NEW FEATURES IN THE SDDS-COMPLIANT EPICS TOOLKIT *

H. Shang, R. Soliday, L. Emery and M. Borland
Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

This paper introduces new features and programs developed to enhance various aspects of the SDDS-compliant EPICS toolkit. A new optimization program, **sddsoptimize**, was added to the toolkit; it employs the Simplex and 1-D scan methods and can be used for both EPICS and non-EPICS optimizations. Several new data logging programs were also developed, including a new, more flexible glitch logger that logs data before and after a glitch occurs. Another new data logger logs data every time the value of a process variable changes. With the data generated from this program, it is now possible to restore settings from any arbitrary time without the need for a snapshot of the system. Another new addition is the capability of saving and restoring waveform process variables. In addition to these new features, performance improvements have been realized in all the toolkit programs by replacing EZCA calls with low-level channel-access calls. Some of the toolkit programs have been upgraded to run on vxWorks to achieve higher performance.

INTRODUCTION

The SDDS-compliant EPICS toolkit is a set of software applications (tools) for the collection or writing of data collected from Experimental Physics and Industrial Control System (EPICS) database records. Although most of the applications essentially do rather simple operations, the combination of these applications and others from the SDDS postprocessing toolkit allows arbitrarily complicated analysis of data and control of the accelerators at the Advanced Photon Source (APS). These tools are general enough to be applied to devices other than accelerators, provided that the devices are under control of EPICS. One can regard the EPICS tools as the layer between the EPICS control system and more functional analyzing tools and scripts, with the SDDS protocol files as an intermediary. EPICS database records are referred to as process variables (PVs), each one having a unique name. PVs can be analog, digital (two or more states), or entire waveforms, depending on the I/O configuration. APS has developed over twenty SDDS-compliant EPICS programs. These programs can be categorized into three functional groups: (1) configuration save and restore, (2) data collection, and (3) process control [1]. The purpose of this paper is to review enhancements, new features, and new programs developed for the SDDS-compliant EPICS toolkit.

* Work is supported by U.S Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

ENHANCEMENT OF CHANNEL ACCESS

EZCA (Easy Channel Access) provides a simplified interface to channel access for C programs. As a generic channel access facility, EZCA is used in many EPICS programs. The main shortcomings of EZCA are instability and speed: using EZCA, channel access errors occur unpredictably and channel access can be very slow. To overcome these problems, all EZCA calls in the SDDS-compliant EPICS toolkit have been replaced by low-level channel-access calls.

NEW FEATURES AND PROGRAMS FOR PROCESS CONTROL

This section describes new features added to the program **sddscontrollaw** and a new optimization program, **sddsoptimize**. Both of these programs are used for process control.

sddscontrollaw

This program provides a generic feedback capability for EPICS. As an accelerator control tool, **sddscontrollaw** is applied to (1) maintain constant energy and trajectory from the linac using an experimentally-derived matrix, (2) correct the orbit in the particle accumulator ring (PAR), (3) steer and maintain the storage ring (SR) orbit using a theoretical matrix and an orbit despiking filter, (4) maintain the APS ring injection trajectory, (5) regulate power levels from linac klystrons, and (6) regulate output of pulsed magnets using pulsing history. It is also used by a Tcl/Tk script that allows on-the-fly creation of one-readback, one-actuator control loops [1].

However, **sddscontrollaw** could not perform fast orbit correction because it was workstation-based and connected to each control and readback PV as a scalar. In order to perform fast orbit correction, the following new features were added:

- Frequency-band overlap compensation [2]

The APS real-time orbit feedback system (RTFS) operates in parallel with the DC orbit correction (DC OC) **sddscontrollaw.** The output of the RTFS is high-pass filtered with a cut-off frequency of 0.02 Hz to avoid interfering with the DC OC. If the DC OC has a bandwidth lower than that of the RTFS high-pass cut-off frequency, then a part of the orbit noise spectrum is not corrected. If the DC OC has a bandwidth higher than the RTFS high-pass filter cut-off, then the bands overlap (i.e., corrections doubled) and the orbit noise increases in the overlapping region. We developed a feedforward scheme in **sddscontrollaw** to allow operation in the latter overlap condition. At each iteration the expected orbit change from the DC OC is added to the value of the RTFS BPM

set points, thus preventing the RTFS from reacting to the DC OC corrector change. An additional file for the corrector-to-BPM set point matrix is required, since the set of BPMs in the RTFS may be different from those in the DC OC. Digital filtering of the BPM set points is also possible to account for corrector response. With this compensation, a large amount of frequency-band overlap is now workable and serves to eliminate the possibility of a gap in the total OC response spectrum.

