THE EPICS-BASED ACCELERATOR CONTROL SYSTEM OF THE S-DALINAC*

C. Burandt†, U. Bonnes, J. Enders, F. Hug, N. Pietralla, T. Schösser
Institut für Kernphysik, TU Darmstadt, 64289 Darmstadt, Germany
M. Konrad, Facility for Rare Isotope Beams (FRIB),
Michigan State University, East Lansing, MI 48824 USA

Abstract

The control system of the Superconducting Darmstadt Electron Linear Accelerator (S-DALINAC) is currently undergoing a migration to an EPICS-based system. Important subsystems are operated successfully by EPICS software since several years. This contribution describes latest developments as well as the current status of the migration.

INTRODUCTION

The S-DALINAC [1] provides beams with energies between about 3 MeV and 85 MeV. Superconducting, radio-frequency (rf) cavities allow the S-DALINAC to operate in continuous-wave (cw) mode. The design beam current is 20 µA for high energies, but it can also be adjusted down to the pA region. Figure 1 shows the layout of the accelerator. There are four experimental sites. Each one allows to perfom different kinds of experiments in the field of nuclear structure physics or nuclear astrophysics.

The accelerator control system was based on VMEbus computers since the last major upgrade in the late 1990s. These have been replaced during further developments by off-the-shelf PC hardware and widely used operating systems. However the control system software itself remained custom-made. Now we work on the migration of the control system to EPICS [2] that is a widely used and well-proven control-system framework. A large community of users potentially providing support via mailing lists and direct communication is quite beneficial. The substantial amount of freely available software related to the EPICS framework is also motivating this migration.

MIGRATION STATUS OVERVIEW

The migration became necessary to support an at that time new low-level rf control system [3]. The EPICS-based system is in operation since 2010 and proved to be stable. It is modified easily and fits to different hardware setups.

The low-level rf control system is part of a whole family of hardware components developed in-house at the S-DALINAC. This family comprises hardware of different types with diverse functionality. The communication interface is always kept identical. A microcontroller equipped with an integrated Controller Area Network (CAN bus) interface runs the same firmware on all variants. Therefore

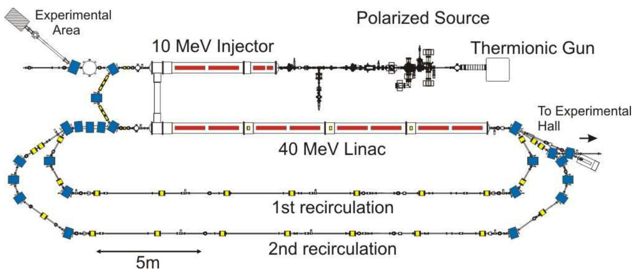

Figure 1: The S-DALINAC is a superconducting electron LINAC. Two electron sources allow experiments with either unpolarized or polarized electrons. The maximum acceleration energy is achieved through double recirculation of the beam. Thus the LINAC's accelerating fields are traversed three times.

the integration of further in-house developed device types could profit from the implementation of the rf control system Input Output Controller (IOC).

Today, the in-house developed QM07 multi-purpose measurement system is controlled by an EPICS IOC as well. It offers modules for precise current and voltage measurements as well as for beam-loss detection, read-out of resistive position sensors, etc. For maximum flexibility each hardware component of this modular measurement system is implemented in form of a separate template file. These template files are combined according to the hardware configuration using macro substitution files. They also allow to configure automatic conversion of the measured data.

The thermionic gun is controlled by a dedicated IOC. It runs on a small industrial PC which is installed inside the gun cage and therefore is located on high electric potential. A fiber connects the PC to a network switch. The gun is equipped with current and voltage power supplies.

The CAN bus attached magnet power supplies are the most numerous devices needed to operate the S-DALINAC. A proven EPICS implementation does already exist for these devices. Still the old non-EPICS control server is used for this subsystem. This is because operators prefer rotary knobs to adjust magnet settings, and there is no stable EPICS compatible solution at hand. However, a prototype with four rotary knobs combined with an industrial touchscreen PC running on Linux has recently been deployed to our control room. As soon as it proves to

* Supported by DFG through CRC 634.
† burandt@ikp.tu-darmstadt.de

be mature enough multiple copies will be installed. Subsequently the magnet power supplies will be migrated.

Commercial devices providing remote access through a serial interface are included into the EPICS-based system. The previous custom-made solution was unstable and hard to maintain.

The helium liquefier is the most critical component of the S-DALINAC by means of machine safety. Most other systems can be immediately shut down whenever problems occur. The cooling system can be heavily damaged when non-regular conditions arise. Therefore a dedicated process data acquisition had been set up long before the work on the EPICS migration started. This solution proved to be extremely stable. However, a parallel data acquisition by an EPICS IOC might be helpful to unify the presentation of information on the GUI level.

