SPIRAL2 CONTROL COMMAND: A STANDARDIZED INTERFACE BETWEEN HIGH LEVEL APPLICATIONS AND EPICS IOCS

C. Haquin, P. Gillette, E. Lemaître, L. Philippe, D. Touchard, Ganil, Caen, France
F. Gougnaud, Y. Lussignol, CEA/DSM/IRFU, Saclay, France.

Abstract

The SPIRAL2 linear accelerator [1] will produce entirely new particle beams enabling exploration of the boundaries of matter. Coupled with the existing GANIL machine this new facility will produce light and heavy exotic nuclei at extremely high intensities. The field deployment of the Control System relies on Linux PCs and servers, VME VxWorks crates and Siemens PLCs; equipment will be addressed either directly or using a Modbus/TCP field bus network. Several laboratories are involved in the software development of the control system. In order to improve efficiency of the collaboration, a special care is taken to the software organization. During the development phase, in a context of tough budget and time constraint, this really makes sense, but also for the exploitation of the new machine, it helps us to design a control system that will require as little effort as possible for maintenance and evolution. The major concepts of this organization are the choice of EPICS, the definition of an EPICS directory tree specific to SPIRAL2, called "topSP2", this is our reference work area for development, integration and exploitation, and the use of version control system (SVN) to store and share our developments independently of the multi-site dimension of the project. The next concept is the definition of a "standardized interface" between high level applications programmed in Java and EPICS databases running in IOCs. This paper relates the rational and objectives of this interface and also its development cycle from specification using UML diagrams to test on the actual equipment.

INTRODUCTION

The Spiral2 Control System is designed with a typical EPICS architecture, relying on OPI Clients and IOC servers communicating using Channel Access (CA) protocol.

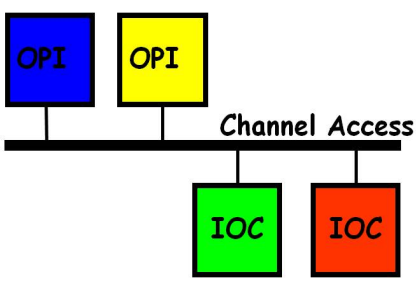

Figure 1: Spiral2 Control System Architecture.

The CA protocol allows OPI to read, write and monitor variables called Process Variables (PV) located in IOC. The CA protocol enable any OPI to access any PV as soon as it knows the PV name, there's no need to know which IOC hosts the PV.

OPI issues CA requests to IOC, which eventually interact with equipment to perform the actual read or write operation. This is the mean by which, over CA, functions are provided to OPI to fulfil its control tasks.

Hence, OPI need to know the names of the PV that correspond to its purposes. This can quickly become a big mess on OPI side since there are many type equipment to be driven and each developer could to adopt its own philosophy on both OPI and IOC sides.

It then appeared obvious that design should be optimized in order to reduce the development effort, but also in the machine exploitation perspective, to be able to face the evolution and maintenance requirements with a small team. Consequently, the decision was made to homogenize the way OPI control the various equipments through PV.

So, starting from the fact that PV names are almost completely determined by the naming convention, which take into account the localisation and the type of equipment [2], and the observation that equipment driving is always achieved through same kind of functions, we started to glimpse the standard interface concept in the sense that the naming convention should be pushed one step further in order to codify the remaining part of the PV name in correlation with the expected function.

This paper explains how the Standard Interface specifies the naming of the PV through which functions are provided to OPI, how it has been implemented on EPICS Data Base side (DB) and the first feedback is presented and next steps are envisaged.

SPECIFICATION OF THE STANDARD INTERFACE

Presentation

- Takes place in CA communication between OPI and IOC.
- Takes advantage of the fact that all equipment are drivable by the same kind of functions.
- Relies on a set of interface PV through which IOC provide the functions needed by OPI.

Goals

- Reduce development effort
- Reduce maintenance effort
- Ease component reuse

Functionalities

We noticed that on OPI side, independently from the equipment type, the following functions are always required:

- Write Consigns
- Write Commands
- Read Back Consign
- Read Measurements
- Read States
- Read Defects

PV Rules

Since functionalities are provided through PV, the name is related to the expected function. The rules are such that OPI can simply deduce the name of the needed PV from the needed functionality.

