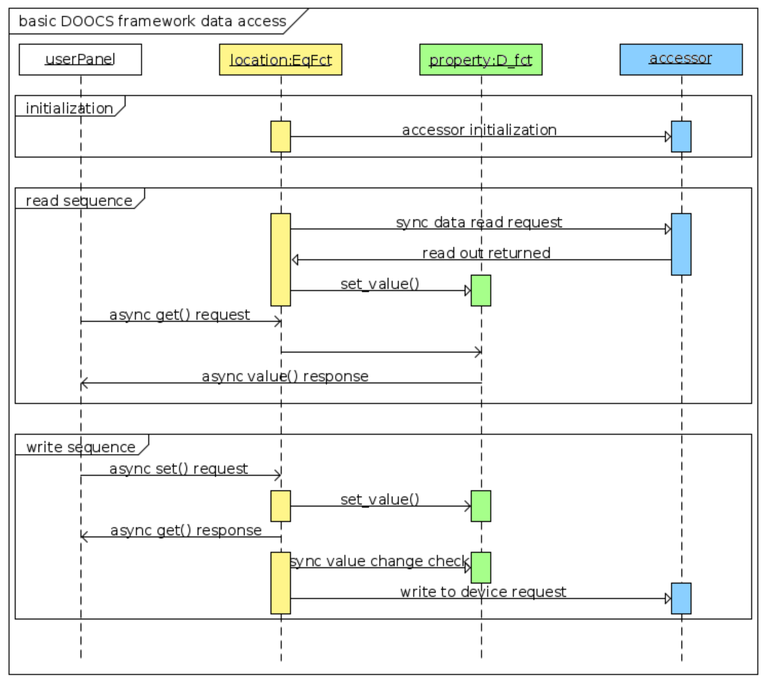

Figure 4:
Sequence diagram for basic DOOCS framework data access

Action-Based Enhanced DOOCS Framework Data Access

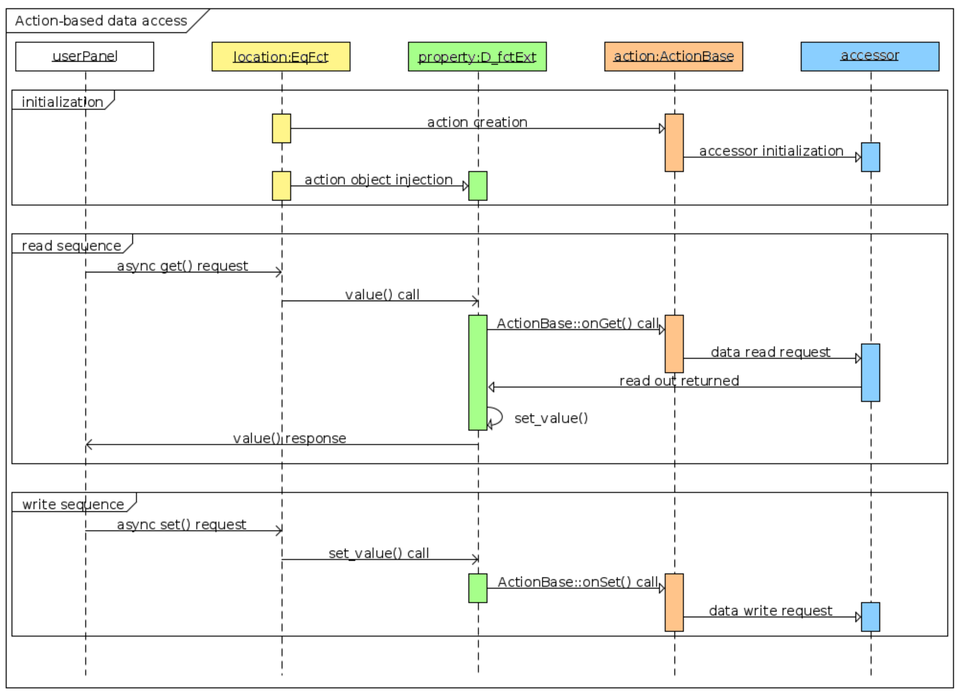

Figure 5:
Sequence diagram for Action-based enhanced data access

The Action-based implementation requires a little more work to be done during the initialization phase, but it pays off in the read/write stages, as it does not require location object to interfere with the data access.

On Fig. 5. the enhanced data access is presented. The major diferences from the sequence diagram shown on Fig. 1. are

- **async-only read/write access**, which simplifies the structure from the sync/async hybrid implementation
- Introduction of another object, which is an Action, that is firstly bind to a specified accessor, and then injected into the component-pattern-extended property.
- Location operations only during the initialization phase; This takes the major implementational burden from the location class (which now consists only of definitions, configuration and other high-level tasks) to the ActionBase derivative classes.

Results & Summary

The idea was proven to work with all of the mentioned LLRF module servers and it allowed fast and flexible application implementation according to the module-specific requirements.

Although the current approach is to use the implementation for the register-based communication with hardware, it can be extended over wide range of other tasks, such as value validation and calculation, conversions and implementation of redundancy mechanisms triggered on the fly while the value is retrieved or set immediately.

Low Level Radio Frequency Workshop 2015
2015/11/03 – 2015/11/06 Shanghai

* corresponding author:
filip.makowski@desy.de