FRONT-END HARDWARE

The IOCs are consequently running Debian Squeeze and EPICS 3.14. Our hardware platforms are either off-the-shelf PCs or virtual machines on a VMware cluster.

Three types of device support modules are in use on the IOCs. While the StreamDevice device support [4] is a widely spread open-source package, the other two device support modules have been specifically developed for the in-house developed devices.

Interfacing Serial Devices

Serial devices providing access via RS-232, RS-422 or RS-485 are attached to so-called serial device servers. These are commercially available as stand-alone units. Those used at the S-DALINAC represent up to 16 individual devices on the network. Thus a single IOC can access all serially attached devices independently of their location on the site. Since it does not need to be physically located in a certain place, it runs on a virtual machine.

Socket-CAN Device Support

The in-house developed hardware family uses the CAN bus as a remote control interface. A custom CAN bus device support has been developed [5]. In contrast to already existing CAN bus support implementations our solution is based upon the Linux kernels Socket-CAN framework [6]. Applications relying on the Socket-CAN framework can operate with almost every CAN bus hardware interface. The only requirement is an appropriate Socket-CAN driver. These are often already part of the Linux kernel mainline or can be provided by the manufacturer. Figure 2 illustrates the Socket-CAN architecture.

A significant advantage of the Socket-CAN approach is testability. Basic command-line tools allow to observe the CAN bus communication and to send arbitrary CAN frames at will while running an application which uses the CAN bus itself:

- `cansend`: send a single CAN frame. Address and data content can be arbitrarily chosen.

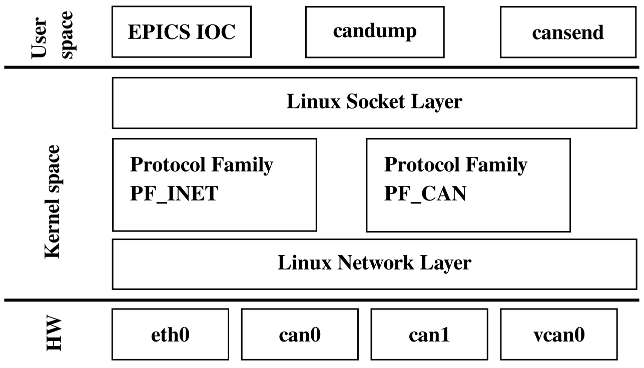

Figure 2: The Socket-CAN framework presents itself in the same fashion as the Linux kernel's network support. Different devices can be accessed by different applications at the same time. This is achieved by the network layer and appropriate protocol families.

- `candump`: dump the traffic to `stdout`. Precise timing information can also be obtained.

Thus efficient development of new template files using the Socket-CAN device support can be done with minimum effort.

The Socket-CAN device support has been written in the C programming language. Single bytes of interest can be extracted from an incoming CAN frame. Thus template configuration is simplified. The following record types are supported:

- analog, long and binary in/out
- multi binary (direct) in/out

Additional types could in principle be supported as well but have not been needed so far.

LLBBCUSB Device Support

The digital Low-Level Base-Band rf Control system (LLBBC) requires another type of device support. The control algorithm works with digital signals at a sampling frequency of 1 MS/s. Bit-level diagnostics at the full sampling rate is desired. Slow control can be carried out with the SocketCAN device support described above. With about 200 Mb/s the required data rate for the signal streaming is too high to be transfered via CAN bus.

Therefore two USB 2.0 interfaces provide the data. The first one provides bit-precise real-time data at full sampling frequency. The second one delivers a decimated data stream of all 1024 channels available. The LLBBCUSB device support is written in the C++ programming language. It utilizes the libftdi [7] open-souce library to retrieve data from the FTDI USB 2.0 chips used on our devices. The data stream is buffered and provided for direct transmission via tcp/ip. In addition the data stream with reduced data rate is fed into EPICS records each for one channel. These serve as indicators for the operators and allow long-term archiving of all channels of the rf control system at a reasonable rate.

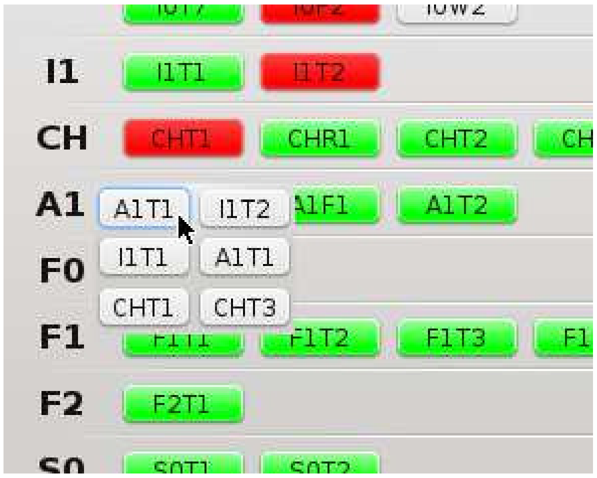

Figure 3: One of the GUI applications which is still to be migrated to an EPICS compatible implementation. The color of each button corresponds to the state of the target screen (red meaning beam is blocked). Left click on a colored button toggles the target's state. Right click will open the subwindow as can be seen on the left. It provides multiple buttons, each representing one of the video multiplexers channels connected to monitors in the control room.

