# LINUX/PCI: THE ESRF BEAMLINE CONTROL SYSTEM MODERNISATION

A. Homs-Purón, D. Beltrán, A. Beteva, M. C. Domínguez, P. Fajardo, A. Götz, J. Klora, E. Papillon, M. Pérez, V. Rey, ESRF, BP 220, F-38043 Grenoble Cedex, France

*Abstract*

The ESRF control system was developed more than 10 years ago, using VME Motorola 68000 CPUs with OS/9, connected to control HP/Sun workstations (WS) through TACO. Needs of faster experiments and change to supported hardware require the modernisation of beamline (BL) control. Industrial PCs with Pentium III CPUs running Linux have been chosen as main control PCI crates. Support of current VME instrumentation is ensured through PCI/VME bus couplers; cPCI crates can also be controlled by PCI/cPCI bus extenders. TACO/TANGO is used to export the instrumentation control through the network. This system is very flexible in terms of configurations; it can control a simple BL from a single PC, and it can fit more complex experiments with dedicated PCI/cPCI crates controlled by a client WS. Fast acquisitions are implemented with a soft real time, kernel-based mechanism called *Hook*, which allows simultaneous reading of PCI, cPCI and VME cards. SPEC is used as main control application, now featuring a server mode that allows PyQt GUIs to easily access the hardware from a higher level.

## INTRODUCTION

The ESRF beamline (BL) control system is based on VME Motorola 68000 (M68K) CPUs running OS/9, controlled by HP/Sun workstations through TACO, a network-based client/server environment developed in-house for the machine control. Such a system, based on the state-of-the-art technology more than ten years ago, needs to be upgraded to fulfil the demand of much faster experiments and the integration of modern hardware.

## HARDWARE IMPLEMENTATION

The evolution of the BL control system includes the upgrade of both the hardware and software architectures. Industrial PCs, featuring high performance Intel x86 workstations, have been chosen as the main control platform due to their strong support and the high availability on the market of PCI-based devices. This environment provides very interesting cost/performance ratios. The compatibility with the current instrumentation is ensured through the use of commercial PCI/VME bus couplers, allowing the smooth introduction of the system into the BLs. The fiber optic link of up to 35 m long used by these bus couplers permits the control of distributed crates. This, together with the possibility of controlling several crates from the same PC, simplifies its introduction in the BLs and reduces the costs of the transition. The system also supports the industrial cPCI platform, either as stand alone systems or as slave crates of the PCI workstations by means of PCI/cPCI bus extenders.

One of the more important features of this architecture is its flexibility: it does not require the network for simple and time-critical applications, and, at the same time it supports distributed system for more complex configurations.

## SOFTWARE STRUCTURE

### *Operating system and drivers*

Linux has been selected as the control operating system because it is stable, open and free. It offers a strong developing community and a growing number of companies supporting it, resulting in an increasing amount of hardware that it can control. In addition, the software maintenance is simplified because of the unification to the other UNIX control operating systems at the ESRF.

The Linux drivers of the VME boards offer, in general, more functionality than the obsolete OS/9 implementation, and they can deal with a disconnection of the VME crate, allowing easy hardware maintenance. This is done through a common interface, developed to keep transparent the real mechanism to access the VME bus [1]. As a result, the same driver codes works for Intel x86 with bus coupler, M68K VME and PPC VME; the porting to Intel x86 VME is in progress.

Driver configuration and start-up tools notably simplify installation and maintenance. A GUI-based application manages the PCI, cPCI and VME devices, including bus couplers/extenders. For PCI/cPCI, the system keeps track of the slots on which the cards are connected, based on a database of known crates and back-planes. This mechanism overcomes the problem associated to PCI board enumeration, which causes the identification number of board to change when another of the same type is added/removed in a slot closer to the PCI host. In the case of VME bus couplers, this changes the crate number; in cPCI bus extenders it changes the sub-bus enumeration.

