HIGH-LEVEL CONTROL SYSTEM IN C#*

H. Nishimura, C. Timossi, G. Portmann, M. Urashka, C. Ikami and M. Beaudrow
Lawrence Berkley National Laboratory, Berkeley, C A 94720, U.S.A.

Abstract

We have started upgrading the control room programs for the injector at the Advanced Light Source (ALS)[1]. We chose to program in C# exclusively on the .NET Framework[2] to create EPICS[3] client programs on Windows Vista PCs. This paper reports the status of this upgrade project.

INJECTOR CONTROLS UPGRADE

Current Status

The ALS control system has been gradually migrating from the original control system[4] to EPICS for over a decade. All the new devices are controlled by EPICS. However, this effort was focused on the storage ring. The original system still plays a primary role in the injector section since it's commissioning in 1991. The key component there is called Display Micro Module (DMM) through which all the original programs access the devices. As the DMM uses hardware that is no longer supported, it must be replaced in the near future with IOCs The old applications that control the injector are also not worth adapting to channel access. With the new demands placed on the injection system by top-off, we believe that the control room applications for the injector should simply be re-written.

Upgrade Plan

We have been very impressed with Microsoft Visual Studio[6] and it's ability to build visual applications quickly and on the .NET Framework 3.5 that is the platform for current development on Windows. We are finally moving off the Windows 2000 OS, which is no longer supported, and onto new hardware running Vista. We plan to use SCA.NET[5], a C#EPICS CA wrapper class in , that we've been using elsewhere for over 5 years.

We decided to keep using WinForms, the GUI framework of .NET 2.0, to reuse the libraries we have developed for EPICS client programs for the storage ring. At the programming language level, we use the new features of C# 3.0 for much better efficiency.

For hardware we have 7 PCs with 64-bit quad-core CPUs: 2 Windows 2008 Servers, 2 development consoles and 3 operator consoles. The operator consoles are equipped with 30" LCD monitors and knob panels that we mention later. These consoles also display existing EPICS MEDM/EDM clients and Matlab programs on the X11 servers. Although we have been using C# for years to create EPICS clients, C# is still relatively new to the accelerator controls community. Therefore, as as a demonstration, we chose to start with a small but operator intensive section of the accelerator and create a program that integrates many older applications.

The region we chose is from the electron gun (EG), through the gun to the linac beam transport line (GTL), the linac (LN), to the end of the beam transport line (LTB) to the booster ring. The most complicated devices to control are the electron gun, the linac system and the timing system. Their control logic is being ported to C# together with some hardware modification were needed. Other devices, listed in Table 1, are relatively simple. We created a new program called CTLBook to replace multiple existing programs.

Table 1: Numbers of Simple Devices

Device	EG	GTL	LN	LTB	Total
Bend	0	0	0	4	4
Steering	0	8	4	8	20
Quad	0	0	2	9	11
Solenoid	0	3	4	0	7
BPM	0	2	1	6	9
Scintillator	0	2	2	6	10

SOFTWARE DESIGN

We have two major design goals. We want all the applications to be data driven, to allow changes to be made with out necessarily requiring code changes, and to be component based.

Data-driven Architecture

The new programs are made highly data-driven. EPICS channels(pv) have been managed by the MySQL database and accessed by the EPICS client programs. There are over 16K channels for the entire ALS, and about 600 are in the target region. We use ADO.NET to access various SQL databases, and load the data to the ADO.NET DataTable objects at runtime. We usually save the data in these DataTable objects to XML files, and repopulate them from XML. This reduces the effort of managing the database systems on the operator consoles. We created several tools to manage XML in a context of accelerator controls. Fig. 1 is one of such tools.

XML becomes even more important when the concept of the devices comes in. As the complexity of the devices varies over wide range, a flat data structure of the database tables cannot support it without relying on the relations among multiple tables. In contrast, XML is flexible enough to support complex devices. We have constructed over 2000 devices by taking the information from the original system that has some concept of devices by imposing a strict naming convention, and applying empirical rules repeatedly to over 16K of EPICS

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

channels. The resulting XML file containing the device definitions is the core of this project. Fig.2 shows one of the devices there. XML is also used for application-specific data and configuration files. This means that all the data files are now in XML.

