EPICS IN THE PXI SYSTEM

Kukhee Kim*, Charles J. Ju, M. K. Kim, M. C. Kyum, and M. Kwon,
KBSI, Yusung-gu, Daejeon 305-806, Korea

Abstract

The KSTAR (Korea Superconducting Tokamak Advanced Research) project [1, 2], which aims to construct a superconducting tokamak, was launched in 1996. Much progress in instrumentation and control has been made since then. Commercial products for system controllers and physics diagnostics such as the PCI eXtensions for Instrumentation (PXI) based on the LabView systems have been considered. The commercial solution has some limitations in performance and offers poor support for a large number of I/O implementations. To overcome these limitations, Experimental Physics and Industrial Control System (EPICS) [3] software will be used for KSTAR control. Now, we must consider the preservation of the pre-investment PXI hardware and how to integrate it into some of the local control systems which are still being developed by based on the LabView, hence the integration of National Instrument's PXI and EPICS is necessary. We successfully constructed a prototype vacuum controller by integrating EPICS in National Instruments products.

INTRODUCTION

We are considering two types of integration; close integration and loose integration. For close integration, we can use EPICS software instead of LabView for the PXI system. We can use DAQmxBase[4]/DAQmx, which are libraries provided by National Instruments (NI), to build EPICS device/driver supports. EPICS works in its own device layer which was built with National Instrument's Application Program Interface (API). For loose integration, we can use shared memory to access LabView data from EPICS Input Output Controller (IOC), which is located on the same CPU. It is also possible to implement user defined protocol on the TCP/IP application layer to communicate with the RTLabView System. In particular, we developed a simple protocol for the NI compact field point which operates on the RTLabView and successfully controlled this device under EPICS. We describe the details of our implementation here.

CLOSE INTEGRATION

We are considering the using of the PXI based Data Acquisition (DAQ) System for the superconducting magnets and structure monitoring systems. The parameters to monitor are temperatures, voltages in the magnet system, and stress and strain in the structures. Some signals need special signal conditioning and also need to be compensated for the nonlinear response of sensors. Commercial adaptive signal conditioning modules are available for the PXI system. Such modules have fast bus speed, enough computational power to handle heavy loads from large number of I/O points, are able to compute compensation for nonlinearity, and have thus reduced development efforts. However, until recently, the PXI system has been supported only by LabView software provided by NI. In order to use EPICS instead of LabView, we have to develop a software layer to support communication between EPICS and PXI hardware. It is also necessary to build device support which is an intermediate layer between the driver support [5] and record support [6]. The device support [7] contains details of each individual record type in EPICS. Hence, we started to develop those driver support and device support named *genericPXI*. The drivers will be used not only for monitoring system, but also for plasma diagnostics and control systems. To satisfy this requirement, the genericPXI has the following structures and features.

*Corresponding author. Tel: +82-42-870-1616; fax: +82-42-870-1609.
E-mail address: kimkh@kbsi.re.kr (K. H. Kim)

Driver/Device support: genericPXI

The *drvgenericPXI* (driver support) is a software layer located in the bottom of EPICS and has access to the PXI hardware through DAQmxBase/DAQmx libraries. These libraries consist of a set of API's for direct access to the PXI hardware.

This driver can correspond with many kinds of device support, such as, analog input (ai), analog output (ao), binary input (bi), binary output (bo), multibits input (mbbi), multibits output (mbbo), waveforms, etc – which are conventional EPICS record types [8]. It has command line configuration and status monitoring features. Users can configure PXI hardware by issuing commands on the IOC shell and also monitor status of driver by command line functions [9]. It is a useful configuration method in EPICS, because, in most of the cases, these commands are called by start-up script when the IOC is booted automatically. The driver has two kinds of operation modes. One is the External driven mode and the other is the Continuous acquisition one.

External driven mode:

The External driven mode corresponds to ao, bo and mbbo types of device support. The sequence of operations is as follows: Events are generated by user threads (other record processes, Channel Access (CA) events, periodic scanner and dbAccess) and this activates record support, see Fig. 1. Record support forwards the request to device support, which in turn puts the request into the queue. This processing chain (previous paragraph) runs on the user thread. After putting the request in the queue, the

