APPLICATION SOFTWARE FOR THE BSP-100 BEAM POSITION MONITOR AT THE APS*

Hairong Shang†, Louis Emery, Robert Soliday, W. Eric Norum, Glenn Decker
ANL, Argonne, IL 60439, USA

Abstract

The BSP-100 beam position monitor (BPM) was commissioned and installed at the Advanced Photon Source (APS) in a fraction of the ring as an upgrade to the present turn-by-turn BPMs. Keeping the same rf front end of the present BPMs, the BSP-100 BPM adds a high-speed analog-to-digital converter and uses a field-programmable gate array (FPGA) to perform the signal processing. The main advantage of the new system is a much better signal-to-noise ratio as all the bunches in the stored beam can now be (selectively) sampled each turn. The implementation requires a much more complex timing control. We report on the high-level software that controls, saves, restores, and compares the timing of the BSP-100 BPM. This software uses Tcl/Tk for the graphical user interface, the SDDS Toolkit for data processing, and SDDS-EPICS compliant tools for saving and restoring.

INTRODUCTION

The Monopulse Beam Position Monitor (MpBPM) is a broadband (10 MHz) beam position monitor designed to measure single-turn and multi-turn beam position at the Advanced Photon Source. Recently, the MpBPM system was upgraded [1] by replacing its aging 12-bit signal conditioning and digitizing unit (SCDU) with the BSP-100 module [2] (i.e., BPM Signal Processor), which consists of high-speed analog-to-digital converters (ADCs) and a field-programmable gate array (FPGA) that performs the signal processing. In addition, the BSP-100 controls the BPM rf receiver's X/Y plane and 0/180 phase selection.

This paper reports on the software used to control the data acquisition block of the BSP-100. Other functions of the BSP-100, such as providing orbit averages and rms orbit motion processing, are not covered.

ACQUISITION CONTROL RAM

The BSP-100 is a standalone Experimental Physics and Industrial Control System (EPICS) input/output controller (IOC) that processes the delta and sum signals of four BPMs, producing one-turn averages of the X or Y data (or both) and a processed sum signal. There are eight high-speed 14-bit digitizers (Analog Device AD6645) running at a sample rate of 88 MHz, which is one-fourth of the APS rf frequency. The acquisition is controlled by an FPGA using an acquisition-control RAM of 3888 32-bit values. This array allows sample-by-sample control of up to 12 storage-ring turns. This is an immense improvement over the old SCDUs, which sampled data only once per turn. The bit assignments for each sample period are shown in Fig. 1.

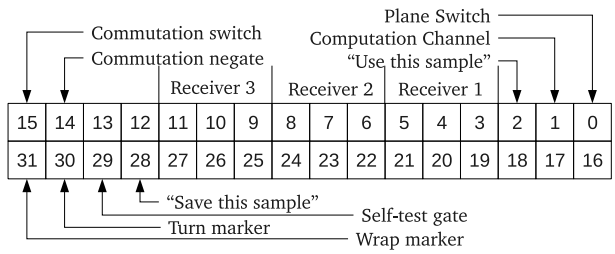

Figure 1: Acquisition control RAM bit assignments.

Some of the bits are specific to a BPM receiver while the rest are global to the four receivers. The assigned bits are "Use this sample," "Plane Switch," "Computation Channel," "Commutation Switch," "Commutation Negate," "Save this Sample," "Self Test," "Turn Marker," and "Wrap Marker," the detailed meanings of which are found in [2]. As illustrated in Fig. 2, the bits are set according to the timing pattern of the stored bunches and the width of the receiver pulses. The RAM is accessed with an 3888-element unsigned integer EPICS waveform PV.

