A STATUS UPDATE ON HYPPIE: A HYPERVISORED PXI FOR PHYSICS INSTRUMENTATION UNDER EPICS

James Rezende Piton[1], Márcio Paduan Donadio[2], Diego de Oliveira Omitto[3] and Marco Antonio Raulik[4], Beamline Software Group/LNLS, Caixa Postal 6192, Campinas – 13083-970, Brazil.

Abstract

Brazilian Synchrotron Light Laboratory (LNLS) has a 1.37 GeV source open to scientific community since 1997. Since 2012 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. This data exchange protocol is called Hyppie. It is ready for some 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. Nine LNLS beamlines are presently running under this system and more beamlines will move to this in the months to come.

INTRODUCTION

Hyppie [1] is a project created by LNLS and National Instruments Brazil to make a bridge between EPICS records and corresponding devices in a 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. In Linux, IOCs device support 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. The hardware was selected not only for the application requirements but also because there are offices and support from the manufacturer in Brazil, an important point for such a long-term decision.

During the time schedule to move the LNLS beamlines into the new control system concept EPICS/Hyppie, additional features are being continuously included into the project, according to specific characteristics of each beamline. Priority is given to implementation which covers a larger number of beamlines with a given need or similar characteristics.

This new approach of distributed control system for the LNLS beamlines opens new possibilities, like remote operation.

[1] james@lnls.br
[2] mdonadio@lnls.br
[3] diego.omitto@lnls.br
[4] raulik@lnls.br

HYPPIE SYSTEM

Since its creation, Hyppie was conceived to enable faster development cycles and to bring new hardware interfaces available under EPICS. Some of those new hardware categories will be presented here. Presently, Hyppie supports EPICS binary-in/out, analog- in, scaler and motor records, Asyn driver and area detector. Nine beamlines were or are submitted to a refurbishment and are running with control hardware under EPICS through Hyppie: XAFS1 and XAFS2 (X-ray absorption spectroscopy), XRF (X-ray fluorescence), XRD1 (X-ray diffraction), SAXS1 and SAXS2 (small angle X-ray scattering), MX2 (macromolecular crystallography), PGM (ultra violet with a planar grating monochromator) and IMX (X-Ray tomography). There is a schedule for the remaining beamlines to move to this control system.

Scaler Record

A considerable amount of features has been added to the new Scaler Record in the Hyppie system. Core changes in the Real-Time software allowed the integration of signals read from different cards to be available in a same Scaler Record. It also made it possible to share the same signals for being integrated independently by two different PVs. A real-time embedded industrial controller was added to the system.

The new structure of the Real-Time software has two layers. The first one acts as a hardware server and consists of LabVIEW VIs making the integration of the signals for each card. The integration is done every 1 ms. The result is published to the second layer. It consists of 2 LabVIEW VIs. Each VI provides the Scaler functionality to its respective Scaler Record through shared memory. The purpose of having 2 Scaler Records is to let one EPICS client operate the Scaler in autocount mode for checking the counting regardless of the experiment status.

Futhermore, having different signal sources on different cards integrated in the same Scaler Record made it possible to use Gate Control (field Gn) between them. The same was not possible if these signals were read through different Scaler Records. It also improved trigger synchronization as it is now done in a lower level inside RT instead of sending the trigger for each PV through channel access.

The current system supports a industrial chassis CompactRIO NI 9144, in which a 4-channel analog input module NI 9215 is being deployed. This new chassis is being installed inside the experimental hutch, reducing the size of signal cables and consequently reducing

ISBN 978-3-95450-139-7

eletromagnetic interference. The communication between PXI chassis and the CompactRIO is made using EtherCAT through a dedicated network card in the PXI. The four channels of the AD are sampled simultaneously at a rate of 100kS/s. This model of CompactRIO has an integrated FPGA, which does the sampling and applies a Butterworth low-pass filter to the signal before publishing it to the second layer of the scaler software in Real-Time.

Serial Port Access

Due to a Linux version limitation, it's not possible to access the PXI-8432/4 and PXI-8433/4 serial port boards using the Hypervisor system nowadays. On the other hand, it's very simple to use these boards in LabVIEW. It was needed a way to make the IOCs running in Linux access the serial boards in RT. The solution was to develop a driver using EPICS Asyn[2] structure, which was called Asyn Hyppie.

Asyn is an EPICS driver that can access generic interfaces such as serial ports, GPIB ports, TCP/IP, etc[3]. It hides the complexity of the read/write/configure operations, transforming it in common C functions, independently of the interface type.

In this case, the implemented driver access the shared memory between Linux and LabVIEW RT and sees it as an interface. To better comprehension, see the shared memory structure in table 1.

Table 1: Allocation of 2328 bytes of shared memory for the Asyn Hyppie driver

Field	Description	Read	Write
0	Command (1 byte)	RT	Linux
1	Response (1 byte)	Linux	RT
2	Asyn type (1 byte)	RT	Linux
3	COM port (5 bytes)	RT	Linux
4	Baud rate (4 bytes)	RT	Linux
5	Data bits (2 bytes)	RT	Linux
6	Stop bits (2 bytes)	RT	Linux
7	Parity (1 byte)	RT	Linux
8	Flow control (1 byte)	RT	Linux
9	Timeout (4 bytes)	RT	Linux
10	Read buffer data size (4 bytes)	Linux	RT
11	Read buffer (1020 bytes)	Linux	RT
12	Write buffer data size (4 bytes)	RT	Linux
13	Write buffer (1020 bytes)	RT	Linux

The listed fields in the table are expected as follows:

Command: can be values meaning “Configure Serial Port”, “Close Connection”, “Write to Serial Port”, “Read from Serial Port” or “Flush Serial Buffers”.

