SOFTWARE FOR POWER SUPPLIES CONTROL OF THE NSLS-II BOOSTER SYNCHROTRON

P. Cheblakov, A. Derbenev, S Karnaev, S. Serednyakov, BINP SB RAS, Novosibirsk, Russia
M. Davidsaver, Y. Tian, BNL, Upton, NY 11973, USA

Abstract

The booster synchrotron of the NSLS-II light source at Brookhaven National Laboratory (BNL) provides electron beam acceleration from 200 MeV up to 3 GeV.

This paper describes software for the booster synchrotron PSs control which is based on EPICS and includes a specially designed data base for electronics configuration, a set of programs to manage ramp functions and to control both ramping PSs for the beam acceleration and pulsed PSs for the beam injection and extraction.

INTRODUCTION

The beam acceleration in the booster synchrotron is performed in 250 - 300 ms with repetition rate of 1 or 2 Hz. This imposes strict requirements on both the accuracy of Power Supplies (PSs) control and synchronization of the ramping process. The use of the BNL developed Power Supply Controller (PSC) [1] allows us to precisely control ramping PSs with a time step of 0.1 ms and a relative time accuracy of about several nanoseconds with 20-bit DAC. At the same time PSC provides nine 16-bit ADCs for each DAC channel. Injection/Extraction pulsed PSs are also controlled with PSCs that allows a flexible driving of the energy storage charging process for a high stability of the pulsed output current.

The Input/Output Controller (IOC) for the PSC is described in this paper. The IOC running in the IBM server prepares 10k setpoint waveforms for uploading them to PSC, processes measured 10k waveforms from each ADC channel and provides alarm signals in the case of unacceptable deviation of control and measured parameters.

High Level Applications are developed in Control System Studio and Python that is suitable for tasks of almost any complexity.

SOFTWARE STRUCTURE

The booster control system [2, 3] is based on EPICS and provides continuous control of the all booster devices and diagnostics equipment during 1 or 2 Hz operation cycle of the booster. In the main operation mode users creates and upload ramp functions into PSC via IOC. A ramp function determines behaviour of the reference voltage at the DAC output during the ramp. The ramp function is been written in the ramp waveform of 10k points that corresponds 1-second time interval and is been uploaded to PCS. After that PSC carries out the uploaded waveform (which contains ramp function values) without interruption and sends out setpoints to DAC point by point. Switching to the new waveform is performed automatically when the new waveform is uploaded and is triggered with booster cycle trigger signal.

Software is built using the classic three-level EPICS scheme (Fig. 1). There are control electronics on the lower level, EPICS IOCs on the middle level and user applications and operator screens on the upper level.

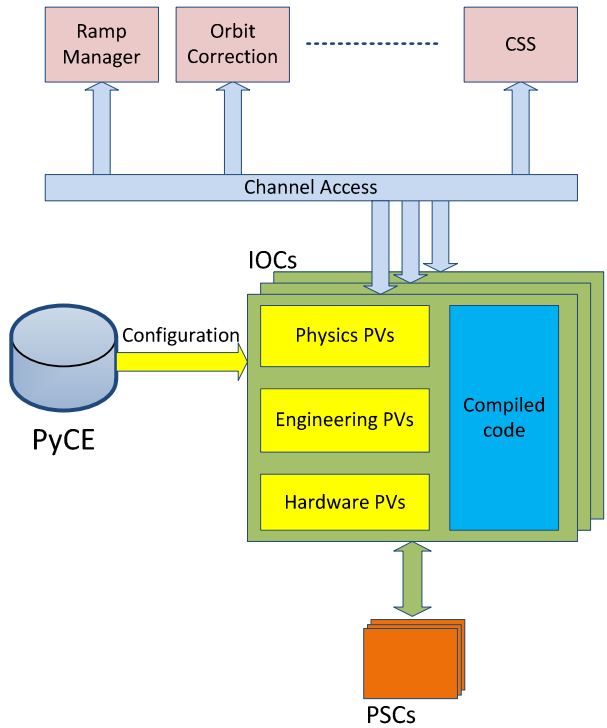

Figure 1: PS control system block diagram.

Low Level

Low level software is executed on PSC. PSC is based on FPGA Spartan®-3A by Xilinx© and a part of FPGA is used for MicroBlaze™ Soft Processor Core. In the FPGA-part the following functions are implemented:

- sending waveform setpoints to DAC,
- ADC measurements buffering,
- smooth waveform switching via transition curve,
- synchronization with the timing system 10 kHz clocks,
- synchronization with the booster cycle.

Software part of PSC is written on C language and consists of:

- support of communication with IOC via TCP/IP,
- ramp tables uploading,
- ADC measurement downloading just after the next cycle trigger.

