DEVELOPMENT OF PHOTON BEAMLINE AND MOTION CONTROL SOFTWARE AT DIAMOND LIGHT SOURCE

N.P. Rees, P.N. Denison, T.M. Cobb
Diamond Light Source, Chilton, Didcot, Oxon, UK

Abstract

Diamond Light Source has opened its first eight photon beamlines to the user community this year. We have developed the control software for the beamlines in parallel, adopting a common set of standards, tools, and designs across all beamlines. At the core of the control system is the EPICS toolset and the widespread use of the Delta Tau PMAC motion controller. The latter is a complex, but flexible controller that has met our needs both for simple and complex systems. We describe how we have developed the standard EPICS software for this controller so that we can use the existing EPICS interfaces, but also enables us to use the more advanced features of the controller.

INTRODUCTION

Since the last ICALEPCS conference [1], Diamond Light Source [2] has commissioned its first 8 beamlines, with first users being accommodated between January and August 2008. These beamlines will continue to be optimised as we install and commission a further 14 beamlines at a rate of approximately 4 per year. The software for the beamlines is based on the Experimental Physics and Industrial Control System (EPICS) toolkit [3, 4] for hardware control and Generic Data Acquisition (GDA) [5, 6] framework for the scientific user interface. The interface between the two layers is defined in terms of XML files which are generated from information in the EPICS database, and which, in turn, instantiate Java objects in the upper layer software. Both layers are sufficiently flexible to be developed in common across all beamlines, with minimal beamline specific software.

This paper focuses on the EPICS layer, with the first portion devoted to the build and release system shared by the beamlines, and the second portion focussing more specifically on our approach to motion control.

EPICS BUILD AND RELEASE SYSTEM

The core of the beamline build system is a spreadsheet (see Figure 1) that specifies all the beamline control points and their associated configuration parameters. Each control point type (Motor, Analog I/O etc) is a different worksheet and so all the parameters for a particular control point type can be viewed at one time. At build time, the spreadsheet is parsed by a Python script to generate a number of configuration files for creating EPICS databases, EPICS displays and the XML interface files for the GDA system. The values in the spreadsheet are inserted as EPICS parameters, or combined with information extracted from the EPICS database to describe GDA objects, or to provide parameters to substitute into displays to configure them they are invoked.

We have found this spreadsheet format an easy way to summarise and exchange a large amount of information. It also keeps the entire configuration in a single place, no matter what the ultimate use of the information is. We have considered using databases and/or XML files instead of a spreadsheet for storage, but have not implemented this at this stage. Whilst these formats do have some software advantages, they also have some disadvantages - the spreadsheet is really a lowest common denominator for this kind of information interchange.

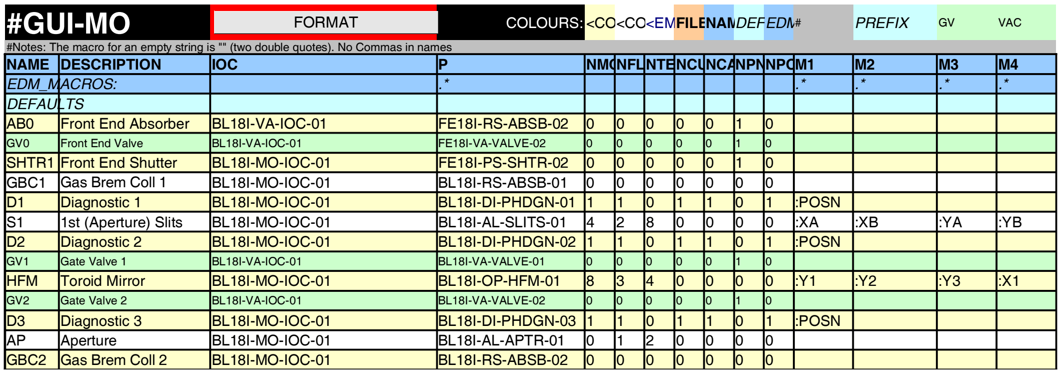
All beamline software is kept in a common Subversion repository. Software is developed in a test area, but during user beam time all systems boot off well defined released versions. The software release mechanism is triggered by the developer, but is done by a script in a well defined and controlled manner:

- The system is initially checked out into a scratch area and built and tested for errors.
- If this is successful the released is tagged and queued for building on the build server.
- The build server checks the release tree out read only from the subversion repository and builds it in the correct place.

To preserve the integrity of the releases the build server only exports the release file system in a read-only form, and it is not part of the normal authentication domain and so access is highly restricted.

A similar approach is taken for identifying which software release is actually running at a given time. All systems boot by reading a read-only file on the build server, which has a two-column table - the first column being the system name, and the second the system initialisation file. Updates to this file can only be done by updating the source code repository and queuing an update on the build server. Whilst this could clearly be done using a database, we have again adopted this files based approach for simplicity because it is the lowest common denominator.