In the following, "$(CODIF)" refer to the first part of the PV name determined by the project naming convention, "PType" refer to the type of parameter to control (example I for current, V voltage).

Rules for consign related PV

Consign handling consist in write, read back, read actual value operations, each operation being handled by a record.

Table 1: PV Naming Rules for Consign Handling

Operation	PV Name
Write	$(CODIF):*PType*Cons
ReadBack	$(CODIF):*PType*ConsLect
ReadActual	$(CODIF):*PType*Act

In addition, filling of operator display fields is mandatory. In case of *PType*Act PV, alarm fields related to limit and severity are also mandatory.

Rules for measurements related PV

Measurement handling consists only in read actual value operation.

Table 2: PV Naming Rule for Measurement Handling

Operation	PV Name
ReadActual	$(CODIF):*PType*Meas

In addition, filling of operator display fields and alarm fields related to limit and severity is mandatory.

Rules for Commands PV

Commands are orders issued used for example to reset an equipment or to put it "on/off", "in/out" … It basically consists in a write operation, but it is also convenient for OPI to retrieve the list of possible commands, this is achieved with a mbbo record in which possible commands correspond to its ENUM strings. The whole command list can then be retrieved with a caget –d 31 *PVName* and a command can be issued by writing one of the string in the same PV.

Table 3: PV Naming Rule for Command Handling

Operation	PV Name
ReadActual	$(CODIF):Cmd

Rules for status related PV

Status handling consists in read operations related to the states and defects of the system. State and defect words must be analyzed bit per bit, each bit being associated with a string giving its signification. To avoid repetition of the same analysis on OPI side, this analysis is performed on IOC side. The result is provided through the standard interface with a record developed for this purpose. The record is in charge of words analysis and hosts all standard interface compliant PV.

Table 4: PV Hosted by Status Record

Operation	PV Name
Read	$(CODIF):Status.STATE_WORD
Read	$(CODIF):Status.STATE_ON_LIST
Read	$(CODIF):Status.STATE_OFF_LIST
Read	$(CODIF):Status.STATE
Read	$(CODIF):Status.DEFECT_WORD
Read	$(CODIF):Status.DEFECT_LIST
Read	$(CODIF):Status.DEFECT

In the previous table, the xxx_WORD PV contain a word to analyse, xxx_LIST contain a list of strings describing each bit of a word. The STATE PV contains the result of the analysis of the STATE word, when an OFF bit ('0') is found, its corresponding OFF string is copied from the STATE_OFF_LIST, when an ON bit ('1') is found, the string is copied from the STATE_OFF_LIST. The DEFECT PV contains the result of the analysis of the DEFECT word, only ON bits are treated, when an ON bit ('1') is found, its corresponding ON string is copied from the DEFECT_LIST. Each xxx_LIST PV can be retrieved with a caget –d 31 *PVName*

IMPLEMENTATION

This section explains how the standard interface is integrated in EPICS DB.

Consign and Measurements

A read interface PV is handled by an ai or longin record and a written interface PV is handled by an ao or longout record.