Response: used to report errors to the IOC.

Asyn type: 0 for RS 232 or 1 for RS 485.

COM port: a text like “COM2”, “COM6”, “COM20”, indicating the used PXI COM port.

Baud rate: a number with the baud rate value.

Data bits: a number with the amount of data bits.

Stop bits:

- 10 for 1 stop bit,
- 15 for 1.5 stop bit and
- 20 for 2 stop bits.

Parity:

- 0 for none,
- 1 for odd,
- 2 for even,
- 3 for mark and
- 4 for space

Flow control:

- 0 for none,
- 1 for XON/XOFF,
- 2 for RTS/CTS,
- 3 for XON/XOFF and RTS/CTS,
- 4 for DTR/DSR,
- 5 for XON/XOFF and DTR/DSR

Timeout: time, in milliseconds to wait for data

Read buffer data size: VI in LabVIEW informs the IOC how many bytes are available to read from the read buffer

Read buffer: raw data read from the serial port

Write buffer data size: the IOC informs the VI in LabVIEW the number of bytes available to write from the write buffer

Write buffer: raw data that must be sent to the serial port

GigE Vision and Firewire Cameras

Cameras are a new Hyppie resource, due to the use of those devices at some beamlines in LNLS, especially at IMX (X-Ray tomography) beamline. The support for the IEEE-1394 (firewire) and GigE Vision standards in the corresponding National Instruments software library for image processing led to a structure of shared memory registers in Hyppie to cover both types of camera, provided that the gigabit specifications match those of AIA (Automated Imaging Association). The use of this resource has been thoroughly tested during the comissioning of a microfocus KB mirror system in the XRF beamline, where one Allied Vision Technologies firewire camera model F-146B is used in tests.

As the Real-Time system should not be occupied by file outputs, the TIFF image is transferred to the IOC in the Linux side, which performs the writing to a storage system. A large number of functions in the corresponding LabVIEW image processing library is available in the real-time side. Thus, some calculation on the beam image is an additional feature, performed at a high rate.

The parameters and data in the shared memory are shown in Table 2.

Table 2: Allocation of 6 Mbytes of Shared Memory for the Camera Hyppie Driver

Field	Description	Read	Write
0	Watchdog (4 bytes)	Linux	RT
1	Acquire (1 byte)	both	both
2	Number of images (4 bytes)	RT	Linux
3	X offset (4 bytes)	RT	Linux
4	Y offset (4 bytes)	RT	Linux
5	Width (4 bytes)	RT	Linux
6	Height (4 bytes)	RT	Linux
7	Command for the camera (128 bytes)	RT	Linux
8	Response from the camera (128 bytes)	Linux	RT
9	File ready (1 byte)	both	both
10	Image number (4 bytes)	Linux	RT
11	File size (4 bytes)	Linux	RT
12	Gain (4 bytes)	RT	Linux
13	Centroid X (4 bytes)	Linux	RT
14	Centroid Y (4 bytes)	Linux	RT
15	X-profile center (4 bytes)	Linux	RT
16	X-profile amplitude (4 bytes)	Linux	RT
17	X-profile FWHM (4 bytes)	Linux	RT
15	Y-profile center (4 bytes)	Linux	RT
16	Y-profile amplitude (4 bytes)	Linux	RT
17	Y-profile FWHM (4 bytes)	Linux	RT
18	Turn on/off profile calculation	RT	Linux
19	X-profile array (18 Kbytes)	Linux	RT
20	Y-profile array (18 Kbytes)	Linux	RT
21	Image file bytes (> 5Mbytes)	Linux	RT

In a future version, more PVs will be created to extend the configurable parameters and the IOC will be modified to exploit EPICS areaDetector features like NeXus plugins. Also, a configuration database will be created to allow a simple way of adding new camera models, as different models of a manufacturer can have different commands for the same function.

CONCLUSION

Through Hyppie a number of PXI instrumentation modules got available to LNLS, as PXI is a standard common to a variety of manufacturers. Also, Hyppie improved the process of getting EPICS as the base of a new distributed control system. The benefits range from the ease of programming (LabVIEW) to the reusing of libraries provided by the hardware vendors. New implementation has been done to include more features. To bring more people into the development of solutions for the beamlines and meeting the needs, training on EPICS, Python and CSS has been provided to the technical staff at the beamlines. Documentation on this open-source project is located at *http://lnls.cnpem.br/sol/hyppie*

REFERENCES

[1] J. R. Piton et al., "Hyppie: A hypervisored PXI for physics instrumentation under EPICS", BIW 2012, Newport News, MOPG031.

[2] M. Kraimer., E. W. Norum, M. Rivers, "asynDriver: Asynchronous Driver Support", April 1, 2010

[3] E. W. Norum., "How to create EPICS device support for a simple serial or GPIB device", April 12, 2010.

ISBN 978-3-95450-139-7