DEVELOPMENT OF EPICS BASED BEAM-LINE EXPERIMENTAL CONTROL EMPLOYING MOTOR CONTROLLER FOR PRECISION POSITIONING

Anupriya Tuli[1], Rajiv Jain[2], H S Vora[2]
1. Amity School of Engineering and Technology, Amity University Noida UP.
2. Raja Ramanna Centre for Advanced Technology, Indore, MP

ABSTRACT

In a Synchrotron Radiation Source the beamline experiments are carried out in radiation prone environment, inside the hutch, which demands to conduct experiments remotely. These experiments involve instrument control and data acquisition from various devices. Another factor which attributes to system complexity is precise positioning of sample and placement of detectors due to inherent small beam size. A large number of stepper motors are engaged for achieving the required precision positioning.

This work is a result of development of Experimental Physics and Industrial Control System (EPICS) based control system to interface a stepper motor controller developed indigenously by Laser Electronics Support Division of RRCAT. EPICS is an internationally accepted open source software environment which follows toolkit approach and standard model paradigm. The operator interface for the control system software was implemented using CSS BOY. The system was successfully tested for Ethernet based remote access. The developed control software comprises of an OPI and alarm handler (EPICS ALH). Both OPI and ALH are linked with PV's defined in database files. The development process resulted into a set of EPICS based commands for controlling stepper motor. These commands are independent of operator interface, i.e. stepper motor can be controlled by using these set of commands directly on EPICS prompt. This command set is illustrated in the above table. EPICS Alarm Handler was also tested independently by running these commands on EPIC prompt. If not using ALH, operator can read the alarm status of a PV using 'SEVR' and 'STAT' attributes.

INTRODUCTION

In a particle accelerators stepper motors are used in beamline experiments to achieve high precision positioning of sample or detector mounted on it according to the need of the experiment.

A multi-axis stepper motor controller was developed by LESD, RRCAT for such applications. Owing indigenously development, its driver for EPICS were not available. It was necessary to absorb this motor controller in beamline experiments in a standard interface. The EPICS software environment used to develop and implement distributed control systems for experimental control. It provides Supervisory Control and Data Acquisition (SCADA) capabilities. It is a distributed process control system build on software communication bus. EPICS includes a set of functional sub systems (Display manager, Alarm manager, Achiever, Sequencer, Application program in C and Channel Access Protocol) which work in collaboration to provide total functionality of a control system. The EPICS is not only a software toolkit, it is also collaboration and control system architecture. It supports client/Server architecture, enabling the design and development of the systems which include large number of networked nodes providing control and feedback [10]. This paper presents the development of control system software using EPICS to interface indigenously developed stepper motor controller.

Control System Architecture

The beamline experiments involve a large number of detectors, controllers, and various test & measurement equipment from multiple vendors applying heterogeneous technology. All these devices and equipment are controlled using control system software and hardware. Clout [1] attributes time scale associated with accelerators and varied complexity of diagnostic equipment as the reason for difficulty in controlling the accelerators.

Singh [2] describe that software architecture plays a defining role in development of control system software. His work highlights the flexibility of multilayer software architecture in accommodating integration of new equipment/machines in exiting design state of accelerator control system.

Operating System for Control System

Development and deployment of any application system highly depends on the operating system. Chepurnov [3] developed an industrial style control system, which uses Linux as operating system for developing and running application. On the basis of their experience they propose Linux as operating system for both middle layer and presentation layer of control system architecture. In an article, Nichols [4] describes the role of Scientific Linux at CERN. The vision behind Scientific Linux was to provide a customizable operating system platform which meets the requirements of high energy physics experiments.

Software Tools for Developing Control System

While developing control system, a plethora of SCADA packages are available in market to choose from. These packages are either commercially available or are open source. In his work, Singh [2] states that though industrial SCADA is an available choice for developing control systems, it is not suitable for developing accelerator control system software. The author mentions EPICS SCADA being a popular choice among accelerator community for developing control systems.