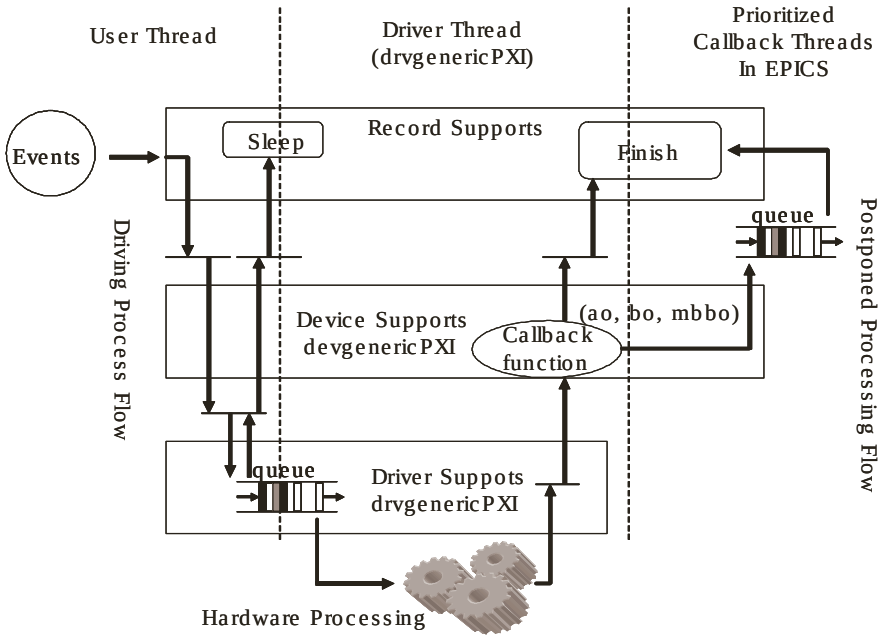

Figure 1: External driven mode.

record processing is temporarily on hold, to be finished later. The user thread is now free to run other things.

When driver thread detects a non-empty queue, it will start hardware processing and will be blocked until the processing is finished. When hardware finishes processing the request, driver supports will invoke the callback function in the device support. The callback function finishes the record processing. If the driver thread has a heavy load, the callback function in the device support could request one of three callback threads (prioritized callback/general purpose callback tasks) [10] in EPICS to complete the record processing. This decision should be made by the programmer for the device layer when the callback function is built. If the record processing is completed by the driver thread, the structure of the program becomes simpler. However, this sacrifices the response of driver thread because the driver thread can not respond immediately to the next external events. We can expect faster response if we choose prioritized callback to complete record processing; however, there is a small overhead associated with changing running threads to complete record processing, callback queuing, monitoring queue, context switching, etc. Overall faster response still does compensate for small overheads.

As the previous discussion, the record processing is not completed by just one thread at a given time, there are more than two threads involved, and also more than two execution time slots are involved thus we can call asynchronous processing for the record processing [11].

Continuous acquisition mode:

The Continuous acquisition mode corresponds to ai, bi, mbbi, and waveform types of device support. There are two types for the continuous acquisition mechanism: driver thread running on infinite loop (acquisition loop) at prescribed rate, and swing buffer mechanism on the Direct Memory Access (DMA) buffer, see Fig. 2. For both mechanisms, the execution rate depends on sampling rate, triggering, and number of data samples.

In the first type, whole processing chains (from driver layer processing to record layer one) are driven by the acquisition loop in the driver layer. This loop has two linked lists (pre-callback and post-callback), each linked list executes callback in the device layer. The pre-callback linked list stores a list of callback functions to execute before starting the hardware acquisition. The post-callback linked stores a list of callback functions to execute it after the acquisition ends. If there is nothing to do before acquisition, the pre-callback linked list is empty. However, if there is some pre-processing to be done, the device programmer can use the pre-callback feature to implement any preprocessing function needed in the device support. The post-callback executes post-processing actions in the device support, and the device support invokes record processing. If the post-processing makes record processing directly, the entire record processing is running on the driver thread. It is not advisable to have a fast execution rate for the acquisition loop. While the driver thread executes record processing, the DMA buffer can overflow if the hardware has a fast acquisition rate. To prevent overflow, prioritized callback for post processing can be used. The post-processing inserts a function pointer to invoke record processing into a prioritized callback queue, so that the driver thread can run again on the acquisition loop. Using prioritized callback results in better performance in the acquisition loops. However, there can be an unpredictable short delay in record processing because the prioritized callback may be processing something else.

