HIGH-LEVEL ALICE SOFTWARE DEVELOPMENT

J. K. Jones, B. J. A. Shepherd, STFC/DL/ASTeC, Daresbury, Warrington, Cheshire, UK

Abstract

The ALICE accelerator is a 35MeV energy recovery linac prototype at Daresbury in the U.K. Due to the highly experimental nature of the accelerator, there has been a strong influence of accelerator physicists in the high-level control software for the machine. Starting from the underlying EPICS-based control system, a suite of interactive commissioning software has been built using traditional software approaches, such as LabVIEW, as well as experimenting with interactive, rapid prototyping programming languages, such as Mathematica. Using the EPICS Channel Access protocols, the control system is flexible and extensible. A wide range of tools can be used to develop and debug high-level software, allowing machine physicists to use the most appropriate and familiar tools for software development.

INTRODUCTION

Modern accelerators require rich and complicated control system software to work effectively. In recent years with the introduction of channel access based control systems, such as EPICS, it has become commonplace to create and use control system written outside of the traditional programming languages such as C or Fortran. The use of high-level software has allowed physicists to increasingly take more of a role in writing the physics software they need to understand and manage these accelerators. The ALICE accelerator is no exception. With limited manpower it is increasingly important that physicists write their own software, leaving control systems engineers to concentrate on the underlying systems.

The increasing ease of programming using software packages such as MATLAB, Mathematica and LabVIEW, has made this task simpler. All of these codes now also allow rich GUIs to be produced in less time than ever before.

This paper will attempt to give a broad overview of some of the packages and applications that have been written for use on ALICE, and attempt to highlight the benefits of a rich-ecosystem of software that is made possible with these high level software packages, as compared to the more traditional approaches previously used. In all cases we attempt to highlight the appropriateness of the software choices that have been made.

EPICS INTERFACES

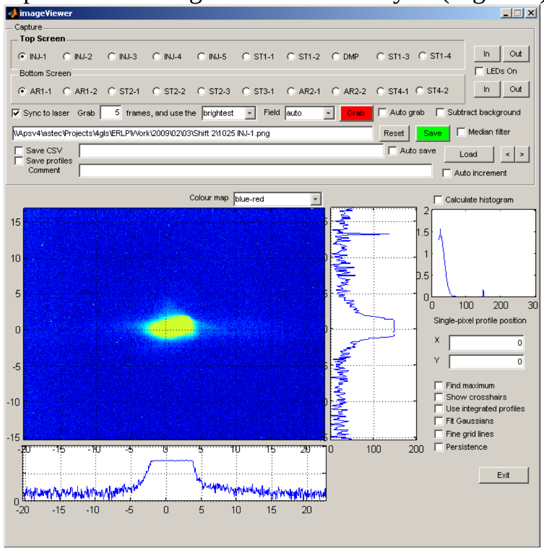
The ALICE control system runs on EPICS and VxWorks [1]. EPICS parameters are changed directly using Linux consoles in the control room. There is also a need for high-level software providing some abstraction from the control system, in order to perform tasks such as degaussing or emittance measurement, as well as visualisation and data acquisition.

Initially, the ActiveX implementation of EPICS Channel Access developed at ORNL [2] was used to develop interfaces to EPICS. However, this software is no longer supported, and the decision was made to develop an EPICS interface in-house that could be used with a wide variety of software platforms. This allowed us to develop high-level software using several different tools, according to the requirements of the task. A .NET Channel Access interface was developed, allowing control system parameters to be accessed and modified from any Windows software platform [3].

Read access to the channel access server is limited to computers connected to the laboratory network, which effectively allows read access anywhere on the DL site. This is a boon for software development, as it allows code generation and testing away from the machine control room. For security reasons write access is limited to control room machines.

IMAGEVIEWER

Twenty YAG and OTR screens are located at various points around the machine. Each screen has a CCD camera pointing at it; the signals from these are brought into the control room and digitised using a PCI frame grabber card. A MATLAB program, **imageViewer**, has been developed to interface to the frame grabber and capture beam images for further analysis (Figure 1).



[✓] Sync to laser Grab 5 frames, and use the brightest Field auto

\\Apsv4\astec\Projects\4gls\ERLP\Work\2009\02\03\Shift 2\1025 INJ-1.png

Colour map blue-red

Figure 1: Screenshot of imageViewer.

The program also interfaces to the control system to move each screen in and out, and to the multiplexer in the control room to select which screen is viewed. It can measure the beam position and size (horizontal and vertical) using a Gaussian-plus-background least-squares fit. All this information can be saved for later analysis, and series of images can be automatically recorded along with measured parameters from EPICS (for emittance measurements, for instance).

MATLAB was chosen in this case because of the ease of handling large datasets. Complex GUIs are straightforward to build and maintain using GUIDE, and the code was optimised for speed using the Profiler tool. imageViewer is able to capture and process images at a rate of about 3 Hz (the ALICE maximum repetition rate is 20 Hz). Better performance could be achieved by using compiled code.