Barana [5], presented a comparison between four control system development software packages, two being commercial (FTV-SE and PVSS II) and remaining two were open- source (EPICS R3.14.10 and TANGO). These packages were tested against SCADA capabilities and ease

of system development. Observations revealed that though FTV-SE provides an ease in system development, windows is the only choice available in terms of compatible operating systems. Also, it requires an intermediate layer when communicating with a third party device. On other hand TANGO do not provides ease in system development, but support rich tools and multiple object oriented languages (Python, Java & C++). On comparing PVSS with EPICS, prior will be given preference when it comes to communication, because EPICS require a number of scripts to be configured for establishing communication. On other hand EPICS is open source and showed better network performance.

White [6] shows the transition in accelerator control system development. In last fifteen years more than 100 projects throughout the world were successful in building control systems using EPICS due to fact that high degree of software reusability supported by toolkit approaches and SCADA approaches results in decrement of development time and cost.

Experimental Physics and Industrial Control System (EPICS)

Dalesio [7, 8] defines EPICS as a software environment which follows the toolkit approach of developing control systems. It is also an architecture and collaboration between accelerator labs and industry. EPICS follows 'standard model' paradigm i.e. EPICS applies standards at every layer to achieve increased performance and balance between cost/performance tradeoff. He further defines the architecture of EPICS which includes a set of sub systems (application programs in C, alarm handler, display manager, achiever and sequencer). EPICS based control system is based on communication between these subsystems (software).

There are various Operator Interface (OPI) tools available in market for the development of each EPICS subsystem. Farnsworth [9] compared different OPI development tools (MEDM, EDM, CSS BOY, EPICSQt, CAQtDM and AS-Delphi) for acceleration control. For performing analysis, a database of 500 records was used, which updated from 0 to 99 with a speed of 10 times per second. These tools were tested against two conditions; one when OPI would skip the update and other is the situation of complete failure of OPI (halt state).The analysis was carried for windows & Linux and for both text & graphical widgets. The result showed that CSS BOY (Control System Studio, Best OPI Yet) which is a Java based tool performed better than EPICSQt which is a C/C++ based tool.

CONFIGURING HARDWARE DEVICES

The stepper motor controller used for this project is developed in-house by LESD group, RRCAT. It is a multi axis stepper motor controller without encoder. The controller can either be connected on RS-232 or USB. This is made possible as the controller has basic RS-232 interface, enhanced to work on USB using FTDI controller. Thus it can be is configured and mounted as a serial device by Linux machine even while using USB. For the basic Motorized Translation Stage, MTS 6565 from M/s Holmarc Opto-Mechatronics, Kochi has been used as positioning device with stepper motor controller. The specifications of both the devices are summarized in table 1.

Stepper Motor Controller	
Baud Rate	9600
Data Bits	8
Parity	None
Stop Bit	1
Flow Control	None
Translational Stage	
Travel	10 mm
Motor Basic Step Angle	1.8 Deg/Step
Lead Screw Pitch	0.5 mm
Resolution (half step)	0.00125 mm
Load Capacity	1 Kg
Holes for mounting	M4
Connector	9 pin connector
Switches	Two inbuilt switches (Home/End)

Table 1 : Hardware Device Specifications

SOFTWARE DEVELOPMENT

The approach followed to build the control system includes the major steps involved in a software development cycle. These steps are illustrated using flowchart in figure 1.

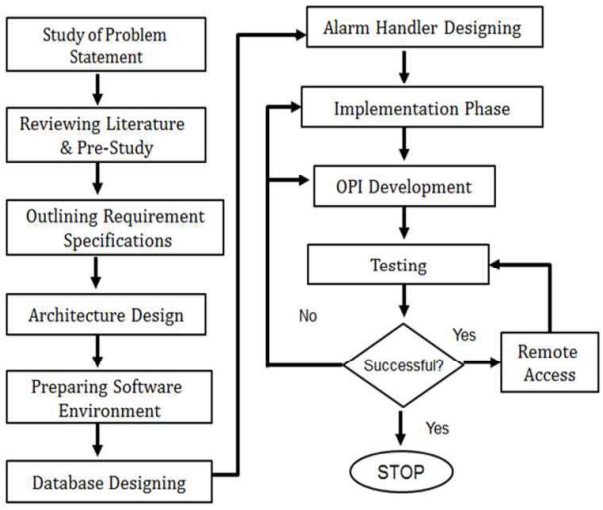

Figure 1: Steps involved in control development

Architecture Design

EPICS is a toolkit of subsystems, which work collaboratively to provide functionality of a control system.