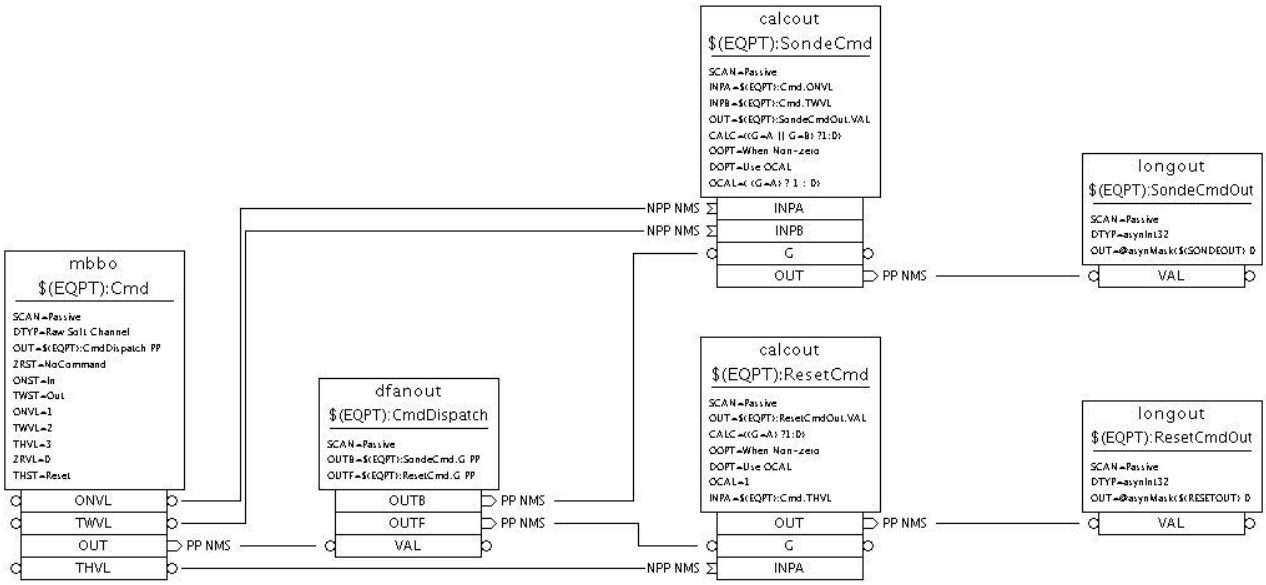

Figure 2: Example of commands handling implementation for an equipment having three commands: "in/out", to be written in the same register and "reset" being written in its own register. When a command is received by the mbbo record, it is transmitted to a dfanout record which dispatches it to two calcout records. The SondeCmd calcout record checks that the command is "in" or "out" if yes the command is actually written via the corresponding longout record, The ResetCmd calcout record checks that the command is "reset" if yes the command is actually written via the corresponding longout record.

Commands

The command interface PV is hosted by a mbbo record, the strings for each individual command are filled in the ZRST, ONST … fields. However, the mbbo record doesn't perform the write operation, this is done by longout record, there are one longout record per command register, depending on the driven equipment. In addition, the path between mbbo record and longout record(s) is secured by calcout record(s) in charge of actually activating or not the longout record(s) and ensuring the validity of what is actually written.

States and Defects

The Status record is in charge of internal state and defect words analysis and interface PV update. The Status record is basically a genSub record in which output fields were renamed and configured in order to comply with the standard interface. The challenging part was to manage each equipment type with its own set of strings through the common to all interface PV. The retained solution consists in defining a version of the Status record specific to each type of equipment, this is simply achieved by creating a copy of the record support structure renamed according to the new equipment and by providing a ".dbd" file in which the equipment strings are defined, in this manner the link between an equipment type and its set of strings is automatically established. Though there are several version of the Status record, they all share the same source code.

At processing time, the record fetches internal state and defect words by dedicated inputs and trigs the processing for each word, if the value has changed. The word analysis process is the same for both state and defect words, a parameter specifying the expected behaviour: "ON" and "OFF" bit values analysis or just "ON" bit value analysis. Strings specified in the ".dbd", basically available in VDCT menus are retrieved with the staticLib functions, and made available to the word analysis process to build the STATUS and DEFECT strings array.

CONCLUSION

We just ended the definition and development of the Standard Interface, so its deployments is still limited to power supply, profiler and faraday cups and the Status Record is not integrated yet. But we used it when setting up and running the control system for deutons source test at Saclay, in a "debug" context it turns out to be really efficient since with the naming convention and the command through mbbo record, driving equipment with simple CA directives in a terminal is a child's play, we can foresee it will ease OPI development. The next step is to integrate the Status record in the power supply, profiler and RF equipments [3]. And soon as possible, all EPICS modules will be made Standard Interface compliant.

REFERENCES

[1] http://www.ganil-spiral2.eu/spiral2-us/keys-sp2/whats-spiral2
[2] D. Touchard "Prel The Spiral2 Control software organization and management" Icalepcs 2009.
[3] D. Touchard "The Spiral2 Radio Frequency Command Control" Icalepcs 2011.