Hardware Abstraction in Analogue

S. Deghaye, L. Bojtar, Y. Georgievskiy, J. Serrano.

CERN, Geneva, Switzerland.

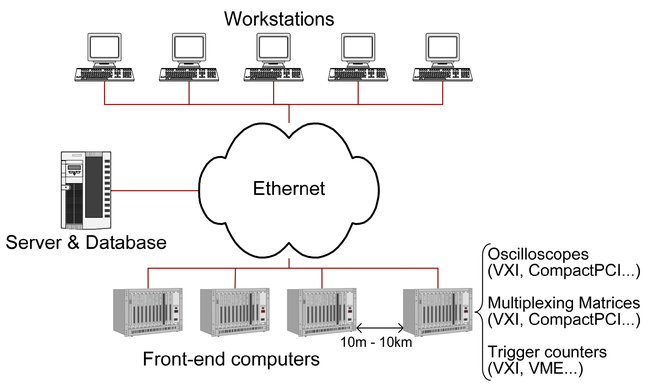
Abstract

OASIS, the Open Analogue Signal Information System, is based on a three-tier architecture. The front-end tier, the lowest part, controls the hardware and provides a uniform interface to the application server in the middle tier. However, we do not want to restrict the functionalities provided to the user to the level of the hardware with the fewest capabilities. To achieve these contradictory goals, the front-end tier model is made of several classes with relationships between them. Each class represents an element of the model e.g. an oscilloscope or an oscilloscope channel. The relationships allow us to navigate in the model e.g. to go from a channel to its enclosing oscilloscope. The model lets most of these elements be optional to account for the heterogeneity of the different installations. The class interfaces are made of properties where a property can be seen as a parameter of the controlled hardware. Each property has a standard structure that provides not only the current value of the parameter but also how it can vary. This way, the system is able to support high-end hardware with a lot of possibilities but also cheaper or older hardware with fewer capabilities.

Today, there are three implementations of the front-end model under development. The first two are oscilloscopes and analogue matrices in CompactPCI and VXI formats. The last one is the diagnostics part of the LEIR RF beam control - a completely digital component - showing that the model is flexible enough to support very different hardware.

OASIS in brief

OASIS is a system for the acquisition and display of analogue signals in the accelerator domain. The signals, distributed all around the accelerators, are digitalised by oscilloscopes lodged in front-end computers (FEC). The acquired data is sent through the Ethernet network and displayed on a workstation running a specific application, the OASIS viewer. When the bandwidth requirement allows it, the analogue signals are multiplexed by analogue matrices which are connected to the oscilloscope channels. This scheme takes into account the fact that not all the available signals are observed at the same time and allows us to save some digitisers, the most expensive devices in the system. The FECs are installed next to the signal sources in order to preserve the signal integrity as much as possible.



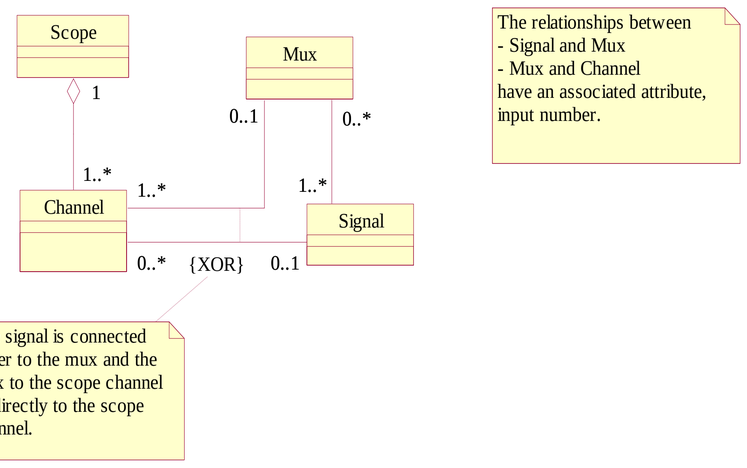
OASIS is based on a three-tier architecture ([1] and [2]). The front-end tier, the lowest one, has the responsibility to handle the hardware, the digitizer and the multiplexer modules. It provides a hardware independent interface to the upper tiers.

Front-end Interface Structure

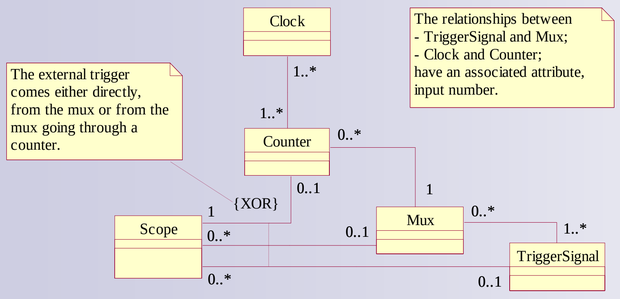
The front-end interface is composed of several classes. Each class can be related to the others in some predefined ways using relationships. The interface can be split in two parts, the signal part which is concerned with the analogue signals and their acquisitions and the trigger part which focuses on the oscilloscope triggering.

