Atacama Large Millimeter Array

Generic abstraction of hardware control based on the ALMA Common Software

B.Jeram, G. Chiozzi, J.Ibsen (ESO), R.Cirami (AOT),
M. Pokorny (NRAO), D. Muders(MPIfR), D. Wischolek (RUB)

ADASS XIII

OVERVIEW

The ALMA Common Software (ACS) is a CORBA-based framework that provides a common and homogeneous infrastructure for the whole ALMA software, from high-level data flow applications down to instrument control. Different application domains inside the ALMA system are supported by specialized packages.
A high level description of ACS will be presented at this conference by J.Schwarz [1]. Aspects of ACS for high level applications will be presented by H.Sommer [2].

This paper focuses on ACS support for the development of Control System applications. In this domain, ACS provides a generic abstraction of hardware control and monitor points that is independent of the hardware underneath. This abstraction layer is coupled to the hardware using the DevIO (Device Input/Output) interface, based on the bridge design pattern. Application developers have to implement DevIO classes that handle the details of the communication with the hardware.

ACS itself provides a default DevIO implementation, which simply writes to and reads from a memory location. Currently there are two other major DevIO implementations available: a CAN bus communication, used by ALMA, and a socket based implementation used by the Atacama Pathfinder EXperiment (APEX) project.
Despite using different hardware and control electronics, the DevIO abstraction allows the ALMA and APEX projects to have the same device architecture down to the level of the DevIO implementation.
A demo will illustrate this and other basic concepts of ACS.

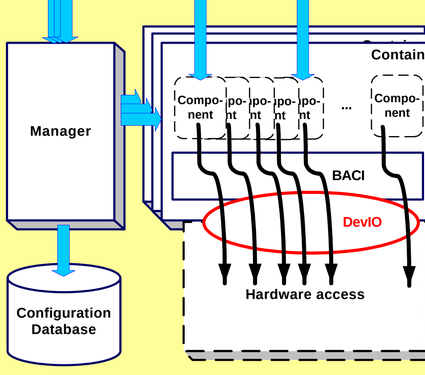
[1] J. Schwarz et. al., "The ALMA Software System", ADASS 2003
[2] H Sommer et. al., "Transparent XML Binding using the ALMA Common Software (ACS) Container/ Component Framework", ADASS 2003

1) What is ACS ?

The ALMA Common Software (ACS) is a set of application frameworks built on top of CORBA to provide a common software infrastructure to all partners in the ALMA collaboration. The main purpose of ACS is to simplify the development of distributed applications by hiding the complexity of the CORBA middleware and guiding the developers to use a documented collection of proven design patterns. It aims at providing an answer to the following needs:

- common application framework and programming model, not just libraries
- standardization offering environment with design patterns and their implementation (rather than pure rules)
- well tested software that avoids duplications
- make upgrades and maintenance reasonable
- incremental development via releases
- common configuration control/installation procedures

2) High-level architecture of an ACS based Control System



The ***presentation tier*** provides a GUI and an API interface for an application written using the ACS software to the end user, through the easy embedding of languages like Java and Python.

The **middle tier** (also called business logic tier) is where the devices and their interactions with the rest of the system are modelled using **Component/Property/Characteristics** design pattern (see box 3).

A device is represented as a **Component**.

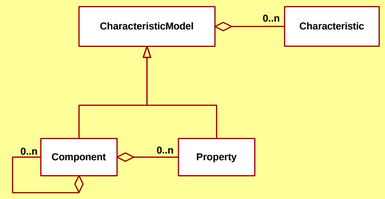
The underlying infrastructure of this tier consists of C++, Java or/and Python **Containers** where **Coomponents/devices** are deployed. One **Manager** manages the system with the help of the Containers.

The ***data-access tier*** is used for working with the information in the configuration database and other databases (e.g., the ALMA Archive which is where all monitor values are eventually stored).

The ***hardware-access tier*** is used for retrieving data from monitor points implemented in hardware, as well as sending commands to control points.

3) Component/Property/Characteristic design pattern

Component is a base class for any physical/logical Device, e.g. temperature sensor, motor.
Each **Component** has **Properties**, which represent monitor and control points,e.g. status value, position. They provide asynchronous (using callback mechanism) and synchronous retrieval and setting of values, monitors and alarms. Properties' interaction with the hardware is done using an abstract interface: **DevIO**. There are only a few different Property types - for each primitive data type one and for each sequence of primitive data types.
Components and **Properties** are described by **Characteristics**, e.g. name, unit, and minimum/maximum.