GRAPHICAL USER INTERFACE

In the past graphical user interface (GUI) applications for the S-DALINAC had been developed in Qt (Linux) or by using the Windows API, respectively. Today the majority of GUI screens is being created through *Best Operator interface Yet* (BOY). It is used as part of the NSLS II distribution of *Control System Studio* (CSS) [8]. The possibility to change the operator interface screens (OPIs) without programming and compiling source code turned out to be quite helpful. Operators can improve the OPIs according to their own needs at will. A custom CSS product might be developed in the future to fit our specific needs.

On the other hand some older Qt applications are still needed for everyday use. Most important is the tool *Qtarget* which controls the optical beam diagnostics. It allows to change both the position of the scintillating target screens (inside or outside the beam's path) and the routing of the corresponding analog video signals. The latter is accomplished by a multiplexer which directs the video signal to one of the analog monitors in the control room. *Qtarget* configures itself from a relational database on every startup. The handling of targets and camera settings is solved most conveniently. A single button for each target/camera pair allows to manipulate both. While left clicking moves the targets, right clicking opens a small window which allows to choose the video channel to be used. This solution is superior in elegance in comparison to OPIs using standard BOY widgets (cf. Fig. 3). However it still does depend on the old non-EPICS control server which is still in charge of controlling the target controllers. Thus an adaption of *Qtarget* to EPICS is being considered.

SERVICE LAYER

Save and Restore

The Backup and Restore Tool (BURT) [9] is used as a save and restore application. It runs on a PC in the control room. Process variables are stored in local files. While these files can be synchronized between different PCs, it is still desirable to store different machine states in a relational database. Information fed into a database can be more easily requested and used by different applications on every PC in the network. A well-suited solution could not be found so far.

Archiving

Archiving process data is done into a relational database. At this time PostgreSQL 9.1 is being used. In the past a set of simple single purpose programs collected data from the helium liquefier and fed it into the database. Today CSS ArchiveEngine is used in addition to store approximately 6500 records into a separate table of the same database. Some process variables (PVs) are stored with a rate of 1 Hz up to 10 Hz. Others rely on an archive deadband configured in the `ADEL` field. However for some values this is an awkward solution, since it only defines an absolute difference but not a relative change. Especially vacuum values given naturally in logarithmic units, cannot be effectively reduced in their rate without the risk to lose details. The average number of samples is 1200 per second.

Storing data into a relational database at the mentioned rate is not too challenging for decent hardware. Data retrieval times on the other hand are not satisfying in all situations but still good enough. Detailed performance considerations for the installed database server can be found in [10].

NETWORK STRUCTURE

The network structure is kept simple but convenient. A private network is reserved for all devices and computers necessary for beam operation. This comprises all the IOCs, the operator PCs, the data acquisition modules and the serial device servers. More general services which need to be accessed from different subnets are placed in a so-called demilitarized zone (dmz). Especially the relational database must be accessible from the office computers.

A different subnet is reserved for the experiments. It is completely separated from the accelerator subnet. There are also IOCs running in this subnet. Read-only channel access is desired between these subnets. On the one hand part of the large number of PVs present in one subnet often is of momentary interest for someone using the other subnet. On the other hand general services like archiving should also store PVs from subnets different from the accelerator subnet. Therefore a dedicated virtual machine is running the EPICS PV gateway [11]. It has a network interface for every subnet of interest. Figure 4 depicts the setup in use.