ISBN 978-3-95450-122-9

Middle Level

Middle level of the control system is represented by the set of EPICS IOCs. Each IOC is responsible for PSs of a certain type. Main tasks of IOC are:

- communication with PSC via TCP/IP,
- three-level PV hierarchy,
- processing of all DAC/ADC data as arrays,
- math processing of signals,
- transition curve generation and preparation of a transition ramp,
- PSC/PSI status monitoring (state machine),
- PSs operation monitoring,
- preparation of the data for archiving.

High Level

It was appropriate to adapt existing approaches to building high-level applications in the EPICS environment. Most suitable tools to achieve this goal being Control System Studio (CSS), an Eclipse-based user interface framework for control systems, and Python, a high-level programming language with EPICS support available. Jython, an implementation of the Python programming language written in Java, is used to provide complex behaviour in CSS-based applications.

A simple "What You See Is What You Get" editor and native Jython script support in CSS in conjunction with object-oriented and highly extensible Python provide successful development of many high-level applications of various kinds as a part of the NSLS-II control system.

IOC SUPPORT

Communication protocol with PSC is implemented in the IOC. Initially asynDriver was used for this communication. But it was too complex and had a lack of possibilities. Therefore in the recent version we refused asynDriver and used direct operation via a socket. The protocol is asynchronous and allows transmitting of ramp waveforms to PSC, receiving waveforms with measurements from all ADCs in a bulk, receiving digital inputs states which indicate a PS status and interlocks and, finally, transmitting commands to switch a PSC operation mode and digital outputs of PSI.

All PVs are organised into three-level hierarchy.

Hardware PVs are located jn the first level. They are responsible for control over PSC and declare a full interface to it. PV's units are equal to PSC's units (e.g. from -10 V up to 10 V for DAC).

PS-related PVs are located on the second level. All PV-related signals are represented in units of PS inputs/outputs (e.g. kVolts, Amps, etc.).

PVs of the third level allow working in terms of physical units: magnet current, magnet field, angle of bending, etc.

The NSLSS-II name convention declares the structure of a PV name which is very important for PV search and processing.

Hardware PVs use the next form: BR{PSC*NN*}*SignalName-Domain*, where *NN* – PSC identificator, *SignalName* – name of a signal in PSC context, *Domain* – PV type which characterizes a signal direction (in/out) and kind of data (setpoint, measurement, command, status). Examples: BR{PSC01}DO:0-Cmd and BR{PSC12}DAC:0-SP.

PS-related PVs have the following form: BR:*Area*-PS{*PSType*:*PSInstance*}*Signal-Domain*, где Area is a part of the booster ring (arc or straight section), *PSType* – type of PS (e.g. corrector PS, quadrupole PS, bend PS, etc.), *PSInstance* – a name of given PS instance. *Signal* and *Domain* are defined same as at low level. Example: BR:A3-PS{6A:CX2}E:PS-I.

For each level it's possible to develop high level applications, which are operate with corresponding devices: to test and debug PSCs on the first level, to control and tune PSs on the second level and to control the booster system on the third level.

Almost all operations on data in IOC are performed with 10k waveforms, e.g. mathematical treatment, monitoring and etc.

A special check algorithm is implemented in the IOC to provide monitoring of PS operation. This algorithm traces PS operation by comparing current settings and measurements with reference values. The algorithm function diagram is shown in Fig. 2. All operations are performed over waveforms point by point. A reference value is subtracted from a live value and modulo function is applied after that. A tolerance PV is subtracted from the result. The tolerance PV contains a maximal allowed deviation of the live value from the reference one. At least, if one point of the waveform is deviated more than tolerance value then alarm is risen and archiving branch is executed.

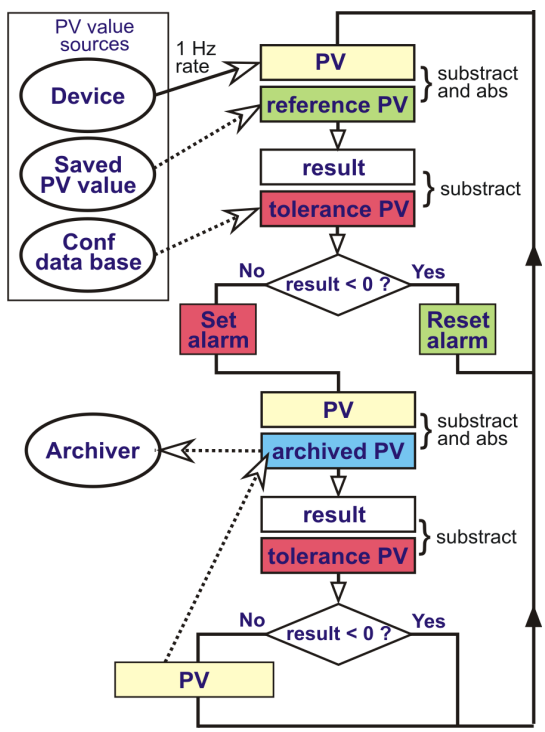

Figure 2: Block diagram of monitoring function.

