NEW DEVELOPMENTS ON EPICS DRIVERS, CLIENTS AND TOOLS AT SESAME

I. Saleh, Y.S. Dabain, A. Ismail, SESAME, Allan, Jordan

Abstract

SESAME is a 2.5 GeV synchrotron light source under construction in Allan, Jordan. The control system of SESAME is based on EPICS and CSS.

Various developments in EPICS drivers, clients, software tools and hardware have been done. This paper will present some of the main achievements: new linux-x86 EPICS drivers and soft IOCS developed for the Micro-Research Finland event timing system replacing the VME/VxWorks-based drivers; new EPICS drivers and clients developed for the Basler GigE cameras; an IOC deployment and management driver developed to monitor the numerous virtual machines running the soft IOCs, and to ease deployment of updates to these IOCs; an automated EPICS checking tool developed to aid in the review, validation and application of the in-house rules for all record databases; a new EPICS record type (mbbi2) developed to provide alarm features missing from the multibit binary records found in the base distribution of EPICS; a test of feasibility for replacing serial terminal servers with low-cost computers.

INTRODUCTION

SESAME consists of a 22 MeV Microtron, an 800 MeV Booster Synchrotron and a 2.5 GeV Storage Ring. Control System Implementation uses (EPICS) base R3.14.12. Servers are implemented as EPICS Input/ output Controllers (IOCs). Clients are implemented using a custom build of Control System Studio (CSS) based on V.3.16. Siemens S7 PLC controllers are used for the machine interlocks. An Allen Bradley PLC controller is used for the Personal Safety System (PSS). VME hardware is used for the timing system. Development and administration platforms use Scientific Linux 6.4. A Git version control is used to track development and documentation. All clients, servers, and controllers are connected to an isolated machine network. There are twelve virtual servers reserved to run the IOCs, archive system, alarm system and Git repositories.

The control systems have been implemented for the Microtron, Transfer Line 1 (TL1) and Booster. The Booster's control system is divided into seven subsystems: vacuum, power, RF, diagnostics, cooling, timing and Personal Safety System (PSS). Each control subsystem consists of one or more clients, servers, and controllers [1]. This paper will focus on the developments on EPICS drivers, clients and tools at SESAME.

TIMING SYSTEM DRIVER

The timing system for the Booster consists of one event generator VME-EVG230 and one event receiver VME-EVR230, both of which are connected to the EPICS network over TCP/IP. New Linux-x86 EPICS drivers were developed for the timing modules from Micro-Research Finland.

The VME-EVG230/VME-EVR230 are traditionally controlled over the VME bus. Both modules can also be programmed over the available Ethernet port however. Building an EPICS driver for controlling the timing modules over Ethernet instead of the VME bus has the following benefits:

- Drops the dead weight: The VME crate, the VME CPU card, the RTOS that runs on the CPU card (along with any required licenses), and the debug terminal that connects to the CPU card are no longer needed.
- Lowers the cost of implementation: A direct consequence of the point above.
- Confines the required development skills to Linux/x86 platforms. Knowledge in VME-bus, VxWorks, or any other RTOS/OS is not required.
- Maintains coherency in the control infrastructure. This point may be specific to SESAME only since all of the IOC's at SESAME run on Linux/x86 platforms.

The device layer uses the traditional asynchronous processing model of EPICS support modules. The driver layer uses the remote programming protocol of the timing modules to control them. This protocol uses UDP. The drivers add feedback, retransmissions, and timeouts to create reliable communication over UDP.

Both drivers are LGPL'd and can be found as public domain on Github [2]. The client of the timing system is implemented in Control System Studio (CSS) and is shown in (Fig. 1).

ISBN 978-3-95450-148-9

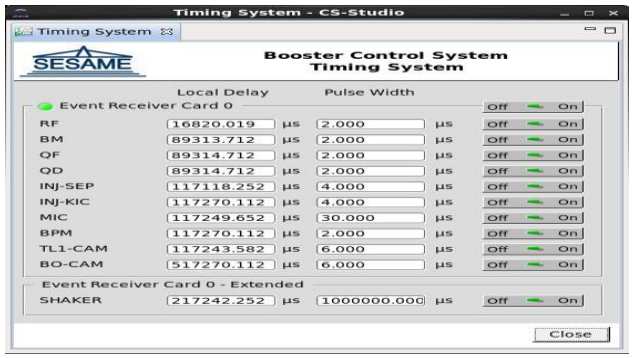

Figure 1: CSS Timing OPI.