4) DevIO - generic abstraction of hardware

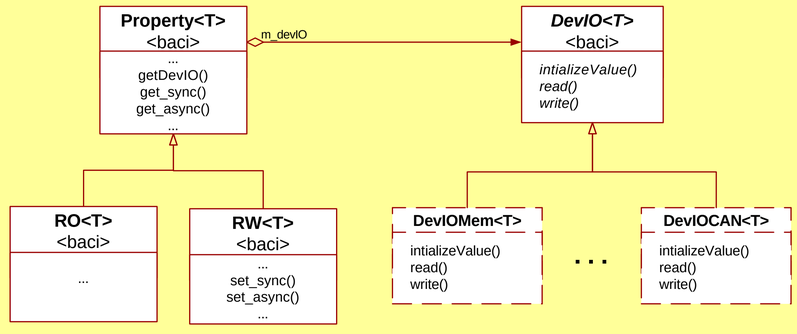
As a framework, one goal for ACS is to enable an easy way of integrating new hardware into the application.
This is done letting the high level applications to use the abstraction of **Properties**, while the **Property** itself interfaces to the hardware using a **DevIO**, hardware specific implementation.
Properties - high-level abstraction of control/monitor points - provide the application with facilities to monitor and retrieve values from monitor points (get_sync, get_async), and to set the value of control points (set_sync, set_async). Or to handle periodic telemetry logging and alarms.
Internally, **Properties** use a **DevIO** implementation to access the actual hardware.
DevIO is a simple and generic abstraction of hardware monitor and control point, based on the Bridge design pattern. The hardware access points are abstracted with the **DevIO** parameterized (template) interface, which defines two methods:

- read
- write

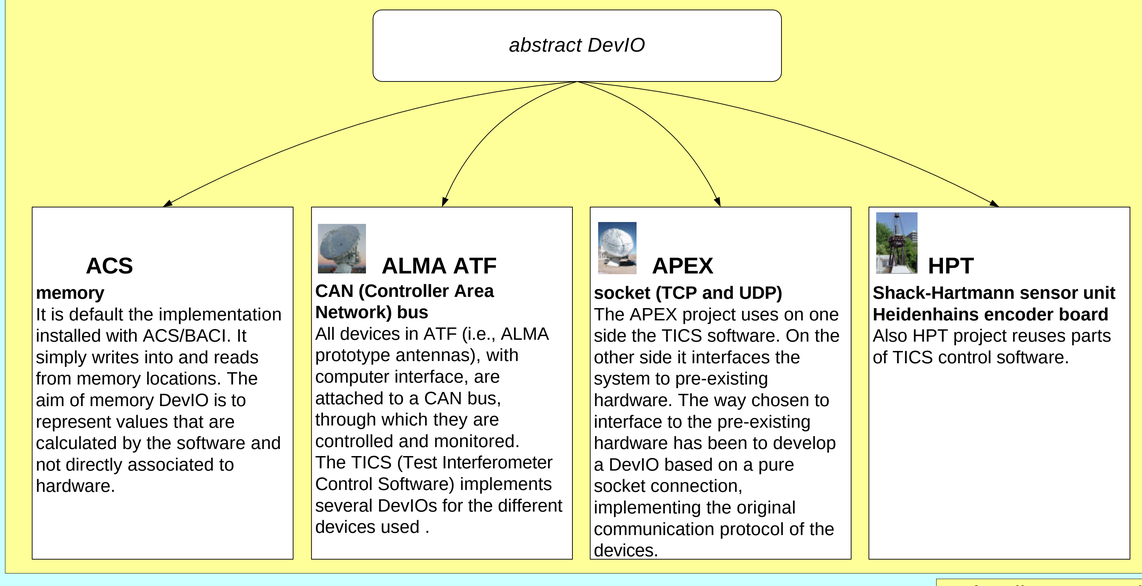
These are the only abstract methods used by the **Property** to access the hardware.
The implementations of **DevIO** interface expose these two methods and hide:

- communication with hardware (e.g. reading from and writing to hardware)
- conversion from raw values to engineer units and vice verse.

For example, the ACS high level code controlling a Power Supply is only aware of the SetPointCurrent control point and ReadbackCurrent monitor point.
If the actual Power Supply is controlled via an Analog I/O board, the properties are configured with a **DevIO** capable of reading and setting control registries of the board.
If this power supply is replaced by a new one controlled via a serial interface, only the **DevIO** implementation needs to be replaced.

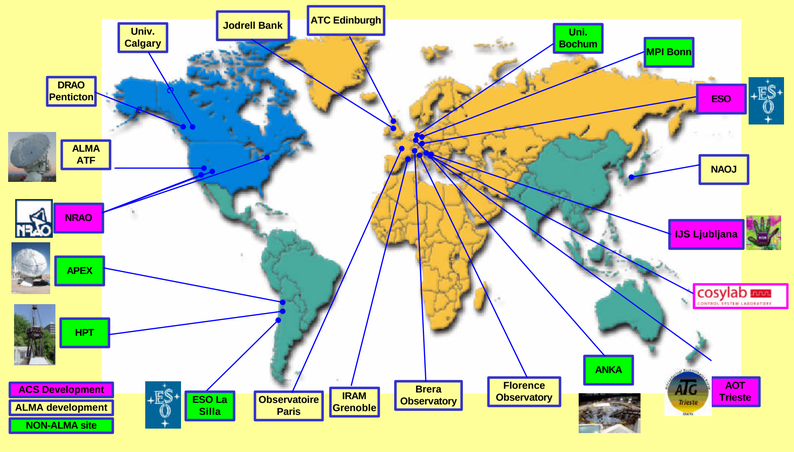


Implementations of DevIO



ACS Collaboration

ACS is developed for the ALMA Project and made available under the GNU LGPL Licence.
The development is distributed among the sites of various ALMA partners and external institutes collaborating to the project.
A number of external projects are already using ACS or are evaluating the possibility of using it.
This map shows the major sites involved in ACS Development, the ACS installations and some external projects using ACS.
The availability of ACS can trigger other collaboration projects, like eACS (embedded ACS)



http://www.eso.org/projects/alma/develop/acs