Overall, this care in source code and release control has provided us with a firm development foundation. Whilst when a software problem surfaces it is often shared by many beamlines, we are confident when it is fixed it is also fixed across site.



#GUI-MO FORMAT COLOURS: <CO <CO <EM FILENAM DEF EDM # PREFIX GV VAC

#Notes: The macro for an empty string is "" (two double quotes). No Commas in names

NAME	DESCRIPTION	IOC	P	NMO	NFL	NTE	NCU	NCA	NPN	NPO	M1	M2	M3	M4
EDM_MACROS:			.*								.*	.*	.*	.*
DEFAULTS														
AB0	Front End Absorber	BL18I-VA-IOC-01	FE18I-RS-ABSB-02	0	0	0	0	0	1	0				
GV0	Front End Valve	BL18I-VA-IOC-01	FE18I-VA-VALVE-02	0	0	0	0	0	1	0				
SHTR1	Front End Shutter	BL18I-MO-IOC-01	FE18I-PS-SHTR-02	0	0	0	0	0	1	0				
GBC1	Gas Brem Coll 1	BL18I-MO-IOC-01	BL18I-RS-ABSB-01	0	0	0	0	0	0	0				
D1	Diagnostic 1	BL18I-MO-IOC-01	BL18I-DI-PHDGN-01	1	1	0	1	1	0	1	:POSN			
S1	1st (Aperture) Slits	BL18I-MO-IOC-01	BL18I-AL-SLITS-01	4	2	8	0	0	0	0	:XA	:XB	:YA	:YB
D2	Diagnostic 2	BL18I-MO-IOC-01	BL18I-DI-PHDGN-02	1	1	0	1	1	0	1	:POSN			
GV1	Gate Valve 1	BL18I-VA-IOC-01	BL18I-VA-VALVE-01	0	0	0	0	0	1	0				
HFM	Toroid Mirror	BL18I-MO-IOC-01	BL18I-OP-HFM-01	8	3	4	0	0	0	0	:Y1	:Y2	:Y3	:X1
GV2	Gate Valve 2	BL18I-VA-IOC-01	BL18I-VA-VALVE-02	0	0	0	0	0	1	0				
D3	Diagnostic 3	BL18I-MO-IOC-01	BL18I-DI-PHDGN-03	1	1	0	1	1	0	1	:POSN			
AP	Aperture	BL18I-MO-IOC-01	BL18I-AL-APTR-01	0	1	2	0	0	0	0				
GBC2	Gas Brem Coll 2	BL18I-MO-IOC-01	BL18I-RS-ABSB-02	0	0	0	0	0	0	0				

Figure 1: Example of a part of tab of a beamline development spreadsheet

MOTOR CONTROL

The Delta-Tau PMAC motion controller was chosen as a standard controller for the Diamond beamlines in part due to the ability to synchronise motion across many axes, which will become important as the beamlines are optimised. Other features of the controller have proved to be beneficial during development — particularly the fact that the entire PMAC range of products use the same communications protocol, and support for VME backplane, serial, and Ethernet connectivity has involved little extra work. This flexibility of form factor is proving to be most useful as more beamlines are designed — each with their own specific requirements. We can retain software compatibilty, using any of the PMAC family of controllers, whilst allowing scope for differing packaging and connectivity.

History

The EPICS 'motor' record [10] is a useful abstraction of the control points needed to drive a motor system, such as demand position, readback, limit positions etc. It also adds the ability to do a simple translation (offset and direction) between "dial" units and "user" units.

However, as it has evolved to support more and more controller types, the structure of the "device" and "driver" layers, between the motor record and the controller hardware have become increasingly convoluted and specific to each controller. The existing structure (shown in fig 2) limits the capabilities of the controller to the common superset that the motor record provides.

The sideways connectivity and the fact that a considerable amount of polling and communications code is in the common modules prevents easy access to the raw communications with the controller. This in turn prevents the use of extended capabilities of the controllers.

We have written a new interface layer for motor controller drivers which is clearly defined and supports the existing motor record, but also allows extension to support controller specific features. We have provided implementations of drivers for the Delta-Tau PMAC motor controller [11], and for a simulation motor [12]. Mark

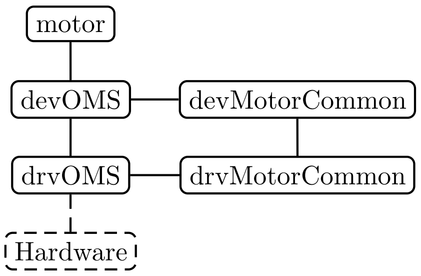


Figure 2: Existing software structure — complex interfaces

Rivers [9] has provided an implementation for the Newport MM4000 and XPS motor controllers.

New Driver Model

In order to support the new interface layer, we have written common "device" and "driver" layer components for the existing motor record. They have been written using the Asyn driver framework [8], allowing both the existing motor record and other standard EPICS records to simultaneously connect to the driver. The synchronisation of the access to the driver is performed by Asyn, which is extremely robust and has been widely used and tested.