BASLER GIGE CAMERA DRIVER

New Linux/x86 EPICS driver was developed for the GigE camera series from Basler. The device layer uses the traditional asynchronous processing model of EPICS support modules. The driver layer uses the Pylon libraries from Basler to control the cameras over Ethernet. Once configured, the driver starts a separate server thread for each camera. Each server thread listens for commands such as image capture or gain and exposure settings. A set of mutexes and signals are used to synchronize operation between the various threads.

BASLER GIGE CAMERA CLIENT

To connect to the Basler GigE camera driver, a custom client was built at SESAME utilizing the EPICS channel access libraries. The client is responsible for providing a graphical user interface to render the camera's video stream possibly doing some image post-processing like colour mapping and to control the camera image settings.

The camera client uses the EPICS channel access libraries to monitor and control the various records provided by the driver. A waveform record contains the grayscale image data and various analog input/output records provide access to camera's gain, exposure, ROI and trigger source. Once the camera client is started it connects to the driver and enables the camera. The camera is usually disabled if unused to save network bandwidth. Once the camera is enabled, the waveform record containing the grayscale image starts updating as the camera is triggered. On each update, a call-back is called in the client to copy the image data to one of two buffers used to render the image. One of the buffers is being read from while the other is written to. And once the image buffer is completely written to and ready, a switch is done so that the next render uses the new image data. The double-buffering is used so that no tearing happens in the rendering of the image.

The client is written in C using OpenGL, SDL, AntTweakBar and of course EPICS channel access libraries.

OpenGL is an API to do high performance 2D and 3D rendering on computers. In the client, it is used to render the image frames maintaining a high framerate while consuming low CPU cycles and RAM memory. SDL is a cross-platform library used to provide access to keyboard, mouse, and windowing system and graphics hardware via OpenGL. AntTweakBar is a library that works with SDL and OpenGL to provide a light graphical user interface to interactively tweak parameters and settings in an application. It is used to control and show the camera image settings like region of interest, framerate and trigger source.

Both driver and client are LGPL'd and can be found as public domain on Github [3]. (Fig. 2) shows the camera client presents colour mapping option.

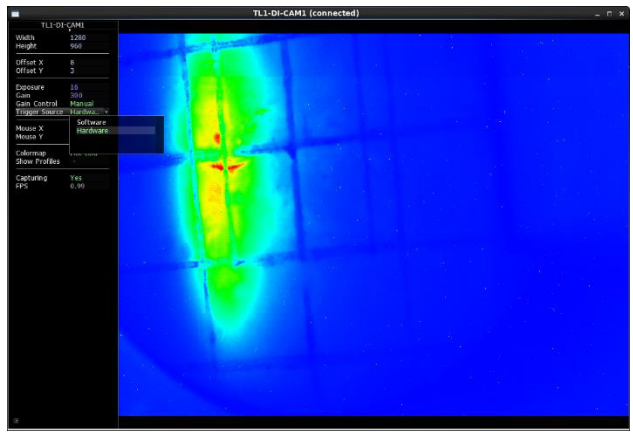

Figure 2: Camera Client.

IOC MANAGER

The control infrastructure at SESAME uses 12 virtual machines to run the different IOC's and 8 physical machines to run the clients and for development purposes.

The IOC manager is a Linux/x86 EPICS driver developed for the purpose of managing all the machines that are part of the control infrastructure at SESAME.

Specifically, the IOC manager provides the following services:

- Hostname and user of the IOC
- Date, time, and uptime of the IOC
- Drive, memory, and CPU utilization
- Control over which IOC's to run
- Enabling/disabling of the IOC's
- Build system

The date/time information give diagnostics on whether or not the IOC is synchronized with the NTP server. The drive, memory, and CPU utilization are indicators on the health of each IOC. They are connected to the alarm handler at SESAME. Control over which IOC's to run enables us to run different IOC's on different machines, and lets us know which IOC's are running where. Enabling/disabling the IOC's allow us to manually shutdown, turn on, or reset the IOC's. Finally, the build system of the manager synchronizes the IOC with the main Git repository, builds the source, and restarts the IOC's.

Each IOC is run inside a separate terminal multiplexer session. This allows us to a remotely attach/detach to a session whenever we need to access the IOC itself.

ISBN 978-3-95450-148-9

The driver layer uses traditional synchronous EPICS processing model of EPICS support modules. The device layer uses standard POSIX API to access machine information. Bash scripts are used to synchronize with the main Git repository. The client of the IOC manager is implemented in CSS and is shown in (Fig. 3).

