INTERFACING OF PERIPHERAL SYSTEMS TO EPICS USING SHARED MEMORY

E. Tikhomolov, TRIUMF, Vancouver, Canada

Abstract

Interfacing of peripheral control and data acquisition systems to an EPICS-based control system is a common problem. At the ISAC radioactive beam facility, both Linux-based and Windows-based systems were integrated using the "soft" IOC, which became available in EPICS release 3.14. For Linux systems, shared memory device support was implemented using standard Linux functions. For Windows-based RF control systems, the soft IOC starts by a separate application, which uses shared memory for data exchange with the RF control applications. A set of DLLs exposes an API for use by the application programmer. Additional features include alarm conditions for read-back updates, watchdogs for each running application, and test channels.

INTRODUCTION

Different user groups at ISAC develop their specific subsystems for automation of experiments, data acquisition and control of peripheral devices. The ISAC Controls group has the task of integrating all these different subsystems into the running EPICS-based control system.

The beam diagnostic systems (BDS) installed in ISAC are implemented in VME crates with Input/Output Controllers (IOC) running Scientific Linux [1]. Such IOCs are referred to as LIOC below. Data acquisition and hardware control applications are running on LIOC and must interface to EPICS clients such as the Extensible Display Manager (EDM [2]), Matlab applications with Matlab Channel Access (MCA [3,4]), and others.

The RF control system (RFCS) [5] in ISAC-II runs in dedicated PCs under the Windows operating system. Several control applications are running at the same time on the same PC. All applications are implemented with Borland C++ Builder. Besides local control, it is necessary to implement remote control of the running hardware. EDM can be used for this purpose but it is necessary to implement a software middle layer between the RFCS and the EPICS clients.

The Laser control system (LCS) at ISAC uses commercial software. This application is used as delivered and does not need to be developed and changed by the Laser development team. It needs, however, to be operated remotely by EPICS clients.

These systems are running on different platforms and have different requirements for operations. All these three systems could, however, be integrated using the EPICS "soft" IOC. The soft IOC provides communication with EPICS clients and exchanges data with other running applications via shared memory. The soft IOC starts as a separate application which is implemented and maintained by ISAC Controls group. In addition an application programming interface (API) was developed for the BDS, RFCS, and LCS developers which provides access for their programs to the shared memory.

LINUX AND WINDOWS SOFT IOC WITH SHARED MEMORY

Linux BDS Soft IOC

For the BDS we used the standard EPICS way of adding new devices to the system. A specific application running on LIOC defines a virtual "card" which provides all required functionality: turning on and off, starting data acquisition, collecting data etc. Virtual "card" number is associated with the sequential number of the shared memory segment. The shared memory segment is organized as a set of arrays of different EPICS record types. The maximum number of elements in each array is predefined. Mapping between EPICS PVs existing in the soft IOC and in the shared memory segment is arranged as follows: the high digit of the signal number defines the record type; lower digits define the index in the array. For example, the string "#C2 S3004" means "card" 2, record type 3 (which is in our implementation an aiRecord), and signal/index 4.

During soft IOC initialization each EPICS record checks whether the shared memory segment ("card") is initialized. If not it creates a new one and puts the PV name into the shared memory record field with the related index/signal. The device support for each record type automatically provides exchange of data between EPICS PVs and the shared memory counterparts.

The soft IOC starts on LIOC as a separate application when the Linux operating system boots. After initialization of the soft IOC the BDS application is started using a system call from a subroutine record. The application is running an infinite loop (with period of 0.5 sec) in which it gets or sets values in shared memory. A Set/Get API is provided by the ISAC Controls and is part of the distributed libraries. Set/Get functions use as an argument the PV name to find the corresponding index for reading/writing a value into the shared memory segment.

We describe the basic operations of BDS using an EDM display which is used in daily operation of this system for control of silicon detectors (Fig. 1). In the top left portion of Fig. 1 are control and read-back of beam line devices provided by ISAC Control System IOCs. The remainder of the display is controlled by soft IOC. The main control operations are as follows:

- Buttons *Start Appl, Stop Appl* are used to start/stop a data acquisition application implemented by the beam diagnostic developers who use the API

