ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL002

PROJECT Nheengatu: EPICS SUPPORT FOR CompactRIO FPGA AND LabVIEW-RT

D. Alnajjar*, G. de S. Fedel, J. R. Piton,
Brazilian Synchrotron Light Laboratory, Campinas, Brazil

Abstract

A novel solution for integrating EPICS with Compact RIO (CRIO), the real-time embedded industrial controllers by National Instruments (NI), is proposed under the name Nheengatu (NHE). The CRIO controller, which is equipped with a processor running a real-time version of Linux (Linux-RT) and a Xilinx Kintex FPGA, is extremely powerful for control systems since it can be used to program real-time complex data processing and fine control tasks on both the Linux-RT and the FPGA. The proposed solution enables the control and monitoring of all tasks running on Linux-RT and the FPGA through EPICS. The devised solution is not limited to any type of CRIO module, setup or combination of modules. Its architecture can be abstracted into four groups: FPGA and LabVIEW-RT interface blocks, the Nheengatu library, Device Support and IOC. The Nheengatu library, device support and IOC are generic - they are compiled only once and can be deployed on all CRIOs available. Consequently, a setup-specific configuration file is provided to the IOC upon instantiation. The configuration file contains all data for the devised architecture to configure the FPGA and to enable communication between EPICS and the FPGA/LabVIEW-RT interface blocks.

INTRODUCTION

Experimental Physics and Industrial Control System (EPICS) [1] is an open-source set of libraries that helps building distributed soft real-time control systems for scientific instruments, like the ones that are used in the LNLS UVX beamlines [2], and those to be used in SIRIUS [3]. EPICS has been used to control LNLS experimental stations' equipment. Consequently, using EPICS to control CompactRIO (CRIO) [4] devices, the real-time embedded industrial controllers by National Instruments (NI), would be highly favourable.

The CRIO chassis contains a Xilinx FPGA and an Intel processor running an NI version of Linux. It also contains several slots where NI C-series modules containing peripherals can be plugged in. The peripherals can be accessed by the FPGA (hardware) and the LabVIEW-RT (Linux real-time operating system). Both the FPGA and LabVIEW-RT can be programmed using a visual programming language (LabVIEW) [5]. In that development environment, the user develops a software (VI) where building blocks can be connected to generate the desired circuitry to be implemented on LabVIEW-RT or the FPGA. Whether the LabVIEW-RT or the FPGA or both are chosen, it depends on the system's requirements. The FPGA can be used to perform high-speed processes and high-speed fine-control. LabVIEW-RT is slower than FPGA; however, it provides the ease and power of programming much more complex task limited by the processor speed and memory.

EPICS runs on Linux or Windows, so hypothetically it should run on the Linux-RT; however, extra libraries need to be developed to allow access of variables available on the FPGA or LabVIEW-RT. Several options developed by the EPICS community/industry were proposed and are being used. NI EPICS was developed by National Instruments [6], which is good for simple demands; however, it is limited in its functionalities. Consequently, IRIO [7] and CA LAB [8] were developed. IRIO developed an EPICS device driver to send and receive data to and from FPGA peripherals; however, all peripherals connected to the CRIO platform need to be supported by the library, and the library needs to be recompiled with every new setup. CA Lab uses libraries developed on top of LabVIEW-RT along with EPICS native libraries. All variables need to pass through LabVIEW-RT whether they belong to LabVIEW-RT or not, and it is difficult to integrate the rest of EPICS support software with this infrastructure (i.e. synApps [9]). For our use in SIRIUS, it would be highly advantageous to our development/debug cycle that the proposed solution can provide flexibility in several aspects, and they are:

- Allow EPICS to read variables generated on the FPGA VI, obtained from FPGA peripherals or generated on LabVIEW-RT VI running on CRIO.
- Allow EPICS to write to variables on the FPGA VI, that can be used internally or connected to a peripheral, or LabVIEW-RT VIs running on CRIO.
- Ease of integration with synApps or any other software intended to support the common requirements of particle accelerator beamlines.
- The deploy process should be as simple as possible. It is favourable that no compilation is necessary at all during the deploy process.

Given the previous requirements, there was a need to rethink a new solution that satisfies all these constraints and, consequently, Nheengatu was born.