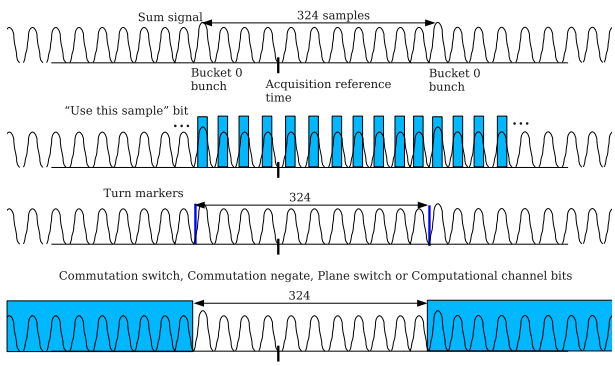

Figure 2: Sketch of a BPM (sum) signal and timing control bits for multiple bunches.

The above indicates that the acquisition control is potentially very complicated. The Advanced Photon Source runs in three bunch pattern modes, and machine studies can use any bunch pattern. To maintain and debug the RAM, a digital-scope waveform recorder was implemented in the FPGA. BPM delta and sum signals are available in eight 4096-sample EPICS waveforms PVs. Another 4096 16-bit integer waveform PV is created taking the lower 16 bits of the control words. These waveforms should be displayed together for inspecting data and control bits.

A control system turn-by-turn time reference is used

* Work supported by U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357
† shang@aps.anl.gov

by the FPGA to start the acquisition sequence. We expect some time offset, different for each BPM, between the passage of a bunch in bucket "0" and the time reference (Fig. 2). Thus the acquisition sequence starts at some bucket other than "0."

The first stage of setting up the control RAM in the FPGA is to create the desired control pattern for the stored bunch pattern. Next one adds a timing offset, generates the values for the RAM elements from these bits, and then sends these array elements to the EPICS control RAM record that resides in the BSP-100.

The control of the hardware switch bits (i.e., plane and commutation bit selection) is constrained by the electrical response of the underlying rf switches, which is of the order of 100s of nanoseconds. Thus sampling should be turned off for a short time after a change of switch state. Normally we switch planes every turn and switch the 0/180 phase commutation every two turns (as with the old SCDU as well). Also, one must decide the number of consecutive samples to use for each isolated bunch. The width of the sum signal profile indicated that is it six (68 ns).

SOFTWARE DESCRIPTION

Given the complications mentioned above, we used Tcl/Tk to develop an interactive waveform graphical user interface (GUI) to view the current control RAM through EPICS, generate predefined control RAM, and edit the control RAM. We also needed a way to save and restore RAM configurations. We approached the project using data and configuration files as primary objects with toolkit programs (SDDS Toolkit [3] and SDDS-compliant EPICS Toolkit [4]) working together to produce the required results. The data processing actions of the GUI itself consists mostly of underlying small tools operating on files.

Some of the basic SDDS-compliant EPICS tools used are: sddswget, which reads values from a list of waveform PVs and writes the data to files; sddswput, which does the inverse; sddscasr, which reads a list of waveform and scalar PVs and writes them to a file, and vice-versa; and sddsgencontrolbits, which is a special-purpose tool that recognizes the structure of the control RAM that will be described later. We created a new SDDS tool, sddsbinarystring, to convert an unsigned long integer into a binary string of 0's and 1's.

The control RAM of a single BSP-100 is manipulated with a single instance of the MpBPMWaveformViewer GUI. Figure 3 shows the GUI for half-sector S38B.

Usually two waveform adjuster displays are visible, the upper one showing one of the control bit waveforms and the lower one showing a BPM data scope waveform. In all, the GUI displays 38 waveforms, which necessitates a tree of tabs, as shown below:

```
Display tabs
    Scope
       BPM number (multiple tabs)
          Top: Use this sample, plane, computation channel
          Bottom: scope delta signal, scope sum signal,
                timing configuration controls with presets
    Control RAM
       BPM number (multiple tabs)
          Top: sample, plane, computation channel
          Bottom: scope delta signal, scope sum signal,
                timing configuration controls with presets
    Common bits in RAM
       Top: scope bits
       Bottom: RAM bits
    Status messages
Action tabs
    get waveforms
    global timing configuration controls with presets
    save waveform
    load waveform
```

The first two display tabs are self-explanatory. The "Common Bits in RAM" displays the RAM control bits common to all fours BPMs.