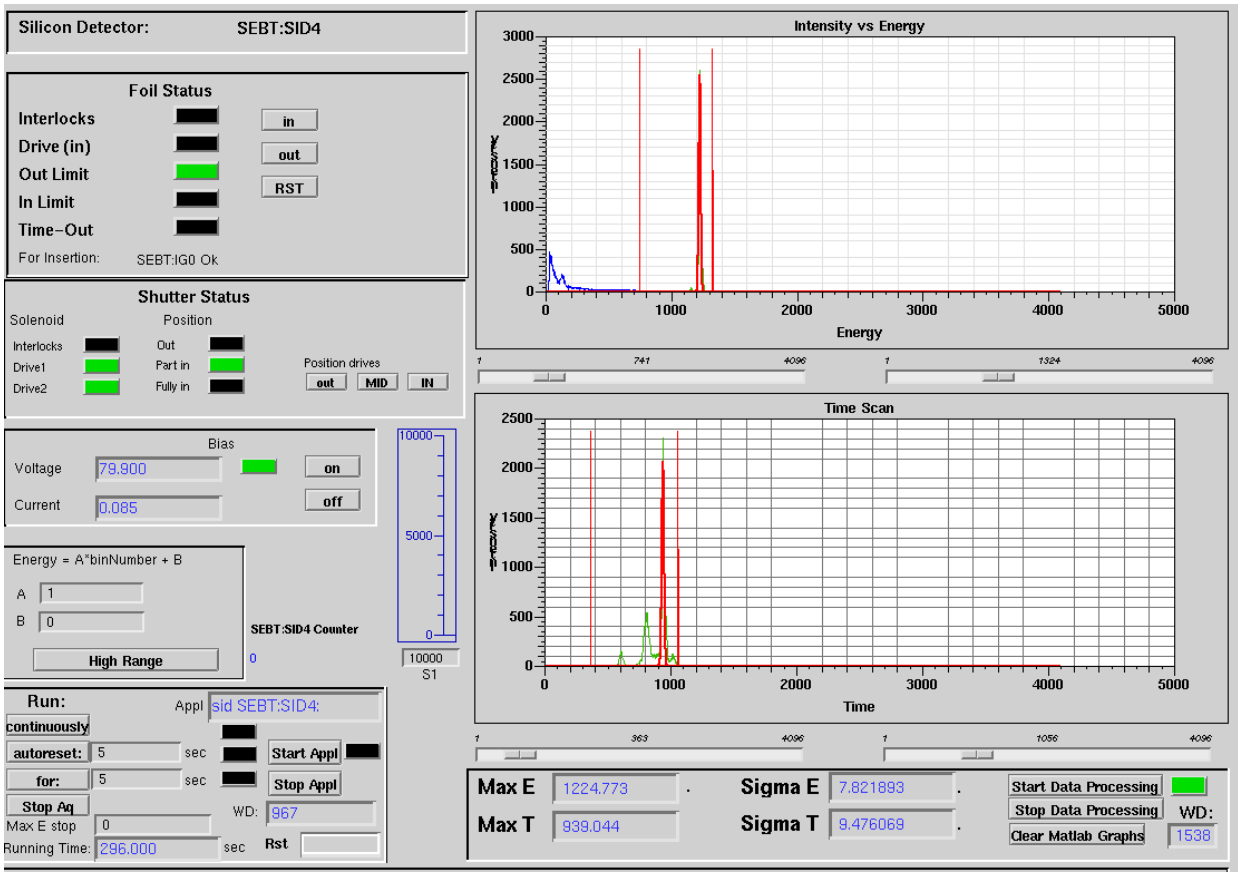

Figure 1: EDM screen used to control silicon detectors at ISAC and ISAC-II.

provided by the ISAC Controls. The buttons message an EPICS subroutine record which starts the BDS application using a Linux system call. The application name (sid) and the PV prefix name (SEBT:SID4:) are specified in the EPICS database. When the application starts, the adjacent LED turns green and the watchdog starts counting. In this state the application loops awaiting commands for data acquisition.

- Different modes for collecting data can be specified by pressing buttons *Run continuously*, *Run with autoreset* (period is specified in adjacent box), *Run for* specified period of time. *Stop Data Acquisition* returns application to the waiting state.
- The right part of the screen is used for visualization of data by using a Matlab application with Matlab Channel Access [4].

The EPICS database is created by using the Capfast tool [6] using ISAC Controls device database and associated web application [7].

Windows RFCS TRIUMF Soft IOC

Several RF Control applications run under the Windows operating system on dedicated PCs. Initially the RFCS was integrated into EPICS by using a customized TRIUMF version of the Portable Channel Access Server [8]. When the soft IOC became available in EPICS release 3.14 it was decided to upgrade the system to use the soft IOC to implement the RFCS-EPICS interface.

The RFCS soft IOC starts as a separate Windows application TRSOFTIOC. All RFCS applications run at the same time and provide the functionality of the system as a whole. For the shared memory interface, one common shared memory segment is defined. Its organization is an array of "generic" EPICS records. Each element of the array has such EPICS components as name, value, drive and alarm limits, etc. At the same time the array element has specific non-EPICS components like watchdog, local/remote flags which indicate whether a local application or a remote EPICS client changed the value. Each PV in the soft IOC is associated with an element in the array, and the index of the array is used as a PV handle.

After starting of the soft IOC the shared memory segment is initialized and a thread starts which updates values between the soft IOC and shared memory. The thread checks the flag local/remote update and sets the value either in shared memory or in soft IOC.

