HYPPIE: A HYPERVISORED PXI FOR PHYSICS INSTRUMENTATION UNDER EPICS

James Rezende Piton, Márcio Paduan Donadio, Diego de Oliveira Omitto, Marco Antonio Raulik and Harry Westfahl Jr., LNLS, Caixa Postal 6192, Campinas – 13083-970, Brazil
Bruno C. Yenikomochian, National Instruments, Av. Paulista, 509, São Paulo – 01311-910, Brazil

Abstract

Brazilian Synchrotron Light Laboratory (LNLS) has a 1.37 GeV source open to scientific community since 1997. Recently, the control system of its beamlines, originally designed within a proprietary Delphi/Windows platform, is going through an upgrade to the open source EPICS/Linux platform. Within this upgrade strategy, the use of off-the-shelf hardware was also considered an alternative to the original in-house developed equipment, while keeping the EPICS/Linux compatibility. A PXI chassis and its modules were made available to EPICS through the NI Real-Time Hypervisor virtualization system that allows running simultaneously EPICS/Linux and LabView Real-Time in the same PXI controller, sharing a common memory block as their communication interface. Hyppie is the data exchange protocol developed in a collaboration between LNLS and National Instruments (NI) to implement motor, scaler and binary in/out EPICS records and channel access in the Linux layer, leaving the low-level hardware control to the LabView RT layer. This solution was tested to control an X-ray absorption spectroscopy beamline, showing stability and reduction of counting dead-time and software development time for integrating new hardware.

EPICS IN A NEW CONTROL SYSTEM FOR THE LNLS BEAMLINES

LNLS has projected, developed and built a large number of components for its accelerator complex and beamlines. As a National Laboratory under the Brazilian Ministry of Science, Technology and Innovation, one of LNLS primary goals is to develop skilled human resources in several technological fields. Back in the 1990s, severe budget constraints and industrial policies in Brazil at that time led to the decision to create at LNLS also its own hardware platform for data acquisition, LOCO [1]. For 20 years, interface boards have been evolved and produced at LNLS, applied to its control system and also transferred to industrial applications.

Nevertheless, such a program of development and specially maintenance of electronics made "at home" needs a lot of well-trained technicians, not always available in a local demanding job market. A process of upgrade of the beamlines allowed to start an internal discussion on alternatives to the control, while a new concept of LNLS hardware is in project for Sirius, the new 3GeV machine to be built at LNLS. Moreover, a recent project for remote access of the beamlines demanded the use of a distributed control system with APIs ready to be integrated in Java. EPICS was the choice as the middleware, for its long development, open-source features, covered hardware and vast collaborative community. But the hardware for the scientific facilities still matters, due to the budget and choices in the local market. In this project, hardware produced by National Instruments was chosen as it achieves the specifications required and the manufacturer has offices in Brazil. Nevertheless, it still should fit in the project of adoption of the distributed control system, EPICS. Therefore, in a collaborative effort, LNLS Beamline Software Group and National Instruments Brazil conjointly pursued a solution.

PROJECT HYPPIE

Hyppie is a project created by LNLS and NI Brazil to make a bridge between EPICS records and corresponding devices in the PXI chassis. Real-Time Hypervisor for Linux uses virtualization technology to run both Red Hat-based Linux and NI LabView RT in parallel on multicore PXI controllers. I/O devices, RAM and CPU cores are partitioned between both OS.

A standard EPICS distribution is installed in Linux and new hardware drivers can be written in LabView. Shared memory provided by Hypervisor is allocated by Hyppie as a map to deal with each piece of hardware, through commands, parameters and readouts. EPICS IOCs running in the Linux side refer to the shared memory instead of any direct I/O access.

Hyppie already supports EPICS binary-in/out, analog-in/out, scaler and motor records. Furthermore, the EPICS installation can have IOCs to normally access other remote devices on the network. In Linux, IOCs device support [2] is implemented and the communication with each device is done reading from and writing to the corresponding shared memory block, accessed simultaneously by VIs running in LabView RT.

Whenever a new device needs to be inserted into Hyppie, the device operation is programmed in LabView, and runs in LabView RT. The complexity of the protocol to communicate to the devices is attenuated by the simplicity of LabView programming. This can increase development productivity.