ONLINE MODEL

As part of the commissioning process it is important to be able to analyse the machine in terms of standard lattice parameters. To this end a standard model of the design machine is implemented within Mathematica. The online model is based on the original MAD model of the machine, with dynamic corrections based on the latest engineering and survey positions. This is especially important for diagnostic positioning, where differences between the model and the final engineering layouts can be large. Analysis is performed either through the linear lattice code MLC, coded in Mathematica, or through the external MAD-8 code. Interaction with the external MAD code is done through the MADInput package, which automates the creation and running of MAD script files and interprets the output into Mathematica formats. The model is designed with ease-of-use in mind and utilizes the advanced GUI building capabilities of Mathematica. Interaction with the control system is controlled by simple "Apply" and "Read" buttons, maintaining the independence of the model from the machine until required. The model can also display 'expected' beam positions and sizes at BPM and OTR screens.

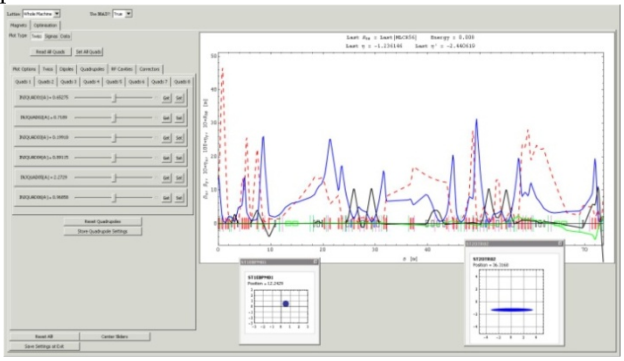


Figure 2: Online model main screen, showing the quadrupole controls as well as BPM and OTR displays.

The use of external codes for analysis is important for the ALICE machine due to its complexity. Currently MAD-8 is implemented to enable continuity with the original design model. However, issues surrounding the treatment of RF focusing in the linac cavities has recently become a concern. The MLC code is fully customisable and allows easy modification of these terms, which should allow a more complete model of the machine to be built at a later date. It is also planned to implement an interface to the ASTRA code to allow further online modelling of the low energy section of the machine from the gun to the exit of the booster cavity.

Calibration of the online model is essential to maintaining parity with the machine. The online model features a built in set of optimisation routines that allow rapid fitting of many parameters on the machine. Optimised solutions can be saved and re-loaded. This should enable simple changes to the operating point of the machine, and allow rapid re-optimisation of the machine state as conditions change.

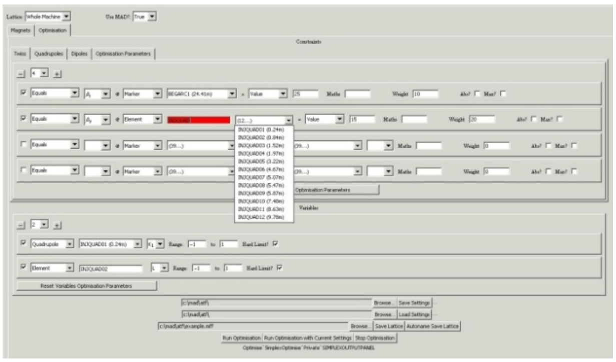


Figure 3: Optimisation screen.

OTHER SOFTWARE

Compression Chicane Tracking

Recent ALICE commissioning activity has focused on setting up the THz beamline [4]. Steering the beam through the compression chicane is effectively done almost 'blind' due to the low intensity on the screen in the centre of the chicane. Using measured field maps, a model was put together in Mathematica to track a beam through the chicane (Figure 4). A drawing of the vacuum vessel is overlaid on the top to ensure the beam passes safely. The notebook is linked to EPICS to update magnet currents and BPM readouts. This simple tool has been particularly effective in optimising the THz output from the chicane.

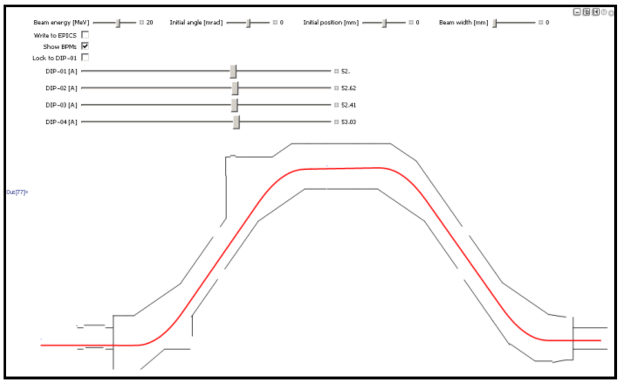


Figure 4: Screenshot of the Compression Chicane Tracking notebook.