* dawood.alnajjar@lnls.br

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL002

NHEENGATU

Nheengatu[1] (NHE) is a complete solution proposed to integrate EPICS into CRIO and to address all of the above stated requirements. A block diagram of NHE is shown in Fig. 1.

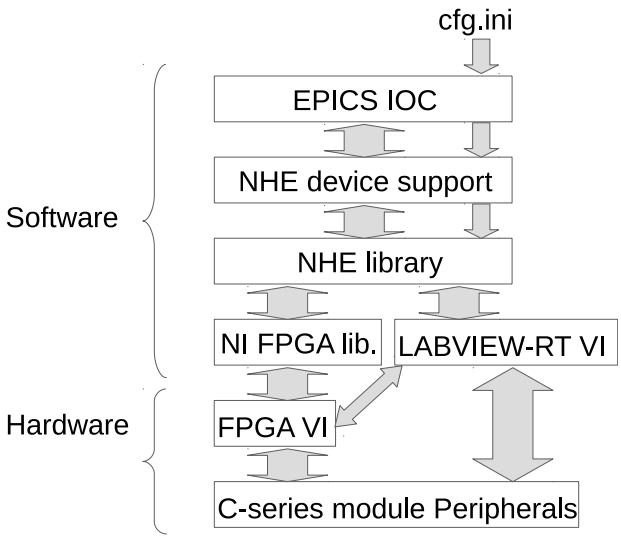

Figure 1: Nheengatu architecture

NHE can be divided into several layers as follows

- EPICS IOC: Software that will run in the CRIO host and that will export all FPGA and LabVIEW-RT available variables to the network though EPICS.
- NHE EPICS device support: Acts as an interface between the CRIO library and the IOC. Also, it provides some functions to configure the CRIO library through the IOC command terminal.
- NHE library: Library to handle all LabVIEW-RT and FPGA variable reading and writing.
- NI FPGA library: Library provided by National Instruments to access FPGA controls and indicators.
- LabVIEW-RT and FPGA VIs: LabVIEW application that contains all application-specific implementations whether they are fine-control state-machines, custom logic, or direct connections to peripherals.
- INI configuration file: Contains all information and addresses required for NHE to write/read to/from the FPGA and LabVIEW-RT.
- C-series modules: Modules with peripherals that will be plugged in the CRIO platform chassis.

Each component will be described more thoroughly in the following sections. In the meantime, a summary of the functionalities that are supported by NHE in the FPGA and LabVIEW-RT is illustrated in Table 1. It shows all the variable types that NHE can read and write to and from the FPGA VI, or from the LabVIEW-RT VI. A distinction is being made between fixed-point analog and floating point analog. The reason is that with fixed-point analog up to 52-bits of variable precision can be achieved. While in the case of floating point analog, a precision of 23 bits can be achieved; however, a wider range of numbers can also be reached (1.175494351 E -38 ~ 3.402823466 E +38)).

Nheengatu Architecture Layers

In this subsection, the NHE architecture layers will be described as follows:

C-series modules Any module that generates a datatype compatible with the datatypes mentioned in Table 1 can be plugged in the CRIO and used. The data obtained from the peripherals can also be preprocessed in the FPGA or LabVIEW-RT before being handed over to NHE or vice versa.

INI configuration file Configuration file that will be passed to the NHE library upon the execution of the EPICS IOC. This file contains all the necessary information for the NHE library to discover the source and the destination of the data to be read or written. Such information include: the IP of the CRIO where the FPGA bitfile will be written to, the Path of the bitfile, the shared memory path with the LabVIEW-RT, whether or not it will use the shared memory with the LabVIEW-RT, signature of the FPGA bitfile, name of the bitfile, shared memory size, binary input 64-bit variable address, binary input bit-name mapping, analog output addresses, analog input addresses, binary output addresses, list of scalers, addresses for scaler input and output ports, waveforms and their addresses, fixed-point analog inputs and outputs and their respective addresses.

LabVIEW-RT and FPGA VIs The FPGA VI has all FPGA-specific logic. For the upper layers to be able to read from and write to FPGA variables, the variables must be defined in the FPGA VI in the form of scalars or arrays of controls (write) and indicators (read). These indicators and controls can be read from or written to through the Linux-RT using the NI FPGA C API library.