LabView RT

In the LabView RT side, there are 3 big groups: a config file, a VI to read the config file and VIs for each record type. The config file is a textual description of every module to be dealt in both sides. Through a *RecordName* section specifying the kind of record (e.g. PARKER_MOTORS, IMS_MOTORS or SCALERS), a block identifier *SharedMemoryName* with its *Parameters*

ISBN 978-3-95450-121-2

Copyright © 2012 CC-BY-3.0 and by the respective authors

defines the memory allocation and the way the data in that shared memory will be interpreted. Parameters can describe the serial port used, a timing factor, a device address or a board identification.

The VI responsible for reading the config file parses each record block and parameters and organize them in arrays that are linked appropriately to the VI of each record type.

IOCs in the Linux side

To simplify some work that must be done in each IOC, a common library was created to put together header files and binaries of the Hypervisor system API. That *PXIcommonlib* provides some useful functions:

- *pxiCreateSM*: function available in the IOC shell. It is used to connect to shared memories and get the necessary handle numbers.
- *pxiSearchSM*: record device support reads OUT or INP fields in order to get shared memory names and use *pxiSearchSM*() to recover the handle number.
- *CHECK_AND_THROW*: macro that prints messages on the return value from shared memory library functions, when an error is raised.
- *pxidebugMsg*: prints some messages useful for application debugging.

Using this common library, all the device support needs to do is to use *VLXSMReadBlock* and *VLXSMWriteBlock* API functions to communicate with programs running in LabView. VIs implementing a record type has a structure as shown in Figure 1.

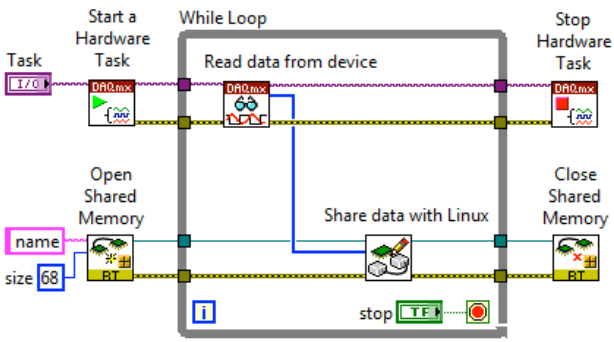

Figure 1: Logical structure of a VI implementing a Hyppie record type.

AVAILABLE RESOURCES IN HYPPIE

At present, Hyppie counts on the resources needed to produce an Exafs experiment at LNLS XAFS1 beamline: counters, motors and digital I/O. This beamline was recently refurbished and has been used as a workbench to test the new concept.

Binary I/O

DigitalPXI is the software module in Hyppie to manage digital I/O interfaces. The IOC structure and device support were derived from instructions and templates from EPICS base R3.14.11.

The software running in LabView RT initializes the boards and assigns the shared memory block. The values of input channels are continuously read and transferred to the memory (binary-in) and, conversely, all output are updated from the shared memory (binary-out). Binary-in and binary-out have 17 fields of 4-byte integer each.

Acting as a watchdog, a counter in the first field of the shared memory block is incremented to indicate that the hardware is being used. If the counter doesn't increase, the record alarm is set, showing the values are not updated. The binary-in fields can be seen in Table 1. Symmetrically, the fields of a binary-out record are read by RT and written by Linux.

Table 1: Shared Memory Allocated for Binary-In Record

Field	Description	Read	Write
0	Counter indicating OK	Linux	RT
1	Input value 0	Linux	RT
...	...	...	...
16	Input value 15	Linux	RT

Counters

The ScalerPXI is based on module std-2-8 from EPICS synApps-5-5, with support to Scaler Record v3.19. This record holds 64 counting channels, with the first one (S1) being the clock counting under the selected hardware frequency (FREQ). Currently, the implemented code in the LabView RT side makes available 4 channels, what is enough for Exafs experiments. The LabView RT side determines how many hardware channels are available to the Linux side through a shared memory block field.

There are 3 data acquisition modes:

a) Pulse counting of low voltage signals
b) Integration of a voltage signal
c) TTL-pulse counting

In the first two modes, LabView RT gets readings from the dynamic signal acquisition module PXI-4462 and a digital I/O counter PXI-6602 is used in the other mode. The way the shared memory works is the same for all modes.

