# RECENT DEVELOPMENTS IN SYNCHRONISED MOTION CONTROL AT DIAMOND LIGHT SOURCE

B. Nutter, T. Cobb, M. Pearson, N. Rees, F. Yuan, Diamond Light Source, Oxfordshire, UK

*Abstract*

At Diamond Light Source the Experimental Physics and Industrial Control System (EPICS) [1] is used with a variety of motion controllers. The use of EPICS ensures a common interface over a range of motorised applications. We have developed a system to enable the use of the same interface for synchronised motion over multiple axes using the Delta Tau Programmable Multi Axis Controller (PMAC). Details of this work will be presented, along with an example and possible future developments.

## INTRODUCTION

Diamond Light Source is host to around twenty operational X-ray beamlines, each of which has various motorised devices in order to control and focus X-ray photons. These devices range from simple diagnostics and attenuators, to mirrors and double crystal monochromators, and complex multi-axis diffractometers and hexapod systems. There can be well over one hundred motors on a photon beamline, from simple stepper motors to servo motors, piezo motors and nano motors.

The control of motors is accomplished by using dedicated motor controllers that typically operate a servo control loop and amplifier for each axis. On top of this we have motor control software that sends commands to the controllers and reads back status information. At the highest level we make use of the EPICS control system and the Generic Data Acquisition (GDA) scientific framework [2] to provide a consistent user interface to each type of motor.

In addition to the standard control of each axis we perform synchronised motion on multiple axes. We do this at various levels, such as sending simultaneous move commands from the high level software, or defining a co-ordinate system at the motor controller level (which lets us achieve true synchronised motion).

## MOTION CONTROL HARDWARE IN USE AT DIAMOND

Diamond Phase I beamlines use the Delta Tau Turbo PMAC 2 VME Ultralight motor controller. This is a 32-axis controller which sits inside a VME crate alongside a CPU card running VxWorks and EPICS. The Ultralight controller uses a fibre optic communication link to two 16-axis Universal Motion and Automation Controller (UMAC) units, which in turn interface to limit switches, encoders and external amplifier crates.

At the end of Phase I Diamond underwent a tender process for Phase II beamline motion controllers. It was determined that the use of Ethernet-based controllers would be a more flexible and economic alternative than the traditional VME-based controller. The controller we chose was the Delta Tau Geobrick Low Voltage Integrated Motion System (LV IMS). This is a Turbo PMAC 2 controller in a 4U enclosure, with an integrated amplifier board. The amplifier board has 4 or 8 axes of output, each of which can be configured in software to be either a stepper or a servo amplifier. This made it possible not only to control DC stepper motors, with various encoder feedback mechanisms, but also to control brushless and DC servo motors. Step and direction outputs are also available to control pico motors and ultra-high resolution external stepper amplifiers.

The product has suffered somewhat from early design teething troubles, mainly concerning the amplifier firmware. However, all solutions to date have been compatible with the existing hardware platform. We are also seeing an increasing number of very small motors, typically below 500mA, and the current resolution of the Geobrick amplifiers (which are rated at 5A continuous, 15A peak, but have a zero current set point accuracy of nearly 100mA) is becoming an issue.

Diamond has a number of other motor controllers on site. These include the Newport XPS controller, mainly used on Newport diffractometers, and a small number of Aerotech Ensemble controllers, required to take advantage of the smoothness and precision of their inbuilt linear amplifier stages. We also have many piezo stack control devices from Queensgate, PI and Jena, which are controlled by either analogue signals or RS232.

## EPICS MOTOR CONTROL SOFTWARE

For each specific motor controller there is a software driver based on the Asyn software framework [3]. Asyn is a low-level driver software abstraction framework, providing synchronous and asynchronous commands and responses. The drivers are either developed in house (normally building on the work of similar drivers) or use is made of drivers developed elsewhere. Above these drivers is a consistent user interface in the form of an EPICS record, called the Motor Record. This provides the user with functions such as *move*, *stop*, *home*, etc., as well as support for modifying the motion properties such as velocity or acceleration. The EPICS Motor Record is documented in full at [4].

The Asyn-based motor driver framework that sits underneath the EPICS motor record has been developed by Diamond and the Advanced Photon Source, and has been previously been presented [5]. The advantages that the driver framework gives us are mainly the ease of adding new drivers for new controllers, the ability to use multiple EPICS records to control a motor and its

parameters and the ability to monitor a motor status asynchronously (by using an event driven polling thread).