The RFCS developers are provided with an API which they use in their control applications. The description of the API can be found in our previous paper [8]. Functions are deployed in the form of DLL files which are used by both TRSOFTIOC and RFCS applications. Thus it is possible to create a shared memory segment available for both sets of applications using pragma instructions. We

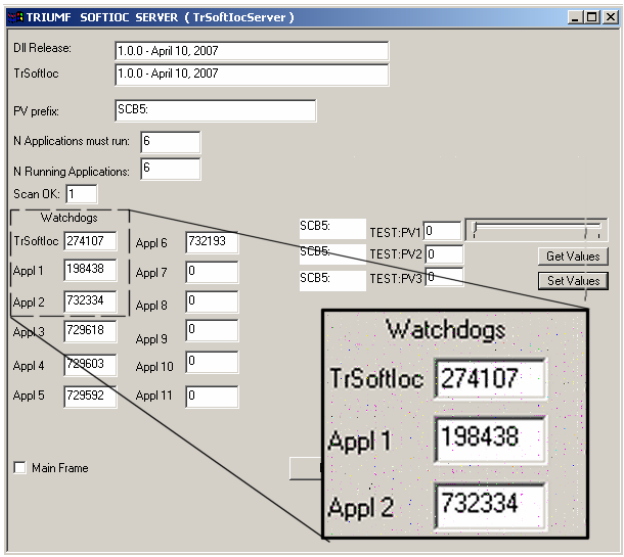

Figure 2: TRIUMF Soft IOC application with shared memory for Windows.

also tested a memory-mapped file model which can be used for the same purpose. This model is more convenient when it is necessary to create several shared memory segments like in the case of Linux BDS soft IOC. However, for RFCS when only one segment is needed we used the pragma model.

The development path for the RFCS applications is dynamic and independent from the ISAC EPICS interface. The addition of variables to the RFCS which must be available as new EPICS PVs should be done without direct participation of ISAC Controls. To decouple the reliance on ISAC Controls, the RFCS developers are provided with a configuration XML template file which contains information about PVs. This file is then maintained by the RFCS developers. Starting of the soft IOC occurs in two steps. A console Windows server application TRSOFTIOC generates the EPICS database "on the fly" from the XML file and then starts a soft IOC that uses this database. At the same time shared memory is initialized.

Figure 2 shows the console window of TRSOFTIOC. It has several watchdogs which indicate running applications. The same watchdogs are shown in EDM screens available for remote operators. When one of the RFCS applications stops an alarm is raised on the remote operator's machine. The application has a local/remote control button which switches the mode of operation during tests and diagnostics. Reliable monitoring of the RFCS requires that the RF programmer updates all read-back variables at regular intervals. If read-backs are not updated for a specified period of time the TRSOFTIOC flags them as “invalid”. The TRSOFTIOC also provides test channels which are used for diagnostics and troubleshooting.

LCS “ Minimalistic” Soft IOC

The LCS developers use commercial software and do not need to change the number of EPICS PVs. Here a minimalistic approach is acceptable. The soft IOC starts as usual, as a separate application. EPICS PVs are specified in a database file in accordance with variables used by the LCS application. Additional code was added to the existing software to communicate with the soft IOC whenever the application starts. Data exchange between the soft IOC and the user application occurs using shared memory as in the case of RFCS.

DISCUSSION

Different local control subsystems were integrated into the ISAC control system using a shared memory interface. Each local control system has different requirements for running applications, development flexibility and interaction with the EPICS-based system. These requirements can easily be accommodated by variations in the implementations of the shared memory interface. These variations, however, are transparent to the EPICS clients and thus create the common operational environment for different subsystems. .

ACKNOWLEDGEMENTS

The authors thank J. Richards for support in installation and tests of the described systems, R. Keitel for valuable suggestions for code development, G. Waters for generous sharing of his experience in working with new EPICS releases and soft IOC, R. Nussbaumer for his advices on solving Linux issues, and Victor Verzilov who started investigation on integration of BDS into EPICS at ISAC-II.

REFERENCES

[1] Scientific Linux, https://www.scientificlinux.org/
[2] John Sinclair, http://ics-web1.sns.ornl.gov/edm/
[3] MCA, http://ics-web1.sns.ornl.gov/~kasemir/mca/
[4] E. Tikhomolov, “Processing And Visualization of Epics Data with Matlab Applications”, this proceedings.
[5] K. Fong *et al.*, “Status of RF Control System for ISAC-II Superconducting Cavities”, Linac2004, Lübeck, p.450.
[6] Capfast, http://www.phase3.com/
[7] R. Keitel, J. Richards, and E. Tikhomolov, “Upgrade of the ISAC Device Database from Paradox to PostgreSQL”, ICALEPCS’03, Gyeongju.
[8] E. Tikhomolov, G. Waters, R. Keitel, “EPICS Portable Channel Access Server for Multiple Windows Applications”, ICALEPCS03, Gyeongju.