Front-end Interface—Signal Part

- Signal class: it represents the analogue signals we want to observe.
- Channel class: here, we find all the settings that are channel dependent such as the sensibility or the DC offset. It contains also the acquired waveforms.
- Scope class: it controls the global settings - those which affect all the channels in a scope - such as time span or trigger delay time. A scope can have one or several channels and its settings have the priority on those belonging to the enclosed channels
- Mux class: the Mux controls the multiplexing matrices that can be installed in front of the oscilloscope channels. The Mux class is optional.



Front-end Interface—Trigger Part



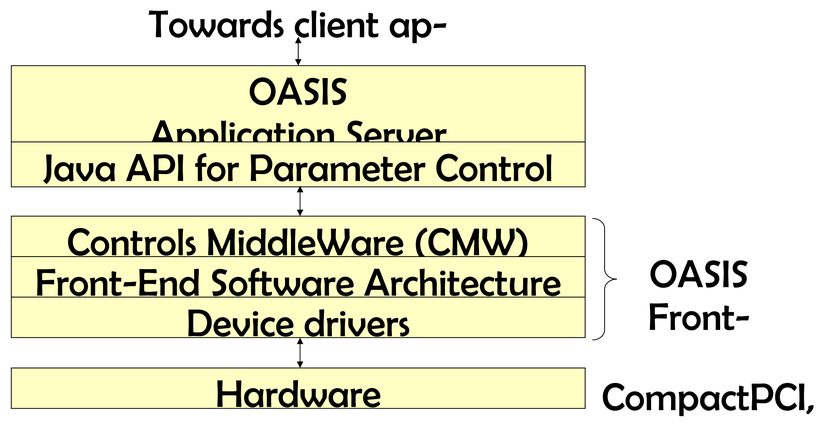
- TriggerSignal class: it represents the trigger pulses we want to acquire the analogue signals with.
- The Mux class has the same role as the one used in the signal part except that it multiplexes trigger pulses instead of analogue signals.
- The Counter class controls the counters that may be installed at the multiplexer output. These counters are used to delay the trigger pulses using an external clock.

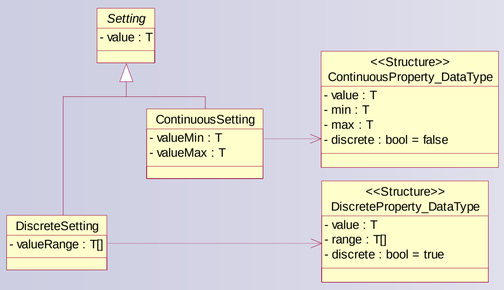
In this part of the interface, only the TriggerSignal and the Scope Class are mandatory.

Implementation & Property Structure

The implementation of the front-end interface is based on the Front-End Software Architecture (FESA)[5] framework. For the links between instances, we use the device relationship database and the directory service [6]. The communication with the OASIS application server is based on the Control MiddleWare (CMW) [7].

Today, we have at least one implementation for all the classes. Implementing the Scope and Channel classes, we have FESA classes for the control of Acqiris CompactPCI oscilloscopes [3](the DC series) and HP VXI oscilloscopes[4]. The Mux class is implemented by the class controlling Pickering CompactPCI multiplexers [8] and other CERN made modules.





In the Front-end interface, all the properties that represent a setting (e.g. Timebase, Sensibility...) are of the type Setting, either ContinuousSetting or DiscreteSetting. This Setting hierarchy allows the OASIS application server to query the capabilities of the underlying hardware in a uniform way and, therefore, to use the module features to their maximum.

However, these classes cannot be used as such in the FESA framework. First, because FESA supports only primitive types and, second, because the CMW is data oriented. To solve this problem, we define a mapping between the ContinuousSetting and DiscreteSetting classes and two data structures that we use in FESA class designs each time we need to implement a property returning the Setting type.

Reference

[1] S. Deghaye *et al.*, "OASIS: a new system to acquire and display the analogue signals for LHC", ICALEPCS'03, Gyeongiu, Korea, October 2003.
[2] S. Deghaye *et al.*, "OASIS: Status report", these proceedings.
[3] http://www.acqiris.com/products/digitizers/8-bit-cpci-6u-digitizers
[4] http://www.agilent.com
[5] A. Guerrero *et al.*, "CERN Front-End Software Architecture for accelerator controls", ICALEPCS'03, Gyeongiu, Korea, October 2003.
[6] J. Cuperus *et al.*, "The Directory Service for the CERN accelerator control application programs", these proceedings.
[7] K. Kostro *et al.*, "Controls Middleware (CMW): Status and use", ICALEPCS'03, Gyeongiu, Korea, October 2003.
[8] http://www.pickeringswitch.com/dynamic/main.cgi?myaction=showprod;ref=2361