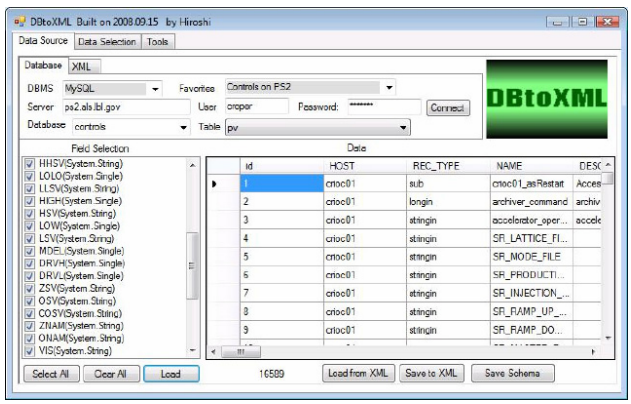

Figure 1: Database to XML Tool

```
<DMMDevice DEVID="23200" NAME="LTB_____Q3,1" BEAMORDER="3032
  <Comment Value="" />
  <STDCHANNELS>
    <DMMChannel NAME="LTB_____Q3,1___AM00" CHANTYPE="AM" DES
    <DMMChannel NAME="LTB_____Q3,1___AC00" CHANTYPE="AC" DES
    <DMMChannel NAME="LTB_____Q3,1___BM01" CHANTYPE="BM" DES
    <DMMChannel NAME="LTB_____Q3,1___BC20" CHANTYPE="BC" DES
    <DMMChannel NAME="LTB_____Q3,1___BM00" CHANTYPE="RDY" DE
  </STDCHANNELS>
  <EXTRACHANNELS />
  <KNOB Comment="EGUL set to 0">
    <AC Unit="A" EGUF="10" EGUL="0" Gain="0.01" Comment="" /
    <BC CheckAC="True" ACmax="0.01" AMmax="0.3" Special="Fal
    <Auto Step="1" Comment="EGUL set to 0" />
  </KNOB>
</DMMDevice>
```

Figure 2: Example of a Device in XML

Component-based Development

We have been developing the components and the application programs in parallel by using Visual Studio. We have created total 40 custom components; 10 for GUI and 30 for controls. About 1/3 of them are in the real use.

Here are some examples. Fig. 3 shows components used in CTLBook. There are three kind of device components for scintillators, magnet power supplies (MPS) and BPMs. They reference the DeviceDB control that manages the devices information by loading the device XML file, and the ScaControl that holds the SCA.NET object and manages channel accesses at runtime. Fig.4 shows a MPS control at design time that is assigned to the device GTL___VC4. This name is passed to the DeviceDB to retrieve its channels and knob parameters. When it starts running, it is registered to the ScaControl to have the access to its EPICS channels. A MPS controls has small button titled "S" for "setting" at the upper-left corner. This pops up a window (Fig.5) to monitor and control all of its channels. These device controls are used to create an integrated program CTLBook (Fig.5).

