THE ESS CONTROL BOX

E. Laface*, ESS, Lund, Sweden
M. Reščič, Cosylab, Ljubljana, Slovenia

Abstract

The European Spallation Source will be a 5 MW superconducting proton linac, with fixed target, for the production of a stream of neutrons. The entire machine, the target and all the instruments will be controlled by an Integrated Control System: this is a set of hardware and software tools created to provide the most possible easy and flexible interface for the operator daily usage in the control room. The hardware core of the Integrated Control System is the Control Box, a Linux-based computer designed to provide a common platform for the ESS hardware developers. The software front-end for the Control Box is the Experimental Physics and Industrial Control System - EPICS, a standard protocol used to control large facilities such as accelerators or nuclear power plants. In this paper the main characteristics of the Control Box and the EPICS system are presented.

INTRODUCTION

The hardware of ESS will be designed and built in different laboratories all around the world. Those components will be assembled and interact together and will be seen as a unique in the Control Room of the accelerator providing an easy user-end experience.

It is not possible to impose a specific hardware protocol to the producers because the kind of hardware to develop is very different: ion source, superconducting cavities, vacuum pumps, power supply etc. so the idea is to have a flexible device between the low level hardware to drive and the software of the final user as shown in Fig. 1.

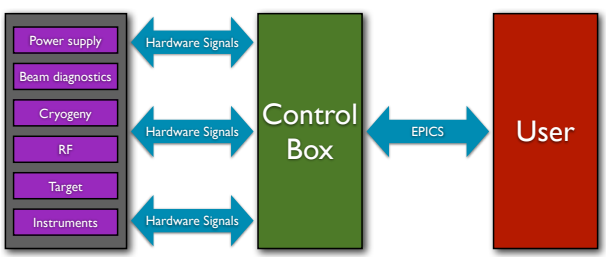

Figure 1: Control Box Scheme.

In this framework the Control Box [3] is a device flexible enough to communicate with the different hardware protocols required by the suppliers and capable to convert it in the Experimental Physics and Industrial Control System (EPICS) [5] signals. In order to serve as such, many things need to be taken into consideration, such as:

- user friendliness;
- robustness together with a generic approach;
- documentation and sample applications.

Although all of the above is considered as given when it comes to software / hardware development the majority of development effort is put to meet all these requirements.

FRAMEWORK

The two main aspects considering in the design of the prototype are:

- hardware platform and components;
- operating system and prototype applications.

The first step was to decide on hardware platform and components. Currently, there are many available platforms (cPCI, VME, VXS, PXI, XTCA, etc.) but the main priority was the time to completion of the prototype. This constraint restrict the choice, meaning the hardware should be available soon and should also present a wide selection of components (analog and digital cards, CPU boards and a selection from different vendors). The ESS Control System Framework, EPICS, also played a major role in the selection process. The final decision was to go with Compact PCI (cPCI) [2] and the vendor Adlink [4], because of the availability of the hardware components and support for the selected operating system.

Linux as the operating system was a natural choice because it is the OS selected for the Control System at the ESS. The open discussion was around the choice of the distribution and the final decision was to go with Scientific Linux [8] because this made many of the Control, Data Access and Communication (CODAC) [7] packages available. Adlink had already some Linux support for their components.

Finally, for the prototype applications development, was selected the Linux Control and Measurement Device Interface (COMEDI) [6]: a collection of drivers for a variety of common data acquisition plug-in boards. This approach will allow great flexibility when writing support for new components, since much of the work can be reused.

* Emanuele.Laface@esss.se

Figure 2: The Control Box.

HARDWARE

ADLINK cPCI-3965

This is the computer itself and it is a 3U CompactPCI (cPCI) single board computer with Core2 Duo processor (2.2 GHz). Front panel I/O includes VGA output, two Gigabit Ethernet ports and two USB ports.