For the Scaler EPICS record, RTL was developed, and imported into LabVIEW as an IP and used as the bases Scaler pulse counter (digital) and Scaler integrator (analog). As per the LabVIEW-RT VI, all the input and output variables are exchanged with the NHE library using interprocess communication (shared memory). A polymorphic VI library was developed to handle all interactions with the shared memory.

NI FPGA library The NI FPGA library is a wrapper to all NI functions that communicate with the FPGA. This set of functions was compiled into a dynamic library and used accordingly.

[1] Nheengatu, is the name of an indigenous language in South America, from the Tupi–Guarani language family. Nheengatu was the language that the different indigenous peoples, the portuguese explorers, black slaves and catholic missioners communicated with during the Brazilian colonial era. The language name is derived from the words *nheen* (meaning "tongue" or "to speak") and *katu* (became *gatu* and meaning "good"), resulting in "good language" or "easy language".

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL002

Table 1: Nheengatu Capabilities

Type	FPGA	LabVIEW-RT
Read	Single, Fixed-point, Boolean, Arrays	Double, Single, I8, I16, I32, I64, U8, U16, U32, U64, Boolean, Arrays
Write	Single, Fixed-point, Boolean	Double, Single, I8, I16, I32, I64, U8, U16, U32, U64, Boolean
Scaler	64 Counters - digital or analog	-

NHE library The NHE library is the core of the Nheengatu implementation. It abstracts all interactions with the FPGA and LabVIEW-RT VIs. Any data point is accessed using just a name (key), and the location of this data point is transparent to the upper layers. For the NHE library to know where to access, it needs to be provided with the configuration file at run-time, and that contains all the necessary information. The library provides checks on naming, and throws exceptions when needed.

NHE EPICS device support The NHE EPICS device support layer is responsible for converting the IOC commands to NHE functions. Six types of device supports were developed, and are believed to cover all the required need by SIRIUS, and are: Binary input, binary output, analog input, analog output, Scaler, and Waveform device support. These device supports are responsible for calling the respective input or output functions provided by the NHE library upon request by the IOC. The device support also provides functions to the IOC to configure NHE using the configuration file, and these functions can be executed from the IOC command terminal.

EPICS IOC The EPICS IOC is responsible for passing the configuration file to the NHE library. In the EPICS IOC database files, all the variables that the IOC wishes to exchange with the bottom layers should be defined using the EPICS IOC database standards. After the initialization of NHE, the IOC loads the database files and initializes the IOC. At this point, all the variables in the database files should be exported to the network through EPICS.

Minimum User Intervention

The architecture was developed so that there is little user intervention required. In most general cases, the user needs to generate the configuration file, the IOC template substitutions, along with the FPGA and LabVIEW-RT VIs. In some cases, if the IOC pre-defined templates are not sufficient and require modifications, the IOC may need to be compiled as well.

Nheengatu Integration with Support Software

Compatibility with support software such as synApps has not been broken since the standard EPICS IOC and EPICS device support design flow has not been violated. The procedure to add support software to the NHE IOC is the same as that of any IOC (add the path of the support software to the IOC makefiles and compile the IOC).

Further Automation

To further automate the process of deployment, python scripts to automatically generate the setup-specific configuration and the IOC template substitution files were developed. This does not only improve the deployment flow, but also drastically eliminates human errors and debug time since the configuration file contains many error-prone details of that specific setup.

Deploy Flow

It is important to remember that simplicity is one of our main objectives of developing NHE. For that purpose, the deploy flow should be as simple as possible. The approach adopted uses Network File System (NFS), where EPICS base, synApps modules, NHE library, NI FPGA Library and NHE device support are on the NFS. It is important to note that none of the NHE libraries need to be compiled and are generic. What changes between one CRIO setup and another is the configuration file, the IOC database substitutions and the FPGA bitstream. Given that the CRIO is configured to see the NFS and all the dynamic libraries are seen by the Linux-RT, the deployment flow is shown in Fig. 2.