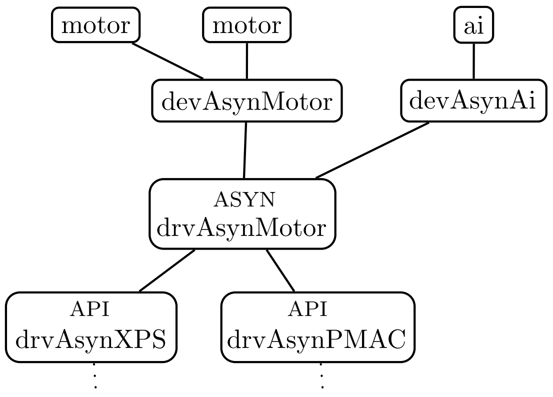


Figure 3: New structure

In the new structure (fig 3), there is a marked reduction in code, the new codebase being approximately 2-3 times smaller than before, both in terms of lines of code and number of files. More code is used in common, improving test coverage.

Extensions to support particular controller features can be introduced into the driver and the new interface layer (marked "API" in figure 3) extended, with upper level code written directly against that, or a feature of the Asyn interface can be used which allows new drivers to be inserted (*interposed*) along an existing communications path. Again, this feature is standard within Asyn, and has been widely tested.

One clear additional advantage of the use of the Asyn framework is that because an individual motor is connected to a named "port", switching from a real application to a simulation can be achieved solely by defining a simulation port, rather than one connected to a real motor controller. Thus the majority of the application is simulated unchanged. This has proved extremely useful when integrating EPICS with GDA, as we have been able to provide simulations of the entire motion control for a beamline well in advance of the arrival of any hardware.

The PMAC driver and the new framework have been in operation successfully on 8 beamlines for the majority of 2007.

Future Work

We are currently developing a driver within the same framework to control PMAC co-ordinate systems, rather than individual motors, which will allow more of the complex capabilities of the controller to be realised. For example, the controller supports complex kinematics to allow structures such as hexapods and multi-joint robot arms to be controlled. Previously, bespoke software had to be written for these systems, but now the standard motor records will be able to be used. The only bespoke software will then be the configuration of the controller itself.

The interface at present includes functions for co-ordinating simultaneous and profiled motion across several axes, but these are not yet implemented in any of the current drivers. We plan to extend the implementation of the drivers to encompass these interfaces, and to develop a set of EPICS records to control such motion.

Support for a number of other motor controllers is at an early stage, including work at Diamond on the OMS MAXv and OMS58 series controllers.

CONCLUSION

Common approaches and code across all beamlines at Diamond has contributed to successfully commissioning all of the initial beamlines simultaneously. Whilst the development overhead for the software has to be met anyway, we are now in a position to continue an aggressive programme of further beamline development and commissioning at a rate of 4 beamlines per year over the next 3 years.

Rewriting the motor controller infrastructure code has given us access to the extra capabilities of advanced motor controllers, with an easy upgrade path for further extensions in the future.

REFERENCES

[1] N.P. Rees, W.C. Pulford, M.A. Norbury, P.J. Leicester, E.L. Jones, M.T. Heron, P.N. Denison, "Development of Photon Beamline Software At Diamond" ICALEPCS 2005, Geneva, Switzerland, Oct 2005.

[2] R.P. Walker, "Progress with the Diamond Light Source Project", EPAC 2004, Lucerne, Switzerland, July 2004.

[3] L. Dalesio, J. Hill., M. Kraimer, D. Murray, S. Hunt, M. Claussen, C. Watson, J. Dalesio, "The Experimental Physics and Industrial Control System Architecture," ICALEPCS 1993, Berlin, Germany, Oct 1993.

[4] The Experimental Physics and Industrial Control System. http://www.aps.anl.gov/epics

[5] M.J. Enderby & W.C. Pulford, "The Generic Data Acquisition Project", NOBUGS 2004, Villigen Switzerland, October 2004.

[6] The Generic Data Acquisition Project. http://www.gda.ac.uk/

[7] Advanced Photon Source Beamline Control and Data Acquisition Group: synApps. http://www.aps.anl.gov/aod/bcda/synApps/index.php

[8] M.Kraimer, E.Norum, M.Rivers, G.Jansa, "asynDriver: Asynchronous Driver Support", http://www.aps.anl.gov/epics/modules/soft/asyn/

[9] M. Rivers, motorR6-2-2.tar.gz see [10], motorApp/NewportSrc/*

[10] T. Mooney, J. Sullivan, R. Sluiter et al., "EPICS: Motor Record and Device/Driver support.", http://www.aps.anl.gov/upd/people/sluiter/epics/motor/index.html

[11] http://www.gmca.aps.anl.gov/TPMAC2/tpmac3-4.zip, pmacApp/pmacAsynMotorSrc/*

[12] motorR6-2-2.tar.gz see [10], motorApp/motorSimSrc/*