### *Middleware*

The next layer in the software architecture is TACO/TANGO, which basically exports the hardware functionality over the network, and at the same time, encapsulates the device in an intelligent remote object. It's implemented through device servers, originally developed in Object In C (OS/9), and later ported to C++. A new TACO framework that exploits the C++ capabilities provides a higher device abstraction and a uniform interface for developers, including high throughput debug, start-up/clean-up, compound (complex) devices, etc. For instance, all VME devices inherit the functionality for automatic reconfiguration when the VME crate is disconnected, making such event transparent for the higher layers in most of the cases. Such an interface will simplify the transition to the new TANGO architecture, completely based on the object-oriented paradigm [2].

### *Control application layer*

The main control application in the system is SPEC, which has proved a very flexible control of an appreciable amount of hardware. One important improvement is its server mode, which implements efficient and robust inter-session communication via network sockets. In order to implement this consistently, the pseudo devices, which are not controlled internally by SPEC but in the macro level, must follow a new interface that permits SPEC to treat them almost like the devices supported in the C code. This feature allows modularised configurations in complex BL set-ups by converting some SPEC sessions in clients of others. Following the same interface, a higher control layer, such as Graphical User Interfaces, can be implemented as SPEC clients. This higher layer is being implemented on Python/Qt, a strongly supported, portable and user-friendly platform. Few GUIs that prove the stability of this interface have been already designed; the current trend is to develop a general GUI common for all BLs.

An important set of tools for BL control is also developed on Python/Qt, including data visualisation and analysis modules, BL configuration managers, software installers, among others.

## PERFORMANCE

As mentioned before, a key feature of the system is its scalability, being able to run in a single workstation as well as on multiple PCI/cPCI crates connected through the network. In the first case, a single access to the hardware through all the control layers is reduced to 50 μs, while the use of the network raises the single access time to 150 μs. These values are notably lower than their counterpart on the old OS/9-M68K-10baseT of 3 ms.

The Linux control system also provides an interface for fast software-based acquisitions, one of the main goals of this modernisation project. This mechanism, called *Hook* after its OS/9 implementation, "hooks" into hardware-related events to trigger synchronous data acquisition. It is implemented in the kernel layer and connects, at run time, a specified event with the reading of hardware cards. The triggering event can be a device interrupt or a software system call. On the event generation, the configured hardware "channels" are read and their contents are stored in a kernel buffer. A *Hook* TACO device server continuously transfers the data to a user-space buffer and, when requested, to the client. Since the configuration of the trigger event and the hardware channels to read is done at run-time, this a software triggering notably simplifies the instrumentation cabling when a good synchronisation is desired, important in multi-purpose, flexible BLs. Acquisition rates of 10,000 points/sec are normally achieved with this system, reading, simultaneously, VME, PCI and cPCI cards. The channel synchronisation in *Hook* acquisitions is fixed by the single hardware access time of 1 μs for PCI/cPCI and 3 μs for VME. It is worth to say that several *Hook* devices are available in the system, allowing multiple acquisitions to take place at the same time.

The next goal in performance terms is the use of dual CPUs control systems. The chosen industrial PCs support this configuration, and the extra CPU cost is well worth compared with the expected performance. This resource increase will break the bottleneck formed when high-level applications (like online plotting) run on the same PC at acquisition rates higher than 10 KHz with VME interrupt generators.

## CONCLUSIONS

As a conclusion, the system allows the modernisation of the ESRF BL control, supporting the current VME instrumentation as well as the new PCI and cPCI platforms. Its ability to coexist with the OS/9-M68K control system permits its gradual integration on the BLs. At the time of writing this paper, the ESRF BLs BM16, ID23 and ID31 are completely based on the Linux control system, while the BLs ID11, ID22 and ID32 have introduced Linux PCs to substitute OS/9-M68K crates. Several BLs (ID9, ID19, ID21) are in progress of introducing the Linux PCs in their control systems, and new BLs are expected to install it from the start.

## ACKNOWLEDGEMENTS

This work has been developed jointly by the BLISS and the ISG-Electronics. Special thanks to R. Hino for his support.

## REFERENCES

[1] A. Götz, A. Homs, B. Regad, M. Perez, P. Mäkijärvi, W-D. Klotz, "Modernising the ESRF Control System with GNU/Linux", ICALEPCS 2001, San Jose, November 2001.

[2] A. Götz, J. Meyer, E. Taurel, W-D. Klotz, "TANGO - A Object Oriented Control System based on CORBA", ICALEPCS'99, Trieste, October 1999.