As per requirement an operator is provided with an integrated access to the set of selected sub systems, where each subsystem performs different task for controlling the remote device.

The distributed multilayer software architecture is used for building this control system. In accordance with the requirement specification, a basic alarm handling system has also been incorporated. The figure 2 illustrates the architecture design followed in this project.

Database Designing and Record Support

Designing of database is most crucial phase of the approach as backend of control system software is made up of collection of distributed databases. These databases define Process Variables (PV's) associated with interfacing device.

Successful communication with stepper motor controller was established by attaching asyn Record to it. This record type facilitates the generic communication to a device having GPIB, serial or Ethernet port. The database was designed around asyn Record and its communication with group of other record types like analog input, string input and string calculation output.

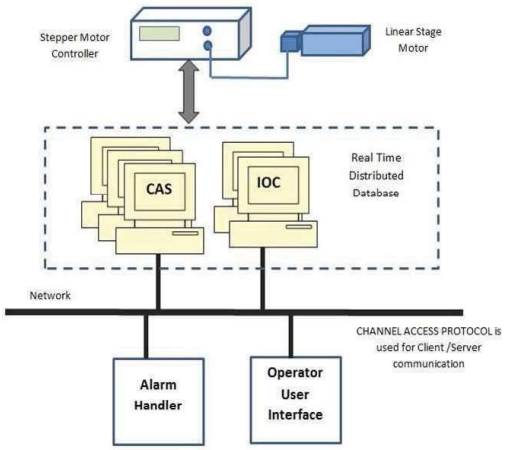

Figure 2: Architecture for Control

OPI and Alarm Handler

There are various OPI tools available in market. Studies reveal that CSS BOY, a Java based OPI development tool has better performance over C/C++ based OPI development tools. Thus the frontend for the system was designed and developed using CSS BOY. Though CSS provides its own alarm handling system, we chose EPICS ALH for the project. Though ALH is MEDM based, it serves as the best fit for the projects with very basic alarm handling requirements.

RESULT

The developed control software comprises of an OPI and alarm handler (EPICS ALH). Both OPI and ALH are linked with PV's defined in database files. Figure 3 shows the captured screenshot of the developed control system software.

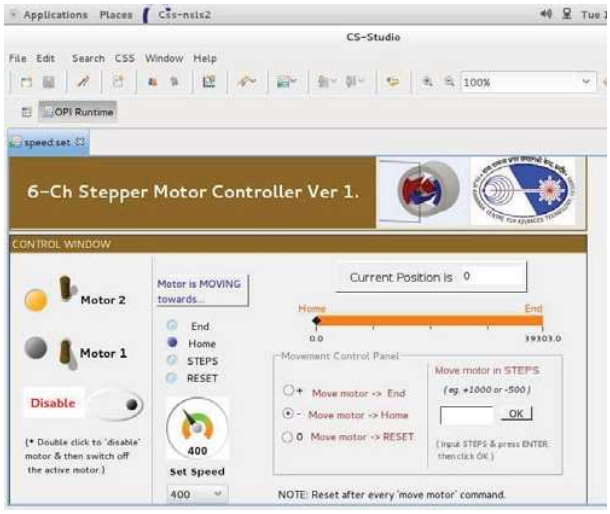

Figure 3: Stepper motor control system software

EPICS prompt command set

The development process resulted into a set of EPICS based commands for controlling stepper motor. These commands are independent of operator interface, i.e. stepper motor can be controlled by using these set of commands directly on EPICS prompt. This command set is illustrated in table 2.

