HIGH-LEVEL CONTROLS UPGRADE AT THE ALS*

G. Portmann, H. Nishimura, C. Timossi, C. Ikami, M. Urashka, M. Beaudrow, H. Mahic
LBNL, Berkeley, CA 94720, U.S.A.

Abstract

The Advance Light Source (ALS) is in the process of upgrading the high-level controls software. This welcome upgrade is driven by the need for a low-level controls hardware upgrade. The risk of a failure in some of the aging controls hardware is reaching a critical level. The dilemma is that replacing the low-level hardware will break some important control room applications. An effort has been started to replace all the high-level software in a way that is compatible with an incremental low-level hardware replacement. As will be presented in this paper, the plan involves combining three very different programming methods: C#, Matlab, and EPICS tools.

INTRODUCTION

The ALS started operations in 1993. A number of high-level software and hardware upgrades and improvements have been made since then, primarily to the storage ring and booster to be compatible with EPICS, but a full upgrade is long overdue. In fact, from the electron gun to the booster ring there have been almost zero changes to the system since it was built. The motivation for the upgrade is not only to take advantage of new technology to improve operator control of the accelerator, but to pave the way for new low-level controls hardware. The ALS is in a precarious situation where much of the original controls hardware is not easily repairable or replaceable. Since the original high-level controls applications are not based on a network API, like channel access, but rather on synchronous calls across a direct physical link, replacing the high-level software or low-level controls hardware is very difficult – the applications are too closely tied to the underlying hardware. As presented here, a plan is in place to change all the high-level software as well as some low-level hardware to facilitate the use of channel access.

Attached to the high-level software upgrade is a plan for all new workstations, new OS, and a new ergonomic design. Logistically, replacing the high-level controls is quite difficult. Not only does it involve the writing and testing of new applications, there is a large operator training and socialization component to the project.

GENERAL PLAN

The ALS has used a combination of Windows, Linux, and Solaris for some time. This diversity creates some difficulty with computer support and some practical problems for the operations staff who have to know something about all three operating systems. The ALS has enjoyed a rich history of developers experimenting with many different computer languages. Although creative and historically interesting, it has reached a point where consolidation is necessary. Recognizing the proper balance between fostering creative solutions and restricting application design and language selection is critical for a healthy control room.

The software plan for the ALS is to restrict high-level applications to two languages, C# and Matlab, and one EPICS display manager, EDM. These methods were chosen to provide a relatively wide variety of software tools to developers and it's well aligned with the expertise of the present ALS staff. The learning curve for each of these tools is drastically different – typically months, weeks, hours for C#, Matlab, and EDM respectively. The type of problem often addressed by each tool is very different.

Another of the design goals is to decrease the number of operating systems in the operator console area. Since the ALS has significant experience with the Windows OS for operator consoles all consoles will use the latest version of Windows Vista. The .NET based applications will be developed on Vista. Portability to Linux/Mono has been shown to work, but will likely not be used. All EPICS tools are run on Linux and displayed to the console using X-Manager.

The new PC hardware will be a substantial upgrade from the present Windows 2000 machines. The starting configuration will be seven operator consoles (64-bit quad-core CPUs) with 30" LCD monitors, 2 Windows 2008 Servers, 2 development consoles. Much more attention to ergonomics will be given this time around. Fig. 1 shows a new fully adjustable (table, monitor, and chair height) duel console table. The old system is mounted in a fixed rack that makes upgrades difficult. For instance, the monitor size was limited to 19". Instead of a home-made rack mounted knob panel, an inexpensive Griffin knob will be used.

Figure 1: Console unit.

*Work supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231

NEW SUPPORT LIBRARIES

A number of support libraries were needed for this effort. Some libraries were developed in order to maintain a connection to the old ALS control system. Although challenging to do, these libraries are likely only of interest locally hence will not be discussed here. For the Matlab-to-CA link, the LabCA, [5], is being used. The main library developed at the ALS which may be of interest to the general community is SCA.NET.

SCA.NET

SCA.NET is a .NET class-library assembly meant for use by .NET applications needing CA client functionality, [1]. An assembly has the same file extensions as a windows executable, either exe or dll, but has a different file format that includes meta-data allowing it to be self-describing.

The goals for SCA.NET are somewhat conflicting:

- Provide a thin layer around the CA library which has proven over many years to be a robust and high performance network layer for accelerator data transport.
- Hide some of the details of the CA interface in a class that is more intuitive to a .NET application builder than the CA API.

The design utilizes two classes. Als.Epics.ChannelAccess is a static class that contains all the direct .NET to Ca.dll mappings using the Platform Invoke (P/Invoke) interface provided by .NET System.Runtime.InteropServices. This class is used by Als.Epics.SimpleChannelAccces exposes a class more suitable for .NET applications.

HIGH-LEVEL SOFTWARE

The effort to replace all the high-level applications started in 2008. As mentioned, this effort combines three very different programming methods: C#, Matlab, and EPICS tools. However, the backbone of the high-level controls effort is C# on the .NET framework on Windows Vista.

The plan is to start at the electron gun and prototype the software tools needed for the entire machine. The first phase of a high-level controls upgrade is nearing completion. It is now possible to tune the accelerator from the electron gun, through the LINAC, to the booster injection point.

.NET & C# Framework

Details on the using the .NET Framework for high-level application development can be found in [1, 2, 4, 6]. Fig. 2 shows screen shot of a number of the C# applications. Considerable effort has been put into a transport line tuning GUI supporting knob controls (both a software and hardware knob).