ADLINK cPCI-7230

ADLINK's cPCI-7230 is a 16 channels isolated digital input and 16 channels isolated digital output card which provides a 5000 V optical isolation protection between the host computer and the I/O ports. The wide input range of the card makes it easy to sense the status of external devices. The non-polarity characteristic is suitable for a wide variety of industry applications. The cPCI-7230 device also features a wide output range from 5 to 35 V, which is suitable for relay driving and industrial automation applications.

ADLINK cPCI-6216V-GL

ADLINK's cPCI-6216V-GL is a 16 channels, 16-bit, analog output card. The card offers 8 voltage outputs with 10 V range, featuring 15-bit monotonicity and 25 V/μs slew rate. The onboard analog switches minimize the power-on glitches. The cPCI-6216V-GL expands the voltage output channels to a total of 16 for higher analog output density requirements. This card provide high-resolution, high-density analog output functionalities and is suitable for ATE, signal generation, industrial process control, servo control and other industrial control applications. It also provides four digital input and four digital output channels.

ADLINK cPCI-9116

ADLINKs cPCI-9116 card is high-density and high-resolution multi-function DAQ cards for PXI/CompactPCI form factors. The devices can sample up to 64 AI channels with different gain settings and scan sequences, making them ideal for dealing with high-density analog signals with various input ranges and sampling speeds. The cPCI-9116 device features flexible configurations on analog inputs. It provides analog inputs with 4 programmable input ranges for both bipolar and unipolar inputs. The A/D on the cPCI-9116 device features a sampling rate of up to 250 kS/s with resolution at 16 bits. These device also offers differential mode for 32 AI channels in order to achieve maximum noise elimination. The cPCI-9116 card also has 1 channel 16-bit general purpose timer/counter, 8 channels TTL digital inputs, and 8 channels TTL digital outputs.

SOFTWARE

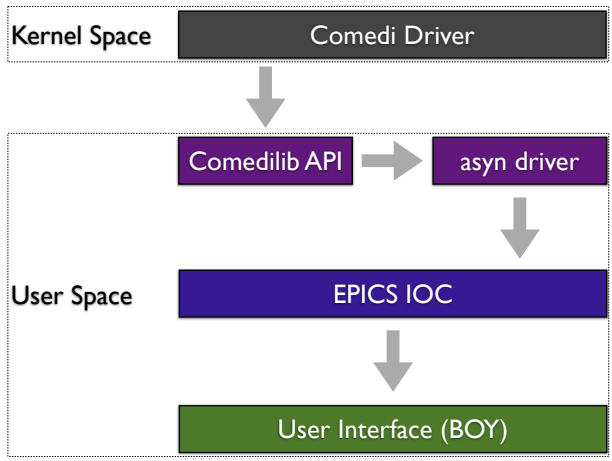

Figure 3: Control Box Software Concept.

The chain in Fig. 3 shows how the information arrives from the Linux kernel to the user: the signal is captured at low level with the Comedi driver and forwarded to the EPICS layer through a set of libraries. The user can access the EPICS signal trough any EPICS front-end software. For prototyping it was selected Control System Studio (CSS) [1] that allows to create graphical interfaces without any knowledge of programming. This gives the possibility to have a large set of beta-testers.

The software implementation was the main issue of the Control Box prototype. During this process several aspects were analyzed to improve the standardization of the Control Box for a future stable release.

Development and Deployment Environments

A crucial aspect was the difference between development and deployment environments. The first release of the Control Box was developed on a different Linux distribution with respect to Scientific Linux and, at the moment to recompile and install everything on the Control Box, the system did not work requiring various changes to adapt the configuration to the deployment system. These problems were solved unifying the environments, this choice become a requirements for the production of the Control Boxes.

Binary Package Distribution and Installation

Another important topic for the software development is the way to transfer and maintain the software updated. It is fundamental to distribute the software from only one

ISBN 978-3-95450-115-1

source without compiling the code on the deployment machine. From this point of view the Linux experience is of great help: it was created a RPM repository with the Control Box software pre-compiled. This repository is in the list of the repositories in the Linux configuration and the installation of the components is done trough the YUM package manager [10].