## GDA SOFTWARE INTERFACE TO EPICS MOTOR SOFTWARE

The GDA is a data acquisition system developed at Diamond to provide a user interface and experiment platform for Diamond beamline users. It has a client-server architecture and includes support for the EPICS channel access protocol. A Java implementation of the EPICS channel access protocol, called Channel Access for Java (CAJ) [6], is used to provide both synchronous and asynchronous communication with EPICS. When configuring the GDA for a beamline, the EPICS motor record name is obtained dynamically from an EPICS-GDA interface XML file, generated during EPICS build and deployment, to ensure consistency between the two systems.

Motor objects in GDA are designed to be in state synchronisation with the underlying EPICS motor record to support the simultaneous use of GDA and EDM (an EPICS-aware Motif-based graphical toolkit called Extensible Display Manager [7]). Access control of EPICS motors is also supported, and can be used to restrict access to particular devices such as insertion devices or front end motors.

Beamline users normally collect data and access EPICS motors using the GDA scan mechanism. This is a very flexible framework capable of capturing a complete data set in experiments, including metadata such as beam conditions and sample environments. It allows users to move motors in multiple dimensions, typically from a start position to an end position in predefined steps, and to collect arbitrary data during a scan.

## COORDINATED MOTION SOFTWARE

We have been able to achieve co-ordinated multi axis motion in several ways:

- Sending simultaneous move commands to multiple axes at a high level, using GDA to control multiple motor records. An example of this is slit scanning, where the GDA performs its own calculations for slit gap and centre positions. This is achieved by creating a GDA object to represent the combined motor axes. The object can then be used by the GDA scan mechanism.
- Sending simultaneous move commands to multiple axes at the EPICS level. A set of EPICS records work together to calculate combined motion and send the move commands one-by-one to the motor records. This is similar to the GDA-based mechanism, except in this case the GDA only knows about one EPICS motor record, for example a slit gap 'motor'.
- Using 'deferred moves'. This is a technique that involves the driver queuing up move commands and sending them all at once to the motor controller. This technique is useful if one is attempting to move multiple axes that are 'grouped' together on the motor controller itself. When axes are grouped together, an axis can go into the 'moving' state when we are actually only moving a different axis. This causes an issue when using the EPICS motor record and attempting to drive two axes by sending separate commands, because, to avoid ambiguity, EPICS needs the motor to be idle before the move starts if the client also wants to be notified asynchronously of the move completion. In effect, the motor record does not permit a move to be sent while an axis is already in motion (unless the previous move is stopped first). Deferred moves are a way around this issue, and provide a greater degree of move synchronisation than can be achieved by using high level software alone.
- Making use of the coordinated motion capabilities of the controller, and defining a 'coordinate system' for multiple axes.

We will expand on the 4th option. On the PMAC controller, this method involves setting up the axes in a co-ordinate system, which means defining forward and inverse kinematic calculations and defining a PMAC PLC program to provide combined axis position read-back. An example for a pair of slits is shown in Figure 1.

```
&2
#1->I
#2->I
OPEN FORWARD
CLEAR
Q1=(P1+P2)/2
Q2=P1-P2
CLOSE

OPEN INVERSE
CLEAR
P1=Q1+Q2/2
P2=Q1-Q2/2
CLOSE
```

Figure 1: PMAC forward and inverse kinematics.

In Figure 1 the notation &2 defines co-ordinate system number 2, and the #1->I means that axis 1 will be placed in that co-ordinate system. The Q1 and Q2 variables are from the PMAC 'Q-variable' range on the PMAC which have co-ordinate system scope. Here we are using Q1 and Q2 to represent the combined motion. P1 and P2 are used to represent the real axis position, based on the forward kinematics. We also need to define a PLC program to calculate the real combined motion positions, and this is used for position read-back by the EPICS driver.

```
OPEN PLC 2
CLEAR
Q81= (m162 + m262)/2  ;real center
Q82= (m162 - m262)    ;real gap size
CLOSE
```

Figure 2: Position readback PMAC PLC program.

In Figure 2 we are using Q81 and Q82 to hold the combined motion positions, calculated from the actual motor positions held by the PMAC (which are the mX62 variable, where the X is the axis number). In the above examples we have omitted scale factors, which are required to convert to and from engineering units and PMAC position 'counts'. We need to define a PLC program such as this in order to provide real combined axis position read-back, because the P1 and P2 variables are specific to the kinematic buffer.

Finally, there is also a PMAC motion program, which is executed whenever a combined motion command is sent to the controller. The motion program then makes use of the co-ordinate system, which we defined above.