Booster is a cyclic accelerator therefore reference signals for PS, their first and second derivative values should be limited - these are required operating conditions for most of PS.

A special procedure with transition table was developed to protect PSs from surges while ramp tables are switched. When user uploads a new ramp function to the IOC a transition curve is automatically calculated in the IOC between the determined point of the previous ramp waveform and the beginning of the new waveform. Transition curve is included to a transition table which is performed once at the waveform switching procedure. The beginning of the transition curve is determined at the moment when there is no beam in the machine cycle. In our case the transition curve is a polynomial of 5-th order.

HIGH LEVEL APPLICATIONS

High Level Applications set consists of CSS-based Best OPI Yet (BOY) screens and Jython scripts, and Python-based stand-alone applications and scripts. Most notable BOY screens are:

- Engineering Power Supply Screens (EPSS),
- Booster Status Monitor (BSM),
- Timing Synoptical Screen (TSS).

EPSS OPI set is designed to provide full control and monitoring of the ramping PSs. The set includes several screens for different types of PSs: two bending types (BD, BF), quadrupole, corrector/sextupole, DC septum, two types of pulsed septum and two types of kicker. Each screen is capable of reading and setting relevant PV values and displays measurement and status data on CSS widgets. Data is available in different forms: as 1-sec signal diagram on a main panel and history plot on a data browse panel. All main screens for each PS are launched from a single start screen by pressing a corresponding button.

The BSM application is designed to provide the operator with tools necessary to track the condition of the entire booster system, its subsystems, and subsystems' elements and to allow, when necessary, a quick access to diagnostics data available. In case of any failure, BSM is also provides alarm notice. The application itself displays status data on a screen so that the operator is able to quickly respond to any abnormal condition. The BSM application also provides a possibility to launch external CSS and Python applications that are relevant to the booster status maintenance process.

The Timing Synoptical Screen application is designed to provide an operator with information and utility necessary to control settings of the booster timing system, giving control over the triggering of the booster devices and to allow, when necessary, an ability to quickly disable/enable selected triggers. TSS offers the possibility to manipulate settings of all booster Event Receiver devices at once without switching between individual configuration screens. TSS is to be used during booster adjustment and operation process without the possibility to accidentally interfere with major system-wide settings like changing the event table. TSS also describes the configuration of the booster timing system and can be used as a reference map.

Most notable stand-alone applications are:

- Ramp Manager (RM),
- Booster State Save/Restore/Browse Tool.

RM is developed in Python for handling ramps. It provides a lot of tools and possibilities for operator: graph or digit handling of the ramp function, selectable observation of ADC readbacks and set waveforms in the one plot with scaling feature, polynomial or stepping stone edition of the ramp function, and many others. RM has a possibility to save/restore the selected set of ramp setpoint waveforms to/from a specified file. The saved data are written in a special format to keep polynomial coefficients or step-stone values. RM provides uploading the selected set of ramps to PSC. When the uploaded set includes waveforms for all PSs RM also saves the data to the file and write the name of the file to a special PV, which can be used for matching the file with the uploaded set with the operation set saved by Save/Restore/Browse Tool (see a description below). Also RM provides special features for control of some devices and systems of the booster: RF system, injection septum, injection kickers.

The Booster State Save/Restore/Browse application is designed to provide the operator tools necessary to fully or partially save a consistent (i.e. relevant to a single revolution) condition of the booster system elements and to allow, when necessary, a quick restore to a previously saved state. The application monitors the value of every tracked PV in real-time and composes a change history, which is afterwards collapsed into a consistent value set and saved as a state. Saved states can be browsed and compared with each other or with current PV values with 2D data being plotted for easy visual perception. New PV sets can be composed based on the existing ones.

CONCLUSION

The booster PSs control software have provided a successful performance of integrating tests of PSs [4]. All applications were tested and approved for use in the booster commissioning work. Some applications are in development stage now and will be completed in a couple of months.

REFERENCES

[1] Y.Tian et al., "NSLS-II Power Supply Controller", PAC'11, New-York, April 2011, TUP193, p.1187 (2011).

[2] P.Cheblakov et al., "NSLS-II Booster Power Supplies Control", ICALEPCS'11, Grenoble, October 2011, WEPMS020, p.1018 (2011).

[3] P.Cheblakov et al., "NSLS-II Booster Timing System", ICALEPCS'11, Grenoble, October 2011, WEPMS015, p.1003 (2011).

[4] G.M. Wang et al., "NSLS II Injector Integrated Testing", these proceedings.

ISBN 978-3-95450-122-9