The first action tab is the "Get Waveform PVs" tab, consisting of buttons that cause the BSP-100 to update the EPICS waveform PVs and to load the waveforms values in the waveform widgets. One could also load and review control RAM data that was stored in a file.

The next action tab "Timing configuration controls with presets" (confusingly called "Read/Set Common Presets" in the GUI) is the core of the application. For simplified and realistic user-beam operations, we have decided to somewhat limit the parameter space of the control RAM by characterizing the control bits with a few timing configuration parameters. Thus new control RAM are generated based on timing configuration parameter selections, such as: the plane switching mode, a sampling mode, computation channel switching mode, and an optional choice of a bunch pattern from the injection filling pattern database. The reduction of parameter space allows one to write a tool such as sddsgencontrolbits to generate the control RAM. The full list of timing configuration parameters is shown at the bottom of the GUI in Fig. 3.

For normal APS operation with 24 bunches, the plane mode would be "x/y every turn"; the sample mode, "Bunch pattern"; the computation channel switching mode, "0/180 every two turns"; transition dead time, 70 clock cycles; the string "0+24S Fill from 0 mA" would be typed into the bunch pattern (this is a valid string value from the filling pattern database); samples per bunch, 6; turn marker offset (i.e., time offset from bucket 0), 199 clock cycles; and turns per wrap, 4. The resulting control RAM would then be loaded into the BSP-100.

At the same time the values of the timing configuration parameters are written to several scalar place-holder PVs. These PVs can then be displayed in a standard EPICS tool (i.e., medm) as indicators of what control RAM configuration is currently loaded. When archiving the control RAM, these scalar PVs must be saved alongside, again to help later identify the timing configuration.

There is an option of selecting one of many timing configuration parameters sets ("presets" selection box), most of which have been used in BPM noise studies and normal operations. There is a button for editing the sets.

The waveform display widgets are editable. One could in principle edit the control bits before they are assembled