Machine Status Web Server

During ALICE shifts, the work is logged using the ELOG electronic logging software [5,6]. To facilitate a simple method of recording the machine parameters, a small web server was developed using AutoIt [7]. This server runs on one of the control room consoles, and constantly monitors the status of various control system parameters. By clicking a button in the ELOG editing screen, the current status of a section of the machine (magnet settings etc.) is requested from this server using AJAX, and a table of parameters is pasted into the log. In addition, a text file containing these parameters is created on a local file server; this can later be used to restore settings using BURT [8].

BPM Viewer & visualSteer

Using LabVIEW, the **BPM Viewer** was developed to provide a graphical display of the horizontal and vertical BPM readouts. **visualSteer** can be used in tandem with this to aid with steering a beam around the machine using the H&V corrector magnets. Clicking and dragging on the display in the visualSteer window adjusts the H&V corrector settings simultaneously. Figure 5 shows screenshots of both programs.

LabVIEW was chosen in this case due to the relative simplicity of the task, and the ease of developing GUIs in this environment.

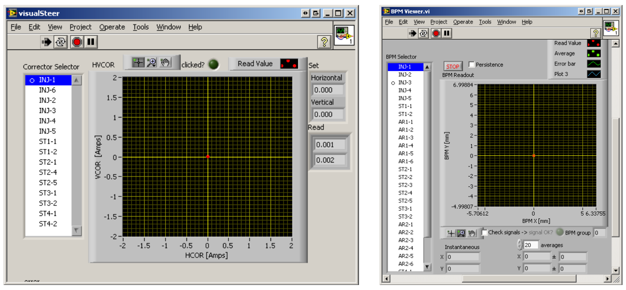


Figure 5: Screenshots of visualSteer (left) and BPM Viewer (right).

Twiss Parameter Determination

As part of the online-model calibration, a simple image analysis and characterisation program has been developed in Mathematica. The seamless integration with both the MAD and MLC codes makes it trivially easy to compute the R-matrix values at different quadrupole strengths and compare these to a series of related YAG images, and thus determine the Twiss parameters. Mathematica's powerful image analysis routines can automate the processing of images, and its interaction with the control system, via EPICS, should allow a fully automated Twiss parameter determination at any YAG screen in the machine. Currently, however, the ActiveX based image grabber controls cannot be accessed directly from Mathematica, though this is a task actively being worked upon.

COLLABORATIVE DOCUMENTATION

Documentation for the ALICE machine has been implemented using the MediaWiki web server [9,10]. This 'wiki' software allows anyone on the project (suitably authorised) to edit the ALICE documentation, using only a web browser. For basic editing, no specialist knowledge of HTML markup is required, and articles can be linked together and categorised easily. Changes can be easily tracked – each page has a history of changes and mistakes can easily be corrected. Using this tool, ALICE documentation is accessible to anyone with a web browser, and can be kept accurate and up-to-date. MediaWiki is in wide use across the internet and at least one other accelerator laboratory [11].

CONCLUSIONS

The use of an EPICS control system on the ALICE machine, and the corresponding ease of data extraction within a multitude of high-level coding platforms, has allowed the rapid development of useable machine software. This software is primarily created without the interaction of control-systems engineers, instead by physicists using the machine, and in the software and using the methods they are most familiar with. This has effectively lowered the barrier of entry for scientists who ordinarily would not write such software, and can generate a suite of packages that are both useful and time-efficient to produce.

The use of many different software programming languages allows the restrictions or disadvantages of any one programming language to be mitigated, and helps to ensure that software is written in the most appropriate manner. All of the software demonstrated in this paper takes advantage of the differing programming paradigms inherent in these different software packages, to create software that is relevant and efficient to the task at hand. With the introduction of advanced GUI building capabilities in all of these packages, the inherent difficulties in using a wide number of packages has effectively been negated, and control software programming is no longer strictly tied to the 'lowest common denominator' programming language.

REFERENCES

[1] A. Oates *et al*, "Development of the Control System for ERLP", ICALEPCS, Geneva, Oct 2005, PO1.044-6.
[2] http://ics-web.sns.ornl.gov/kasemir/axca/index.html.
[3] G. Cox, "A .NET Interface For Channel Access", PCaPAC08, Ljubljana, Oct 2008, TUP022, p134-136.
[4] S.L. Smith, "Progress on the Commissioning of ALICE, the ERL Based Light Source at Daresbury Laboratory", these proceedings, TU5RFP083.
[5] S. Ritt, ELOG software: http://midas.psi.ch/elog.
[6] ALICE eLog: http://www.4gls.ac.uk/erlp/elog/.
[7] AutoIt script language: http://www.autoitscript.com.
[8] EPICS Back Up and Restore Tool (BURT): http://www.aps.anl.gov/epics/extensions/burt/index.php.
[9] MediaWiki software: http://www.mediawiki.org/.
[10] ALICE wiki: http://projects.astec.ac.uk/ERLPManual/.
[11] Niedziela *et al*, "Dynamic Collaborative Documentation At The BNL Collider-Accelerator Department", PAC07, Albuquerque, 2007, MOPAS098.