- The ability to read and set waveform PV values was added to the program. Waveform PVs can be used to provide rapid access to arrays of scalar PVs of similar properties. In this case, waveform PVs are used to provide collections of corrector power supply set points and of beam position monitor readbacks.
- EZCA channel access function calls were replaced with the low-level channel access functions. Testing has shown that low-level channel access is more efficient and reliable compared to the EZCA channel access.
- The workstation version of the program **sddscontrollaw** is limited in performing DC OC for the APS storage ring (SR) because it has to contact many different IOCs at each iteration to send and receive the BPM and corrector data. To overcome this limitation, the program was ported to VxWorks and hence can run in IOCs using that operating system. For the APS DC OC application [3], the IOC in question has special hardware to send and receive vectorized corrector and BPM values to and from a memory backplane, which moves the data to and from the other related IOCs at a very high rate.
- **sddscontrollaw** has a feature that suspends or resumes correction based on whether test conditions are satisfied by a group of PVs. For example, DC OC is inhibited when beam is lost, when a corrector changes by a large amount, when a BPM has a bad status, etc. The necessary testing of so many PVs is time-consuming, so we developed a special-purpose program, **sddspvtest**, that performs this function and sets a single PV with the result. **sddscontrollaw** then need only test this one PV.

As a result of these new features, **sddscontrollaw** is able to correct the APS orbit at a rate of 20 Hz [3]. In addition, **sddscontrollaw** is used in linac IOCs to perform klystron power regulation. Having the same code running on a workstation and in the IOC both reduces code maintenance requirements and expedites development of new features and control loops.

sddsoptimize

This is a new addition to the SDDS control tools. It provides a generic optimizer that can be used for cases where feedback is not applicable. The main features of the optimizer are detailed below:

- The optimization criterion is the rms value of one or more PVs, or else the value obtained by running a script. In the former case, one may optionally assign weights, target values, and tolerances for each PV. In the latter case, the measurement script can be used to perform more general operations, which may or may not involve accessing PV values.
- Two optimization methods are provided: simplex and successive 1D optimization (also called 1D scan). Simplex is a multidimensional minimization method that requires only function evaluations [4]. It is frequently the best method if the computational burden is small. By default, our simplex method makes explicit use of a one-dimensional minimization algorithm as a part of the computational strategy, since this often will make the optimization proceed faster; this can be disabled in cases where it is found not to help. The successive 1D scan method allows minimization of the target with respect to each parameter separately and in turn. The main disadvantage is that if the optimal changes of the parameters are mutually dependent, this method may converge very slowly toward the minimum. Nevertheless, it runs efficiently when the variations are quasi-independent.
- Setting the values of PVs can be replaced by running a "variable script" (given by the varScript option) so that the program can effectively set PVs in an arbitrarily complicated fashion or even perform optimizations that do not involve PVs. There are no CA calls in **sddsoptimize** if both variable and measurement script are provided, so non-EPICS optimizations are possible. For example, one can optimize the results of a simulation.
- The program performs minimization by default and will perform maximization if the "-maximize" option is given.
- **sddsoptimize** can be used to adjust knob PVs, which are predefined linear combinations of PVs. Examples are knobs for orbit bumps or ganged timing control for a set of kicker magnets.
- To make the optimization robust, a series of validity tests on PV values are implemented by means of an additional SDDS file containing the names of PVs and their corresponding limit values. The optimization is suspended if one of the tests fails. This can be used to avoid processing invalid data and to terminate the program if it adjusts settings beyond a safe or reasonable range.
- The optimization can be stopped at will by the user using ctrl-c (i.e., the UNIX SIGINT signal). The best settings obtained so far will then be implemented before the program terminates.
- **sddsoptimize** optionally logs settings and results to an SDDS file. This file can be used to view results during or after an optimization, and also to set up a new optimization.

As a result of these features, **sddsoptimize** has been applied to the APS SR for (1) maximizing injection efficiency, (2) storage ring beam x-y coupling minimization, (3) booster-to-SR rf phase adjustment to center injected beam in the rf bucket, and (4) on-axis injection setup and closed bump setup. **sddsoptimize** has also been used in the APS linac for beam-based optimization of rf phase and power, and in the PAR for maximizing capture efficiency. It was also employed for fitting APS linac bunch compressor measurements using

the simulation code **elegant** to evaluate the values of the fit (through particle tracking) at the experimental points.