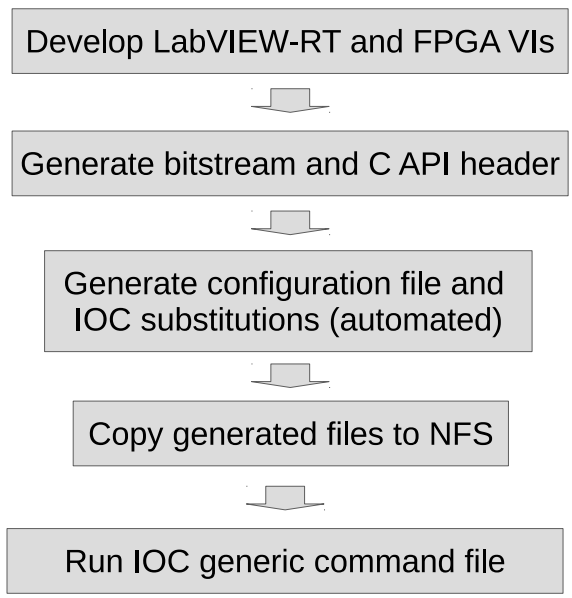

Figure 2: Nheengatu deploy flow

It can be seen that at no point there is a need to compile code. The configuration file is passed to the IOC during run-time and NHE becomes aware of available data points whether for reading or writing and whether from the FPGA VI or from the shared memory of the LabVIEW-RT VI.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL002

NHEENGATU IN ACTION

NHE was tested with software of the following versions: LabVIEW 2018 32-bit sp1, VI package manager 2018 f2, LabVIEW 2018 FPGA Module Xilinx Compilation Tool for Vivado 2017.2, FPGA Interface C API 18.0, LabVIEW 2018 Real-Time Module, EPICS 3.15.6, Synapps R6.0, NI Linux Inter-process communication v.1.5.1.19 (NI package manager), CRIO firmware version 6.5.0f0. Initial tests were done with both CompactRIO 9035 and CompactRIO 9045. NHE was permanently deployed in the XAFS1 (X-Ray Absorption and Fluorescence Spectroscopy) beamline of the UVX at LNLS to substitute the National Instruments PXI system previously used. So far the solution has shown an excellent performance.

CONCLUSION

The Nheengatu architecture for integrating EPICS with CRIO was proposed. Nheengatu enables the control and monitoring of all tasks running on Linux-RT and the FPGA through EPICS. The devised solution is extremely flexible and is not limited to any type of CRIO C-series module. The Nheengatu library, device support and IOC do not need to be compiled for every new CRIO setup. Setup details are all specified in a setup-specific configuration file and IOC database substitutions file that is only used at IOC runtime. The development and deploy procedure was simplified as much as possible. The devised architecture is believed to be the simplest-to-use and most flexible solution of integrating the CRIO controller with EPICS that exists up to date.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the Brazilian Ministry of Science, Technology, Innovations and Communications for financial support. The authors also would like to thank the Brazilian Synchrotron Light Laboratory (LNLS), specially the Beamline Software Group (SOL), namely: Luciano Guedes, Douglas Araújo, Marcelo Moraes, George Kontogiorgos, Lais Carmo, Allan Bugyi, Pedro Hirasawa, Gabriel Previato and Gustavo Aranha for their technical support and feedback.

REFERENCES

[1] EPICS:Experimental Physics and Industrial Control System, https://epics.anl.gov/

[2] UVX - LNLS, https://www.lnls.cnpem.br/uvx-en/

[3] A. R. D. Rodrigues *et al.*, "Sirius Light Source Status Report", in *Proc. IPAC'18*, Vancouver, Canada, Apr.-May 2018, pp. 2886–2889. doi:10.18429/JACoW-IPAC2018-THXGBD4

[4] CRIO: National instruments Compact RIO, http://www.ni.com/pt-br/shop/compactrio.html

[5] LabVIEW, https://www.ni.com/pt-br/shop/labview.html

[6] NI EPICS: National instruments EPICS solution, https://www.ni.com/pt-br/innovations/white-papers/12/introduction-to-epics.html#UsingEPICSinLabVIEW

[7] IRIO: IRIO Technology for advanced FPGA data acquisition systems, http://www.i2a2.upm.es/idi/instrumentacion-aplicada/sistemas-de-adquisicion-de-datos-avanzados-2/irio-technology/?lang=en

[8] CA Lab: LabVIEW + EPICS solution by HZB, https://www.helmholtz-berlin.de/zentrum/locations/it/software/exsteuer/calab/index_en.html

[9] EPICS synApps support software, https://www.aps.anl.gov/BCDA/synApps