The EPICS driver for the PMAC has been modified to support the functionality we have described above. There is a coordinate system driver that writes combined motion demand values into Q-variables on the PMAC and executes the motion program on the PMAC. It also reads the Q-variables from the PLC program in order to obtain the position for the combined axes.

This is implemented in the EPICS co-ordinate driver in the motorAxisMove function. It is accomplished by sending three commands to the controller:

- Write new demand values into the appropriate Q-variables for the coordinate system
- Abort the current move (to ensure that the axes are enabled)
- Run the motion program

In the co-ordinate system driver there is a separate thread which polls the actual combined motion axis positions (provided by the PLC program described above). This provides continuous position updates to the motor record, at a configurable rate (typically 10 or 20Hz).

The EPICS coordinate system driver fits into the same asyn motor framework as the existing drivers, and so can be used with the same EPICS motor record code. This means that we have a consistent user interface to both real axes and combined coordinated axes. The PMAC co-ordinate system driver is packaged with the tpmac EPICS support module [8], or is available stand-alone as the pmacAsynCoord module.

## DIAMOND STANDARDS FOR DELTA TAU PMAC DEVELOPMENT

The coordinated PMAC driver does make some assumptions about how the PMAC co-ordinate system has been set up. For example, it assumes that the PLC read-back positions will be stored in the variables Q81 to Q89. In order to ensure that PMAC software development is kept in sync with the EPICS driver we developed a set of PMAC programming standards, both for co-ordinate systems and general purpose PMAC PLC programs. These programming guidelines have been written down and we routinely forward them onto equipment suppliers who develop PMAC motion control systems for us. They also serve as a useful guide within the group when we develop new PMAC software. We hope that these standards will be useful to other EPICS sites that are using PMAC controllers, as this will ensure the PMAC co-ordinate system driver will be applicable outside Diamond.

A selection of the standards we have developed is described here. First, we list the reserved PLC program numbers and their intended use in Fig.3.

| | |
|---|---|
| PLC1 | initialization routines |
| PLC2 | motion stop detection |
| PLC3 | amplifier enable routine |
| PLC4 | encoder loss detection |
| PLC5 | CPU use reporting |
| PLC6 | amplifier setup |
| PLC7 | auto amplifier power off |

Figure 3: Reserved PLC numbers and their function.

PLCs 8 to 16 are free for application-specific code and PLCs 17 to 31 are reserved for co-ordinate system position reporting (as described in this paper). In order to avoid conflicts PLCxx (where xx is the PLC number) should only use PMAC P-variables (which have global scope) in the Pxx00 to Pxx99 range.

Motion programs and co-ordinate systems should use Q-variables where possible, and only use P-variables in the range P3200 to P4200 (to avoid conflicting with PLC programs). For co-ordinate systems we reserve the following Q-variable ranges:

- Q71 to Q79 – co-ordinate system demand positions.
- Q81 to Q89 – co-ordinate system readback positions (these are the positions calculated by the position reporting PLC program).
- Q400 to Q439 – backlash compensation settings in coordinate systems.

In addition to the above, Delta Tau reserve various variable ranges for their own use. We also document various PMAC programming guidelines, such as best practice for developing a homing PLC program. For example it is bad practice to disable limit switches during a homing program, in case the program fails and therefore fails to re-enable the limit switches.

## CONCLUSION AND FUTURE DEVELOPMENTS

Significant progress has been made with the new Delta Tau Geobrick controller on the final Phase II beamlines. On most beamlines we have deployed PMAC co-ordinate systems to drive combined motion systems such as mirrors and slits. We have found this to be very reliable and to provide true synchronised motion at the controller level.

Further work is planned on the EPICS PMAC driver to provide support for trajectory scanning. Eventually we are aiming to have a consistent trajectory scanning interface that is controller- and driver-independent, similar to the existing abstract interface that the motor record provides for standard axis motion.

## REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] http://www.opengda.org/
[3] http://www.aps.anl.gov/epics/modules/soft/asyn/
[4] http://www.aps.anl.gov/bcda/synApps/motor/
[5] N.P. Rees, P. N. Denison, T. M. Cobb, "Development of Photon Beamline and Motion Control Software at Diamond Light Source", ICALEPCS 2007, Knoxville, Tennessee, USA.
[6] http://epics-jca.sourceforge.net/caj/
[7] http://ics-web.sns.ornl.gov/edm/
[8] http://www.gmca.anl.gov/TPMAC2/index.html