NEW PROGRAMS IN THE DATA COLLECTION TOOLKIT

Data collection programs provide various types of data logging into SDDS files [1]. Three new programs, **sddsglitchlogger, sddslogonchange,** and **sddssynchlog** have been developed and are discussed in this section.

sddsglitchlogger—This program is able to log data before and after a glitch occurs, where a glitch is a sudden change in readings from a process variable. Although another data logging program, **sddsmonitor**, provides a similar capability, **sddsglitchlogger** is much more flexible. For example, with **sddsglitchlogger** one can specify multiple glitch conditions and corresponding multiple output files. This allows one **sddsglitchlogger** process to do the work of many **sddsmonitor** processes, thus decreasing loads on IOCs. As in sddsmonitor, the PVs to be logged are defined in an input file. The trigger PVs can be defined either in the input file as parameters or in a separate trigger file. In the former case, many output files may be specified for different sets of logged PVs, each triggered by a different, single PV. If multiple trigger PVs are required, then a separate trigger file is used. There are three types of triggers, detailed use of which is described in the manual page: (1) Alarm-Based Trigger – Occurrence of an alarm of a specified severity or severities results in dumping of data; (2) Transition-Based Trigger – Data is dumped when the specified PV transitions through a certain level from above or below (as selected); and (3) Glitch-Based Trigger – The trigger fires when the difference of the PV value from the average of recent values is greater than the glitch threshold. To make **sddsglitchlogger** more robust, a conditions file is supported that lists conditions that must be satisfied at each time step before the data can be logged. This prevents accumulating and logging invalid data.

sddslogonchange—The program **sddslogonchange** logs data every time a PV's value changes. With the data generated from this data logger it is possible to restore settings from an arbitrary time without a snapshot from the system. With the introduction of the program **sddslogonchange** it is now possible to log slowly-changing PVs more efficiently than previous SDDS data loggers. Data loggers such as **sddsmonitor** and **sddslogger** log every PV at a specified iteration, while **sddslogonchange** logs a PV only when its value changes. Therefore no unnecessary data is logged. By logging the initial values of every PV it is possible to determine each PV's value at any given time from **sddslogonchange**'s output file. As in **sddsalarmlog**, disk space efficiency is enhanced by using the SDDS array feature to store PV names in a coded format, obviating the need to store the same string each time a PV changes.

sddssynchlog— EPICS is by its nature an asynchronous control system. This can present problems in data collection and correlation analysis. The program **sddssynchlog** addresses this issue by collecting time-stamped data from many PVs, then organizing the data to line up the time stamps. This is necessary given that different IOCs may have different loads and hence serve data at different rates or even with gaps. Even in such an environment, **sddssynchlog** provides data suitable for reliable correlation analysis. The program was used extensively in the APS linac, helping to pinpoint problems with BPMs and to find sources of beam motion.

NEW PROGRAMS IN THE CONFIGURATION SAVE AND RESTORE TOOLKIT

Previous SDDS-based configuration save and restore programs only work for scalar PVs. In order to be able to handle waveform PVs, two new programs **sddswget** and **sddswput** have been developed for the configuration save and restore toolkit, as will be described in this section.

sddswget—This program is developed to provide a convenient and fast method for collecting waveform data, including character and string types. The input file to **sddswget** specifies the waveforms to be read; this may be done with a file that is compatible with **sddswmonitor**, or using a **sddswget**-specific SDDS file. The latter is formally identical to the *output* file produced by **sddswget**, which is convenient in many cases. Optionally, one may specify the names of the PVs on the command line.

sddswput—This program was developed to provide a convenient and fast method for setting waveform data from SDDS files. Like **sddswget**, it supports character and string types. It accepts as input the output files from **sddswget.**

sddswget and **sddswput** are used to retrieve booster corrector ramp tables, initialize waveform PVs in the DC OC, and save and restore waveform PVs in the data pool IOCs. **sddswget** and **sddswput** are also used in the rapid reconfiguration of the APS monopulse BPM trigger timing systems for different bunch patterns.

REFERENCES

[1] M. Borland, L. Emery, Proceedings of the 1995 ICALEPS Conference, Chicago, Illinois, pp. 653-662 (1996).

[2] C. Schwartz, L. Emery, Proceedings of the 2001 Particle Accelerator Conference, Chicago, IL, pp. 1234-1236 (2001).

[3] R. Soliday, M. Borland, L. Emery, H. Shang, "Use of a Simple Storage Ring Simulation for Development of Enhanced Orbit Correction Software," these proceedings.

[4] W. Press, S. Teukolsky, W. Vettering and B. Flannery, "Numerical Recipes in C," Cambridge University Press, p. 408 (1992).