In the second type, the record processing could be triggered by events originating from the user thread. In this case, the record processing is synchronous. As shown

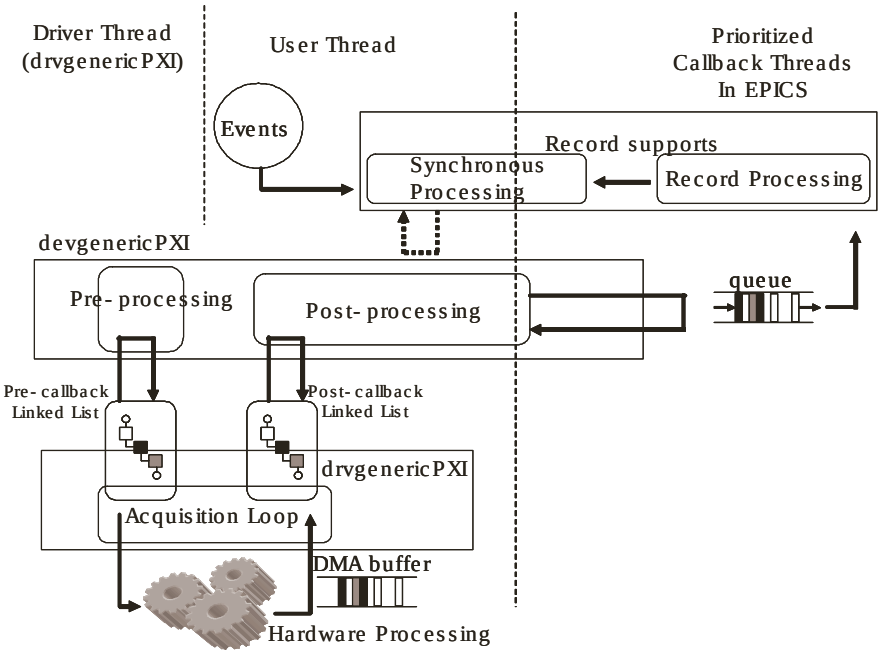

Figure 2: Continuous acquisition mode.

by the dashed arrow in Fig. 2, record support immediately reads out the acquired data from a buffer in device layer and completes the record processing. The acquired data could come from the previous step in the acquisition loop.

Multiple device layer considerations:

The *devgenericPXI* is designed for conventional EPICS record supports which have been developed to correspond to generic machine control and monitoring. However, the genericPXI needs additional capabilities (e.g., streaming and archiving features) because it will be used for continuous data acquisition systems for many of the KSTAR diagnostics systems. It has to handle heavy data transfer to the archiving system without data loss during the acquisition. To achieve this requirement, the drvgenericPXI can work with a special device support (i.e., streaming device support) or can also work with both of them (a special device supports and devgenericPXI). We have included some features in drvgenericPXI to handle multiple device supports. One of the features is pre- and post-callback linked lists. There are no limitations to register callback functions which are located in individual device supports. The only limitation is the execution time of the linked lists. We have to reduce the callback execution time as in the previous discussion. The other feature is re-entrancy and thread safety for APIs in drvgenericPXI. Individual device supports which are related with drvgenericPXI can make connection with both of the features.

Portability of the genericPXI

The devgenericPXI doesn't have any knowledge about the PXI hardware and its library, but drvgenericPXI does. However, the driver does not have direct control of PXI hardware. Direct access to PXI hardware is done through the DAQmxBase library, provided by National Instruments. Actually this will be replaced by DAQmx. NI started support for the DAQmxBase in the Linux environment, but this seems to be a small pilot project, supporting only two kinds of NI hardware, the E series and M series. Therefore, NI wants to move to DAQmx because this software is their base software chassis, supports almost all NI hardware and is platform independent [12]. If they move to DAQmx, little or no change needs to be made to drvgenericPXI. Since the DAQmxBase library has almost the same structure as DAQmx, genericPXI can support most NI hardware.

LOOSE INTEGRATION

In the previous section we have discussed EPICS implementation to access PXI hardware instead of using LabView software. In addition, we need to implement run-time data exchange between EPICS and LabView. Some of the KSTAR local control system will be built with LabView because it will be developed by other institutes with their own hardware and devices. Other institutes prefer to implement systems using LabView because of lack of experience with EPICS and moreover there is no software engineer available. In this case, we can use two kinds of integration methods; one is a simple channel access (SCA) library [13] which is a Dynamic Link Library (DLL) type shared library in the MS Windows system. LabView can access EPICS process variables (PV's) with this library. The other one is using shared memory in EPICS soft IOC which is running on same CPU with LabView. Thus, the soft IOC can access control variables in LabView through shared memory [14]. We already have used SCA to make some of our operator interface (OPI). However, we will minimize the use of those two methods because we aim to make the KSTAR control system as a unified as possible under the EPICS framework.