EPICS Alarm Handler was also tested independently by running these commands on EPICS prompt. If not using ALH, operator can read the alarm status of a PV using 'SEVR' and 'STAT' attributes.

Software Test Results

The software was tested to control movement of single stage motor attached to stepper motor controller. The following results were observed while testing the software,

- The software requires minimum CPU utilization.
- The software was tested for baud rate 9600 and 19200.
- Software also supports EPICS Alarm Handling system.
- EPICS Alarm Handling system successfully generated alarms even when tested directly on EPICS prompt using EPICS prompt command set.
- Ethernet based remote access was successfully.

SUMMARY

This work is a result of development of control system software for a stepper motor controller. This software will be used to assist beamline experiments at Indus-2, RRCAT. Indigenously developed motor controller was used for interfacing successfully with EPICS under Linux environment.

The asyn Record was used to establish communication with the controller. The system's operator interface was developed using CSS BOY, whereas EPICS ALH performs the alarm handling. This development process also resulted into an OPI independent EPICS prompt command set. This command set independently serves the purpose of stepper motor controller including alarm handling.

FUNCTION	MOTOR CONTROLLER COMMAND SET	PROCESS VARIABLE	USER INPUT VALUE	EPICS PROMPT COMMAND SET
Controller Identification	V4<CR>	ID:is	Nil	caget ID:is.SVAL
Current Step Position	V1<CR>	curr_position:is	Nil	caget curr_position:is
Move Motor to End	H+<CR>	mv_motr:to	+	caput mv_motr:to +
Move Motor to Home	H-<CR>	mv_motr:to	-	caput mv_motr:to -
Set Motor Position	-(no of steps)<CR> +(no of steps)<CR>	set_pos:mv	Direction & No. of steps (+ or -)(X)	caput set_pos:mv (+ or -)(X)
Set Motor Speed	T(100 - 1000)<CR>	speed:set	(100 -1000)	caput speed:set (100 -1000)
Motor Selection	B (motor no 1-2)<CR>	set_motr:numbr	(1 or 2)	caput set_motr:numbr (1 or 2)
Disable Motor	Z<CR>	disable:motr	(1 or 2)	caput disable:motr (1 or 2)

Table 2: EPICS prompt command set for the given stepper motor controller

ACKNOWLEDGEMENT

The development work for was carried out RRCAT, Indore, India. The first author wishes to express her gratitude towards Mr. C.P. Navathe, Head, LESD and authorities of RRCAT for giving opportunity to work at RRCAT and co-operation received from all quarters.

REFERENCES

[1] P. Clout et al., "Distributed Computers in Accelerator Control Systems," *Distributed Computer Control Systems,* pp. 135-141, 1986.
[2] S. Singh et al., "Particle accelerator Control System," Proceedings of the DAE Symposium on Nuclear Physics 55, p. 118, 2010.
[3] A.S. Chepurnov, et. al., "Operating System Linux as developing and runtime platform for control system of particle accelerator," in Proceeding of EPAC 2000, 2000.
[4] S. J. Vaughan-Nichols, "High-Energy Linux: Linux & the Large Hadron Collider," December 2009.
[5] Barana O., et. al., “Comparison between commercial and open-source SCADA packages -A case study," vol. 85, pp. 491-495.
[6] K. S. White, "Status and future developments in large accelerator control systems," in *Proceedings of ICAP 2006*, France, 2006.
[7] L.R. Dalesio, et. al., "The experimental physics and industrial control system architecture: Past, present, and future," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 352, no. 1-2, p. 179, 1994.
[8] L. R. Dalesio, "EPICS Architecture," ICALEPS, pp. 278-281, 1991.
[9] Farnsworth, et. al., "Experimental Physics and Industrial Control System," Aragonne National Laboratory, Feb'2015. http://www.aps.anl.gov/epics/
[10] "Experimental Physics and Industrial Control System," Aragonne National Laboratory, Feb'2015. http://www.aps.anl.gov/epics/