AN INTEGRATED ENVIRONMENT FOR CONTROL SYSTEM DEVELOPMENT

S. Hunt, PSI, Villigen, Switzerland

Abstract

As the cost of manpower is the major factor in the cost of a modern control system, reducing the time to develop and test control systems can have a major impact on overall cost. An integrated development environment has been developed at the Swiss Light Source for building and maintaining control systems using the EPICS toolkit. This package guides the developer through the stages of control system development, and as well as speeding up the process, helps the designer to more closely follow the development guidelines and so makes the project easier to maintain. By automating many tasks the package reduces errors, while still allowing customisation where necessary. The tool supports the following features: CVS interface including browse, checkout, addition, commit and tagging of modules to and from the repository; Automatic generation of documentation including lists of hardware needed. Automatic generation of configuration files, including files for the Archiver, Alarm handler, and Cdev. Checking of EPICS database syntax before installation; Automatic generation of basic user screens; Running the project in a simulation mode (without I/O hardware) while still in the development environment, allowing rapid development of user screens; Installation of files to the correct directories in the production area. With this tool, the first version of an average project, requiring analogue, digital, temperature, and Motor I/O, with the associated user screens, alarm panels, archiving, and trending can be running in under one hour. Rollback to a constant set of config files and user applications takes less than one minute.

INTRODUCTION

The largest single component of control system cost, over the lifetime of a project in manpower for software development and integration. It follows therefore that reducing the time needed for the development and maintenance of control system will have a very large impact on reducing overall cost. This paper presents the motivation behind, and implementation of an integrated environment for developing accelerator control systems. Although built to support the SLS, much of the functionality can be useful at other locations, and some effort has been made to support site specific customisation.

STAGES OF BUILDING ACCELERATOR CONTROL SYSTEMS

The activities involved in the building of accelerator control systems can be classified into a number of overlapping stages:

Providing the controls infrastructure

For hardware components, in the not too distant past, this stage typically consisted of establishing the requirements, in terms such as resolution, accuracy, update rate needed for I/O signals of each device to be controlled and monitored, then designing and building, using in-house expertise, the necessary hardware. Now it is more normal to purchase the necessary components from industry.

The software infrastructure, consists of providing the computers, operating system, compilers, as well as the libraries or packages used for visualisation, communication, hardware access, error handling, trending, archiving etc. While in the past this effort took up a large part of the control group effort, this stage has been simplified by the availability of existing stable, high performance components, both from within the accelerator community (EPICS) and from industry (Scada packages). It is no longer necessary, or desirable, to write and maintain the software components that make of the infrastructure elements of your control system.

Implementation

The implementation of the control system consists of using the infrastructure already available to provide the necessary control and monitoring of accelerator devices. In the past this would often have involved writing individual applications in a high level language, even building individual user screens using libraries of graphics routines such as xlib, or motif. These applications might have run on general purpose computers, on consoles or servers, or on real time front end computers. In some accelerator projects, the implementation stage was largely the responsibility of individual equipment groups, not the controls group. Now it is often the case, both with EPICS and Scada systems, to configure systems, rather than write application in the traditional sense. Standard tools are available for functions such as alarm handling, archiving, and trending. Tools are available to build user screens in an interactive fashion Control algorithms can be implemented by providing parameters to existing prebuillt routines (such as PID).

INTEGRATORS ARE WE

So in the modern paradigm, the control groups task has become that of system integrators, selecting from a pallet of existing tools, then using those tools to control and monitor accelerator devices. This has potentially reduced the effort (cost) of providing the control by a large factor. So what is missing?

Developing using EPICS

Developing using EPICS largely consists of producing configuration files for each software component and system. The configuration files are ascii text files, to make them easy to produce and maintain using standard tools. Configuration files are produced for:

- EPICS core – the real-time EPICS database which provides the input output routines to read and write I/O hardware, and implement control algorithms.
- Alarm handler – A tool to monitor the state of EPICS channels and provide that information to the operator in a hierarchical display.
- Archiver – a tool to record the state of channels, either periodically, or on change.
- Striptool – a trending tool.
- Medm – An operator interface builder

For the novice, building and managing these tools can be quite daunting, particularly learning the syntax of each. Also a change in one, perhaps the addition of a channel in the EPICS core configuration, requires changes in all others.

Furthermore each site establishes (perhaps without realising it) procedures for change and configuration management. As the EPICS configuration files are text files, they are well suited to using CVS to track and manage changes in system configuration

So what is missing?

What was missing in the EPICS world, was a tool for managing the whole development process, making it easy for novice users to quickly control accelerator devices, in a way which is compatible to the overall controls structure.

WHAT IS SIDE?

Side[1] is a program that integrates the existing EPICS tools using a graphical interface. It provides a central point in the control system development process. By using a tool it is easier for a user to follow site specific standards such as file locations, and file names. A project goal was to make it possible to provide control and monitoring of a new device of medium complexity, fully integrated into the control system, within one hour.

CVS functions

At SLS all configuration files used in the accelerator are stored in CVS. This allows us to track changes, and if necessary rollback to an old version. Functions provided are similar to those in tkcvs, with the addition of the ability to browse the cvs repository.

File creation function

Side manages the creation of the configuration files necessary for the project. Once the basic configuration files for EPICS core have been created (template and substitution files) the other files can be auto generated in a consistent manner. These can later be customised.

Testing Functions

Side can run a basic syntax check on the EPICS core configuration files, which finds many common basic errors. It can start a simulation of the real-time system, replacing calls to I/O hardware with simulated devices, this provides a further check of the syntax and logic, an can be used to test high level applications and user screens.

Installation functions

After testing, Side can use the SLS (or other site specific) programs to install configuration files in there correct final locations. At SLS files are installed directly from CVS to their final location, rather from a user directory, this avoids the risk that something is installed that is not in cvs

CONCLUSIONS

At SLS we have built a tool to integrate the EPICS control system development process. It meets the initial goals of making it possible for a novice user to quickly integrate a new device in the control system, while respecting the local rules for documentation and testing.

REFERENCES

[1] http://www.sls.psi.ch/controls/software/side