We now discuss about other aspects of KSTAR control. Many of KSTAR local control systems, such as the vacuum and cryostat controllers, need robust and cheap I/O modules that are not necessarily fast. There is no need for either fast response or heavy computational capability similar to the VME system. The requirements specify the use of remote I/O modules. The EPICS softIOC and NI's CompactFieldPoint (cFP) solution satisfied the design requirements. The cFP has a small CPU module for the RT-LabView program on PharLab RTOS [15] and also has a variety of I/O modules – ai, ao, di, and do. We have developed a small protocol based on TCP/IP to enable communication between EPICS and cFP. One half of the protocol was developed on RT-LabView to execute on cFP CPU module. The other half of the protocol was developed in EPICS, namely, *drvcFP20x0* and *devcFP20x0*. This protocol has a self configurable feature. When the communication is established, the cFP notifies EPICS about their hardware configuration (module types, order of modules, number of channels for each module, etc), then EPICS decides how to parse the message from cFP and how to address individual channels in the cFP I/O modules. We have implemented cFP and the above protocol with linux based softIOC for our vacuum system prototype. We can control and monitor all of our vacuum valves and pumps with this protocol.

We hope that the cFP remote I/O solution with the communication protocol can replace some of planned Programmable Logic Controller (PLC) hardware. At first, we have considered PLCs and their remote I/O modules for slow control systems. We now can use soft IOC with State Notation Language (SNL) programs instead of PLC CPU modules and ladder programs. In addition, we plan to use the cFP remote I/O solutions instead of PLC I/O modules. The SNL program is a native feature of EPICS, it has a powerful state programming language, deep coupling with EPICS, simplified development, easier maintenance, and relieves the need for a PLC engineer. Moreover, the cFP remote I/O is cheaper than PLC modules.

CONCLUSIONS

We have discussed EPICS PXI support which is device/driver software running on NI's DAQmxBase abstraction layer. It will support almost all NI hardware on PXI/PCI bus without any modification after NI releases their DAQmx library instead of DAQmxBase. The device/driver now has a boot configuration feature, configured by IOC shell command and start-up script. However, we plan to upgrade it for run-time configuration in the near future. Then, we can configure the hardware configuration on the fly through the EPICS records which are related to the run-time configuration feature.

We also discussed NI cFP remote I/O solution with our communication protocol. This was successfully implemented in the vacuum prototype controller. We hope it can replace PLC hardware in the KSTAR control system. We can use EPICS soft IOC with the SNL program instead of the PLC CPU module and ladder program. Thus, the development is simplified and maintenance becomes easier.

ACKNOWLEDGEMENTS

This work is supported by the Korean Ministry of Science and Technology (MOST). Thanks to Dr. Ajit Gokhale at NI for his explanation about DAQmx, DAQmxBase and the roadmap of the products.

REFERENCES

[1] G. S. Lee, J. Kim, S. M. Hwang, et al., "The KSTAR Project: Advanced Steady-state Super-conducting Tokamak Experiments" 17th IAEA Fusion Energy Conference, Yokohama, Japan, Oct. 19-24, 1998, and "The design of the KSTAR Tokamak Engineering" Fusion Engineering and Design, Vol. 46 Issues 2-4 405 (1999).

[2] M. Kwon, J. S. Bak, G. S. Lee, "Progress of the KSTAR Tokamak Engineering," Fusion Science and Technology 42, 167 (2002).

[3] "Experimental Physics and Industrial Control System", http://www.aps.anl.gov/epics

[4] "Advanced Data Acquisition Series: Programming Data Acquisition for Linux with NI-DAQmxBase", http://zone.ni.com/devzone/conceptd.nsf/webmain/42B73A7B82F0FEC786256FB1007227EC

[5] Marty Kraimer, Janet Anderson, Andrew Johnson, Eric Norum, Jeff Hill, Ralph Lange, "EPICS: IOC Application Developer's Guide R3.14.4", pp 175

[6] Marty Kraimer, et al., *ibid*. pp 153

[7] Marty Kraimer, *ibid*. pp 169

[8] Philip Stanley, Janet Anderson, Marty Kraimer, "Record Reference Manual"

[9] Marty Kraimer, et al., *ibid*. pp 231

[10] Marty Kraimer, et al., *ibid*. pp 215

[11] Marty Kraimer, et al., *ibid*. pp 157

[12] Ajit Gokhale, Private Communication about the DAQmxBase and DAQmx, ajit.gokhale@ni.com

[13] "Simple Channel Access Library", http://www-controls.als.lbl.gov/epics_collaboration/sca/

[14] Dave Thomson and Willem Blokland, "LabView Shared Memory Interface to EPICS IOC", http://www.sns.gov/diagnostics/documents/epics/LabVIEW/SNS_LabVIEWEPICS.html

[15] Ajit Gokhale, Private Communication about RTLabView, cFP and PharLab RTOS