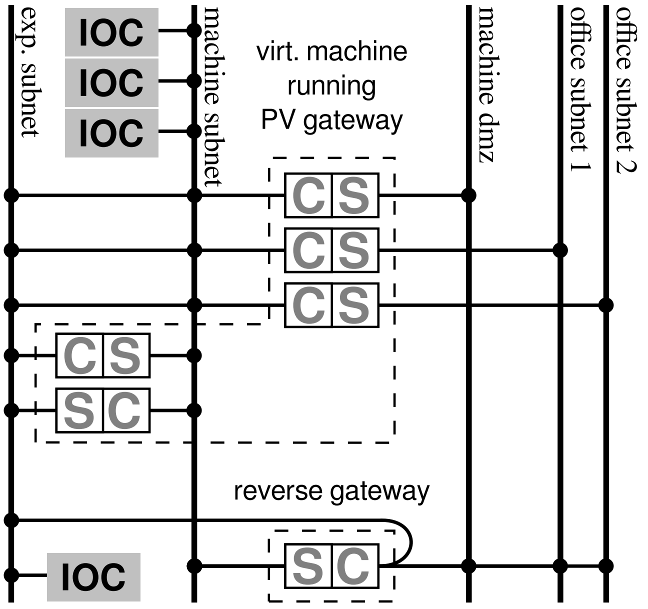

Figure 4: The IOCs required for beam operation are located in the machine subnet. Read-only access is granted to users in different subnets through an EPICS PV Gateway. C and S depict the client respectively the server side of a single PV Gateway instance. Five instances running on a single machine provide the necessary connections. An additional instance on a separate machine acts as a reverse gateway. It gives access to the gateways status PVs.

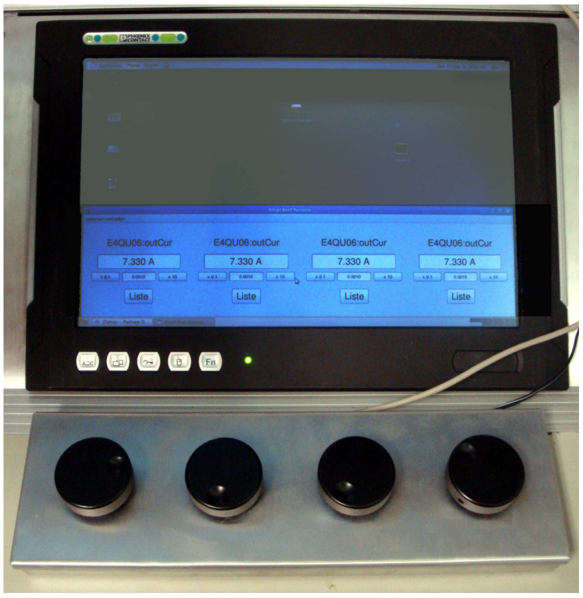

Figure 5: A prototype of the rotary knob user interface. An industrial touch-screen PC runs a Qt-based application. Four rotary knobs are connected to an in-house developed interface board which is in turn attached to the PC by USB.

FUTURE WORK

Further developments are now focussed on the final steps which will complete the migration to EPICS. Since setting up the beam is a quite complex procedure at the S-DALINAC due to its recirculating layout, the release of a stable user interface based upon rotary knobs is most important. A prototype is shown in Fig. 5. It runs on debian Linux. The application is written in the C++ programming language and the Qt-based QE framework [12]. A different project is the digitalization of the analog video signals. This will allow to remove the analog signal cables and to analyze beam spot pictures using digital algorithms.

Concerning the service layer, an alarm handler is still missing in our setup. Attachment of the beam dynamics simulation code VCODE [13] as a real-time model for the operators is desired to optimize the complex beam setup procedure at the S-DALINAC.

CONCLUSION

EPICS is now the basis of the S-DALINAC accelerator control system. New devices are consequently integrated into the existing EPICS-based infrastructure. Due to limited resources, there are still some subsystems left to be migrated.

REFERENCES

[1] A. Richter, in: Proceedings of European Particle Conference 1996, Barcelona, Spain (Institute of Physics, London, 1996), pp. 110-114

[2] EPICS, http://www.aps.anl.gov/epics/

[3] M. Konrad et al., "Digital base-band rf control system for the superconducting Darmstadt electron linear accelerator," Phys. Rev. ST Accel. Beams 15, 052802 (2012)

[4] EPICS StreamDevice, http://epics.web.psi.ch/software/streamdevice

[5] C. Burandt et al., "SocketCAN device support for EPICS IOCs", THPD13, Proc. of PCaPAC'12, Kolkata, India.

[6] The Socket-CAN project at berliOS.de, http://developer.berlios.de/projects/socketcan

[7] FTDI USB driver, http://www.intra2net.com/en/developer/libftdi/

[8] Control System Studio, http://controlsystemstudio.github.io/

[9] BURT: Back Up and Restore Tool, http://www.aps.anl.gov/epics/extensions/burt/

[10] M. Konrad et al., "Control System Studio Archiver with PostgreSQL backend: optimizing performance and reliability for a production environment," WEPD03, Proc. of PCaPAC'12, Kolkata, India.

[11] EPICS: The Process Variable Gateway, http://www.aps.anl.gov/epics/extensions/gateway

[12] The QE framework, http://sourceforge.net/projects/epicsqt/

[13] S. Franke, "Moment-Based Simulation of the S-DALINAC Recirculations," WEPC092, Proc. of IPAC2011, San Sebastián, Spain.