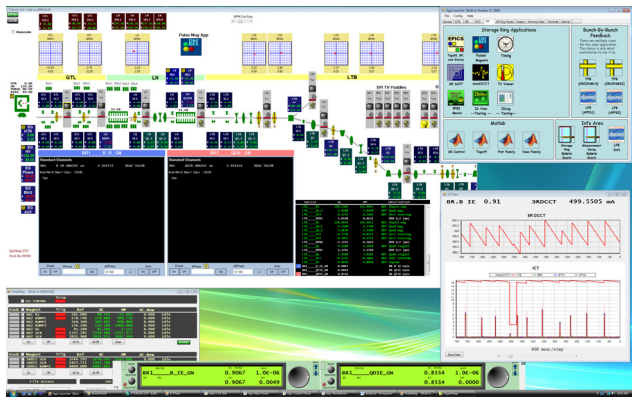

Figure 2: Some C# applications including the gun-to-booster tuning application.

The plan is to create a tool similar in concept to the EPICS display managers using C#, .NET, and Expression Blend 2. By doing so, the application GUI designer will not need C# knowledge to create applications. However, if complicated logic is required in the GUI, then the power of the C# language can easily be incorporated, [4].

MATLAB

Matlab has been heavily used at the ALS almost from the beginning. It is an easy language to learn with a large number of numerical receipts readily available. Matlab is commonly found in accelerator control rooms. The ALS uses Matlab for many things including topoff control, save/restore, magnet conditioning, energy ramping, orbit control, tune and coupling compensation of insertion devices, as well as a scripting language for automating physics experiments. Fig. 3 shows the main Matlab GUI for controlling the ALS storage ring and Fig. 4 shows a general orbit correction GUI that can be used at a number of accelerators.

The MML software package (Matlab Middle Layer), [3], is extensively used. Although starting as a "middlelayer," this software has expanded into a much larger analysis and control package for accelerators. Since the middle layer title is no longer appropriate the MML acronym has replaced the name. The MML toolbox, AT simulation toolbox, and LOCO application [10] (all written in Matlab) have become a standard tool for optimizing light sources. The plan is to expand the transport line and linac tuning capabilities of the MML toolbox with this high-level controls upgrade.

EPICS Display Manager

The EDM EPICS display manager provided by the EPICS community, [9], is used for a number of applications. The simplicity of use is quite appealing.

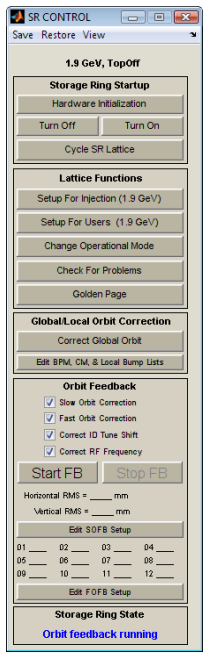

Figure 3: Main Matlab GUI for Storage Ring Control

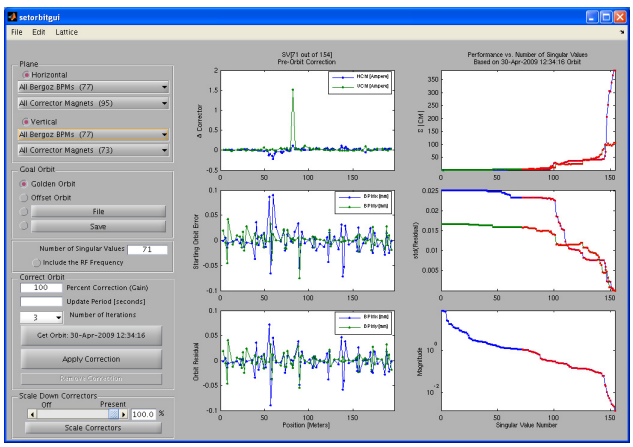

Figure 4: Orbit Control GUI in Matlab

SOFTWARE TOOLS

Graphing

There are many graphing tools available as freeware or for purchase, however, it was difficult to find the one that met the ALS operations needs. A C# application is under development to both mine the EPICS Channel Archiver and continually update as new data arrives. This has the advantages of not having to make CA connections and the graphs can be fully populated on start.

Version Control

Although CVS is still the main version control system at use in the ALS, for the upgrade project, we are using Subversion. The primary SVN client tool for Windows is TortoiseSVN, [8], which is implemented as a file explorer extension.

AKNOWLEDGEMENTS

The authors thank A. Biocca and D. Robin for their support. We appreciate the patient cooperation of machine operators group for using new programs and giving us constructive feedback.

REFERENCES

[1] C. Timossi, H. Nishimura, “A .NET Assembly for EPICS Simple Channel Access,” PCaPAC 2008, Ljubljana.
[2] H. Nishimura et. al., “Re-writing the ALS Control Room Software in C#”, PCaPAC 2008, Ljubljana.
[3] G. Portmann, J. Corbett, A. Terebilo, “An Accelerator Control Middle Layer Using Matlab,” 2005 PAC, TN.
[4] H. Nishimura et. al., “ALS Control System Upgrade in C#,” PAC 2009, Vancouver.
[5] T. Straumann, http://www.slac.stanford.edu/grp/cd/soft/epics/extensions/labca/manual/
[6] H. Nishimura and C. A. Timossi, “Control Room Application Development Using .NET”, PCAPAC 2005.
[7] H. Nishimura and C. Timossi, “Mono for Cross-Platform Control System Environment”, PCAPAC 2006.
[8] http://tortoisesvn.tigris.org/
[9] J. Sinclair, http://ics-web.sns.ornl.gov/kasemir/train_2006/ 1_4_EdmTraining.pdf.
[10] J. Safranek, G. Portmann, A. Terebilo, C. Steier, “Matlab-Based LOCO,” EPAC, Paris, June 2002.