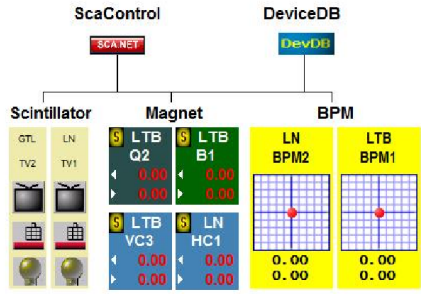

Figure 3: Components used by CTLBook

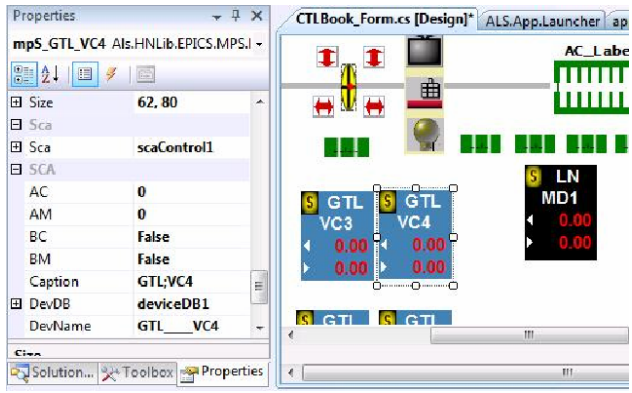

Figure 4: MPS Component

Figure 5: MPS Control Panel

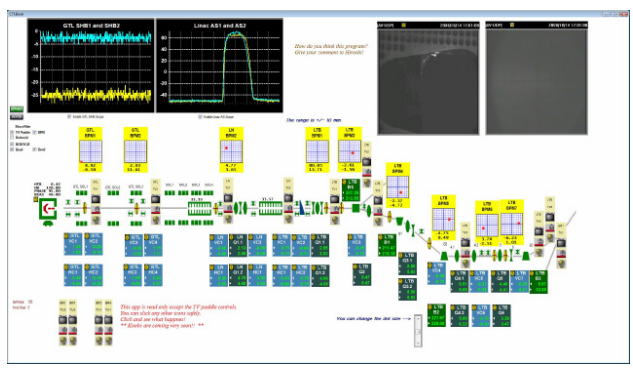

Figure 6: CTLBook

Here is another example. Accelerator operators always use the proprietary hardware rotary knobs (Fig.7) to tune up the injector. The new system must support them.

HARDWARE

Some of the functionality that we need for operating the injector requires upgraded hardware as well as software. The existing hardware was just too deeply tied to the existing control system to be able to use it.

Knob Panels

Figure 7: Hardware Rotary Knobs

We developed a software knob component (Fig.8) and linked it to the existing rotary knobs and also other USB knobs, such as PowerMate[7]. Once the device name is assigned, it retrieves its information thorough the DeviceDB component.

Figure 8: Software Knob Component

We used 3 of them to create a software knob panel (Fig. 9) that locates itself always at the bottom of the 30" display. CTLBook is one of its client programs that can talk to it. The knob assignment to a device is done by using the MPS control panel in case of CTLBook.

Figure 9: Software Knobs

Scope and TV Signal Displays

CTLBook has two scope displays in the upper-left region. They read the EPICS wave form data and simply display them by using a plot component. Among many options, we chose TeeChart[8] for plotting. CTLBook also displays two live TV images in the upper-left region. They are the beam spots on the scintillator plates. We use MOXA video encoders[9] to transfer images from TV to the network. We modified one of their sample programs in C# to create the component.

DISCUSSION

We have created and released a highly integrated program CTLBook for real online testing. The data-driven and component-based architecture works efficiently. Compared to our previous experience with Delphi on Win32[10], development in C# on .NET bring much better productivity. We will continue to develop in this manner.

One issue we have encountered is the very tight security of Windows Vista. We managed to keep the User Account Control (UAC)[11] enabled on the operator consoles. Although it requires extra steps in deployment, it helps to maintain the system secured and stable.

The performance of a PC console is very satisfactory. It can run CTLBook, several other C# programs, and dozens of X11 windows hosting EPICS DM and Matlab sessions with very low overhead.

AKNOWLEDGEMENTS

The authors thank A. Biocca and D. Robin for their support, E. Williams for firmware development of rotary knobs, S. Jacobson for supporting virtual device channels.

REFERENCES

[1] ALS CDR, LBL PUB-5172 Rev. LBL,1986

A. Jackson, IEEE 93PAC, 93CH3279-7,1432, 1993.

[2] http://www.microsoft.com/net/

[3] L.R. Dalesio, et al., ICALEPCS '93, Berlin, Germany, 1993.

http://www.aps.anl.gov/epics/

[4] S. Magyary, IEEE PAC93, 93CH3279-7,1811,1993.

S. Magyary et al, NIM A 293, 36-43, 1990

[5] H. Nishimura and C. Timossi, PCaPAC 2005

H. Nishimura and C. Timossi, PCaPAC 2006, p37

C. Timossi and H. Nishimura, PCaPAC 2006, p56

C. Timossi and H. Nishimura, PCaPAC 2008 (this proceedings)

[6] http://msdn.microsoft.com/en-us/vstudio/default.aspx

[7] http://www.griffintechnology.come

[8] http://www.teemach.com/

[9] http://www.moxa.com/

[10] C. Timossi and H. Nishimura, IEEE PAC'97, 0-7803 -4376-X/98, p805, 1998

[11] http://technet.microsoft.com/en-us/library/ bb629420.aspx