The corresponding VI in LabView RT waits for commands from the ScalerPXI IOC through the shared memory. Only two commands are needed to implement Hyppie Scaler Record: one for triggering and other for halting, if necessary. Previously to every counting trigger, the IOC writes into the shared memory the counting time, given in tenths of microseconds. The default frequency is 10MHz.

The balance between time resolution and processing cost has been optimized by choosing 2 ms as the time window for hardware integration. Smaller amounts of time are unnecessary for Exafs experiments at LNLS.

Copyright © 2012 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-121-2

The VI is compatible with a preset-counter operation. If a channel has its gate field active, a preset will be used to determine the end of counting.

Table 2 shows the fields of the shared memory, each one a four-byte integer. The IOC scaler 1 (clock counting) is defined as a counter in the shared memory, being the hardware counting channels (scalers 2 to 64) defined as channels 1 to 63.

Table 2: Shared Memory Allocated for a Scaler Record

Field	Description	Read	Write
0	Command (start/stop counting)	RT	Linux
1	Status (0:idle/1:counting/2:hw error)	Linux	RT
2	Number of channels (1 to 63; default: 4)	Linux	RT
3	Frequency in Hz (1 to 4.2 GHz; default: 1E7)	RT	Linux
4	Counter value (on freq.)	Linux	RT
5	Channel 1 value (counts)	Linux	RT
...	...	...	...
67	Channel 63 value (counts)	Linux	RT
68	Counter gate control	RT	Linux
...	...	...	...
133	Channel 1 Preset	RT	Linux
...	...	...	...
195	Channel 63 Preset	RT	Linux

Motors

A motor record uses a shared memory block of 18 fields, each with 4 bytes in size. Some fields are shown in Table 3. Using this fixed structure, Hyppie Motor Record device support does not have to be changed when implementing a new motor family.

Table 3: Shared Memory Allocated for a Motor Record

Field	Description	Read	Write
0	Command (see *Table 4*)	RT	Linux
1	Error (3 bytes)	Linux	RT
2	Absolute movement (steps)	RT	Linux
3	Relative movement (steps)	RT	Linux
4	New desired position (steps)	RT	Linux
...	...	...	...

Implementation of device support is based on the commands provided by motor record support, using the function *start_trans* [3, 4]. Table 4 lists some command codes transmitted from the IOC in Linux to the LabView RT program.

Table 4: Command Codes for the Hyppie motor Record

Command	Description
1	Move to absolute position
2	Move to relative position
3	Go to home in forward direction
4	Go to home in reverse direction
5	Change desired position
6	Change base velocity
...	...
20	Read limit /home switches
21	Read movement status
22	Read position

At the present time, IMS MDrive and Parker OEM 750 are working in Hyppie Motor Record.

CONCLUSION

Hyppie is proving to offer the advantages it is designed for. A wide array of PXI instrumentation modules is available, as PXI is a standard common to a variety of manufacturers. In addition, Hyppie benefits from the ease of programming (LabView) and the use of libraries provided by the hardware vendors. Tests performed in the LNLS XAFS1 beamline shown gain in signal to noise ratio and reduction in the acquisition dead-time, with the new system of hardware and software [5]. Further implementation is in course to include more features like aSyn records and CCD detectors. That will meet the needs of other LNLS beamlines to be refurbished in the next months. Source code and documentation of this open-source project will be soon available at: *http://www.lnls.br/sol/hyppie*

REFERENCES

[1] P. F. Tavares et al., "LNLS Control System" EPAC'96, Sitges, June 1996, TUP062L; http://www.JACoW.org
[2] M. R. Kraimer et al., "EPICS Application Developer's Guide - EPICS Base Release 3.14.11", February 2010.
[3] M. Davidsaver and D. Chabot, "Motor Record Device Support" v. 1, February 2010; http://pubweb.bnl.gov/~mdavidsaver/epics-doc/epics-motor.html
[4] T. Mooney, J. Sullivan and R. Sluiter, "Motor Record and Related Software", June 2003; http://www.slac.stanford.edu/grp/cd/soft/epics/site/motor
[5] D. C. Coelho et al., "The upgrade of XAFS1 beamline at the Brazilian Synchrotron", SRI 2012, Lyon, TH-M-P-28.

ISBN 978-3-95450-121-2

Copyright © 2012 CC-BY-3.0 and by the respective authors