Consistency

Although Linux distribution versions do not change rapidly, the kernel version does, together with all the packages and libraries that the application may depend on. Since the COMEDI drivers are compiled as kernel modules, they depend on the kernel source code. This became an issue when the developer's kernel version and the user's kernel version were not in sync (user was performing automatic updates whereas the developer was not). To reduce the risk of dependency issues, the version of the kernel and the core software are changed according to the development requirement and not the usual Scientific Linux update.

Interface

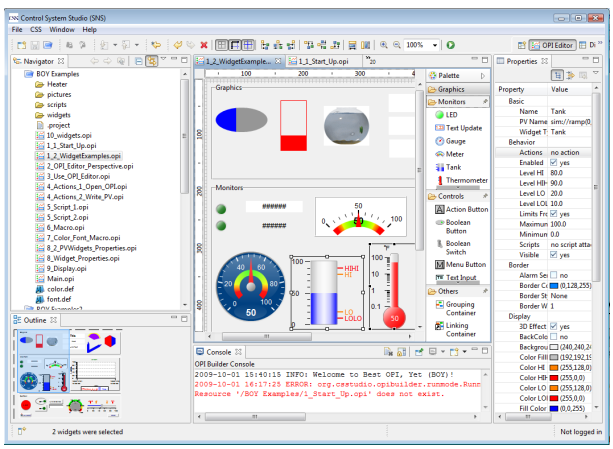

Figure 4: CSS editor example.

The front-end interface can be any tool capable to send and receive EPICS signals. This give a large flexibility to the developer because the EPICS API are available for any platform and many codes are already on the market. In the prototyping phase CSS is the main tool, but for the operations the Control Box will be controlled trough the OpenXAL [9]. This is a suite of Python, Java and XML tools to manage large accelerators. This code is based on XAL, the code used to control the Spallation Neutron Source in Oak Ridge. Another considered interface is the one for mobile devices. A basic prototype was already tested on iOS and worked to read EPICS signals. The mobile computing is growing rapidly and to have the possibility to monitor the Control Box trough a mobile device like a tablet PC can be of great value in the commissioning phase.

DOCUMENTATION

The third main component of the Control Box experience, after the hardware and the software, is the documentation. The most advanced tools are useless if the user is not able to understand what he can do. For this reason several documentation tools are available and constantly updated during the development process. The main resource for the Control Box is the Wiki page [3]. The core of the documentation is represented by many examples available: the possibility to start with a working example reduces dramatically the training time.

CONCLUSIONS

The whole process of prototyping was longer than planned and it took more effort as estimated. We expected such outcome, since this is a preliminary prototype work and many unknowns are encountered. More importantly, the experience and lessons learned from this prototype will provide good feedback into future prototyping work.

The decision to provide various stakeholders (RF, Beam Instrumentation, Neutron Instruments, etc.) with prepackaged Control Boxes, together with all required hardware drivers, development tools, configuration scripts and a standalone Development Environment puts a lot of risk on the Controls group side. The end-user should be provided with a working set of tools that he requires (e.g. custom hardware component drivers) but also, at the same time, the Control Box should include all the services and additional software and hardware that Controls group and the EPICS may require (Control System Service, e.g. archiving, alarm services, timing receivers).

The prototyping process is still improving with a trial and error process based on the feedback of the users. The development of a stable and reliable Control Box is the base for the whole control system of the ESS: any effort invested in the improvement of this project is an investment in the future of the accelerator.

REFERENCES

[1] Control System Studio: http://css.desy.de

[2] Wiki CompactPCI: http://en.wikipedia.org/wiki/CompactPCI

[3] Wiki Control Box: https://twiki.esss.dk/ad/index.php/The_Control_Box

[4] Adlink: http://www.adlinktech.com

[5] EPICS Web Site: www.aps.anl.gov/epics

[6] Comedy: http://www.comedi.org

[7] CODAC: http://www.iter.org/org/team/chd/cid/codac

[8] Scientific Linux: http://www.scientificlinux.org/

[9] OpenXAL: http://xaldev.sourceforge.net

[10] YUM: http://yum.baseurl.org