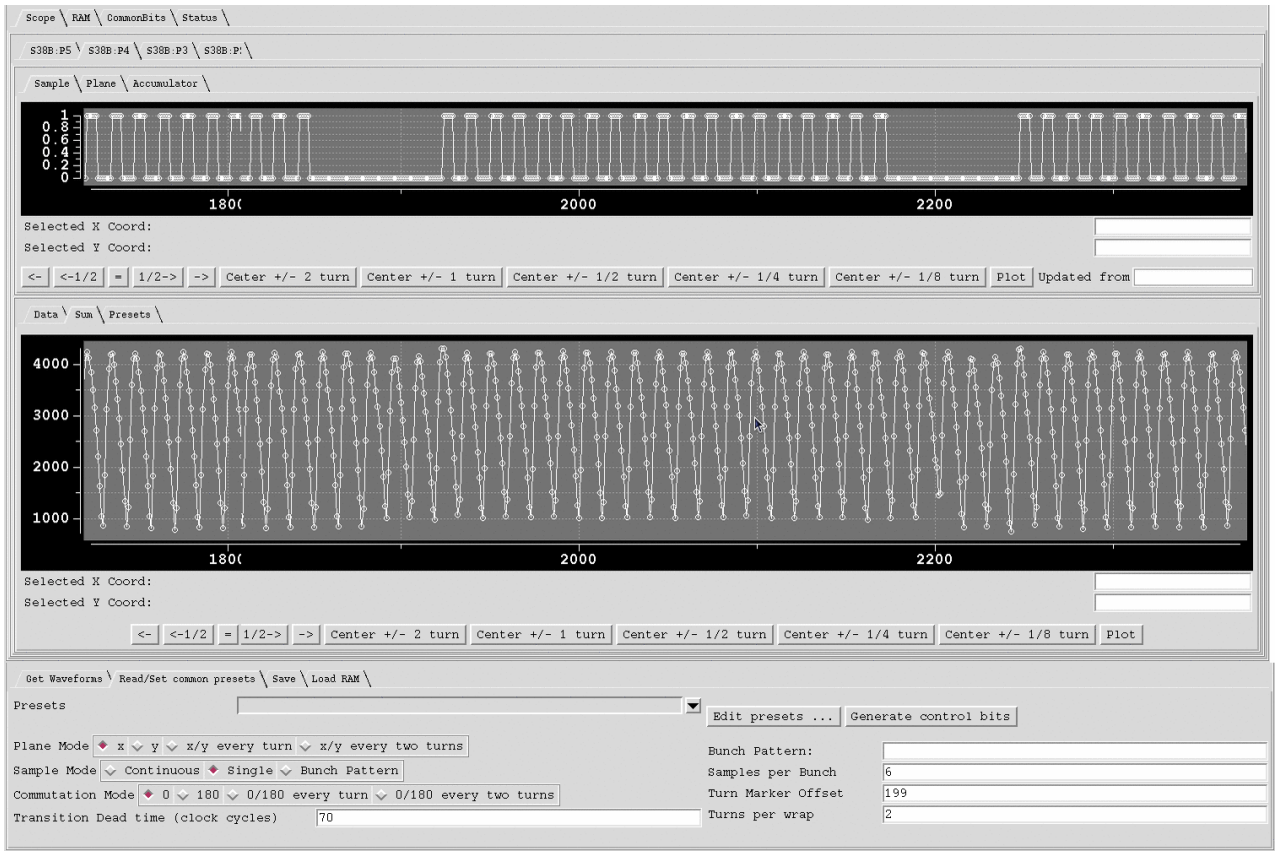

Figure 3: MpBPM Waveform Viewer and Controller for S38B showing the "Use this sample" bit with BPM sum signal.

as control RAM and written to EPICS; so far there hasn't been the need in real beam applications. Actually, that would have been the default mode of setting up the control RAM if we hadn't thought of using timing configuration parameters, and the special tool that goes with it.

The same controls for generating a control RAM is available on a single BPM basis in one of the tab layers of the RAM and Scope tab trees.

Action tab "Save" has one button for saving the control RAM as represented in the edited control bits in an archive. Action tab "Load" has one button as well, which loads the edited control bits into EPICS RAM waveform.

When the GUI is launched, the current EPICS RAM and scope waveform are read and displayed together with the current scope data and sum signals. The user can generate a new but related control RAM by selecting a different bunch filling pattern preset or changing one of the timing configuration parameter values.

The C program sddsgencontrolbits was written to handle the control RAM words and the separated bits for the GUI. One usage mode is to read the control RAM PV and generate a file of control bits; another is to use timing configuration parameters on the command line to generate a file of control bits and RAM. This tool has also been used in script-based experiments where one or more aspect of the acquisition (e.g. time delay) is to be modified in a systematic way.

Once the desired timing is achieved and loaded into EPICS along with the timing configuration parameters, all of these are saved using APS's configuration archiver SaveCompareRestore. This GUI allows the user tp restore the full or partial list of PVs and do comparisons between saved configurations. One can also design a custom script for comparing the saved RAM waveforms, though this hasn't been done yet, nor has it been necessary. For now, one compares the timing configuration parameters.

SUMMARY

We presented a GUI that allows the user to set up and visualize the timing control and data collection of the new APS storage ring BPM BSP-100 module. It is possible to make full use of the flexibility of the FPGA-based design. Other existing control room software provides the save, compare, and restore functions.

REFERENCES

[1] A. Pietryla et al., Proc. of PAC07, p. 4390 (2007).

[2] W. Norum, private communication.

[3] http://www.aps.anl.gov/Accelerator_Systems_Division/Operations_Analysis/manuals/EPICStoolkit/EPICStoolkit.html

[4] http://www.aps.anl.gov/Accelerator_Systems_Division/Operations_Analysis/manuals/SDDStoolkit/SDDStoolkit.html