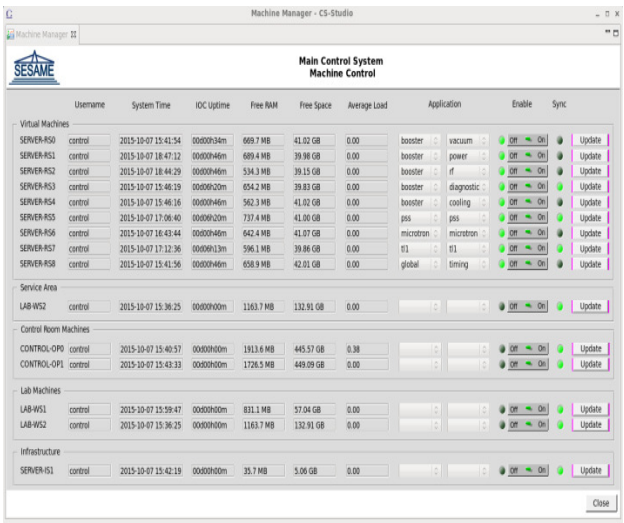

Figure 3: IOC Manager Client.

DB-CHECK TOOL

The db-check tool is a command line tool built at SESAME that automatically analyzes, validates and aids in applying continually evolving in-house rules for EPICS records databases. It was primarily built to help in reviewing, unifying and maintaining the numerous EPICS databases present.

Features provided by the db-check tool include: listing all records by name, or by type; querying a record to show all of its details and Process Variables (PVs); producing reports of records usage in each system, and what fields are set in those records; producing up-to-date alarm and archiver XML configuration files; and checking some in-house rules that PVs should follow.

Examples of the PV rules applied include: always having a DESC field present in all records: always having a ZNAM, ONAM, ZSV and OSV for BO records; always having the name of PVs to start with a one of a predefined set of prefixes (eg. is, get, set, reset, enable or disable)..

The way the db-check works is by parsing the DB and DBD files using a PEG grammar. The tool is built in D programming language and using pegged library for the parser [4]. The D programming language was chosen because it provides modelling power, efficiency and high performance in a coherent set of well-thought features, and because it interfaces easily and natively to the C programming language. An important feature that specifically helped in this project is compile-time function evaluation (CTFE) and mixins. This feature is used in pegged to build the parser using the grammar definition at compile-time instead of depending on a separate build step. And it reduced the compile-run-debug cycle time.

MBBI2 RECORD

Mbbi2 is a new record type developed at SESAME to overcome some limitations present in the official mbbiDirect record type in regards to alarms. An mbbi2 record is almost exactly the same as an mbbiDirect record except for its ability to specify alarm states for individual bits.

Frequently, devices provide read access to status registers where different bits mean different things. Some of those bits hold on/off status, others hold warnings and others hold interlock status. Using an mbbiDirect record to model this in EPICS creates a problem for alarm handling as value alarms are not supported. To support this without introducing a new record type will require using multiple CALC fields. The way mbbi2 resolves this is by setting an alarm severity on zero and one for each bit. The alarm severity for the record is equal to the highest alarm severity of its bits.

The main problem of introducing a new record type is the maintenance requirements of integrating it with the different drivers and their build systems.

TERMINAL SERVER

The terminal server is an experiment to provide simple serial to Ethernet converters using low cost computers. The computers have multiple serial ports that are mapped using the accompanying software to TCP ports available over an Ethernet network. The software was built to run primarily under minimal x86 or x64 Linux systems using libevent2 for asynchronous non-blocking communication. The software, once started, reads a configuration file and accordingly connects to a serial device on the specified baud rate and starts listening for clients on a TCP port. Once a client connects, data coming from the serial device is forwarded to the TCP connection and vice versa. Some simple statistics are printed periodically to the console for simple diagnostics.

Tests for the system showed that it was working well, but it has not been used in a production settings as of yet.

CONCLUSION

The control system of SESAME is based on EPICS and CSS. Development of new drivers, clients and tools at SESAME is important to make the control systems up to date and more consistent. Standards are used for both EPICS databases and CSS client OPIs.

REFERENCES

[1] A. Ismail, I. Saleh, Y. Dabain,, “CLIENTS DEVELOPMENT OF SESAME'S CONTROL SYSTEM BASED ON CSS”, Proceedings of PCaPAC2014, Karlsruhe, Germany, 2014.
[2] https://github.com/aismail2
[3] https://github.com/sesamecs
[4] https://github.com/PhilippeSigaud/Pegged

ISBN 978-3-95450-148-9