synApps: EPICS APPLICATION SOFTWARE FOR SYNCHROTRON BEAMLINES AND LABORATORIES*

T. M. Mooney[#], ANL, Argonne, IL 60439, U.S.A.

Abstract

synApps[1] is a collection of EPICS [2] application software originally intended to support the needs of scientists performing experiments at synchrotron-radiation beamlines. The collection contains general-purpose software that extends or exploits capabilities of EPICS base, and a large amount of instrument-specific software that uses EPICS to control and provide a user interface for off-the-shelf electronics.

This paper will provide an overview of synApps, describe how the software is deployed at the Advanced Photon Source, and highlight recent additions.

OVERVIEW

synApps is a collection of EPICS modules that supplement the record types, device support, and other software infrastructure included in EPICS Base. Because it was written to support scientists conducting a wide variety of experiments, most of the software in synApps is general in purpose, and was engineered to serve many needs at once, by abstracting from specific sets of requirements general solutions for classes of problems.

But this focus on general solutions does not distinguish synApps from other EPICS-application software. Most EPICS software is general purpose, in part because EPICS is a collaborative effort. synApps differs from mainstream EPICS-application software in three ways: it contains a small amount of synchrotron-specific software, it provides infrastructure to support run-time programming, and it provides infrastructure to support data acquisition.

synApps consists of the following modules, grouped according to the kinds of applications they support.

General-Purpose Modules

- **autosave** – Saves the values of EPICS process variables, and restores them after a reboot.
- **busy** – Extends EPICS' execution tracing to include client software.
- **calc** – Provides variations of the EPICS *calcout* record for systems of expressions (*transform* record), string expressions (*sCalcout* record), and arrays (*aCalcout* record).
- **sscan** – Supports *scans* (systematically set conditions; acquire and store data).
- **std** – Supports scalers, sequences of operations, and PID loops.

Hardware Specific Modules

- **areaDetector** – Supports multidimensional detectors.
- **camac** – Supports CAMAC hardware.
- **dac128V** – Supports an IndustryPack digital-to-analog converter.
- **delayGen** – Supports delay generators.
- **dxp** – Supports DXP digital-signal processing spectroscopy systems.
- **ebrick** – Supports the EPICS Brick, a PC104-based computer running Linux, as an EPICS *IOC* (Input/Output Controller).
- **ip** – Supports various message-based (e.g., serial, GPIB) devices.
- **ip330** – Supports an IndustryPack analog-to-digital converter.
- **ipUnidig** – Supports an IndustryPack digital I/O module.
- **love** – Supports Love controllers.
- **mca** – Supports multichannel analyzers and multichannel scalers.
- **modbus** – Supports Modbus devices.
- **motor** – Supports stepper and servo motors.
- **quadEM** – Supports a four-channel electrometer.
- **softGlue** – Provides user-programmed digital logic and I/O.
- **vac** – Supports vacuum-related devices.
- **vme** – Supports VME hardware.

Synchrotron-Radiation Specific Modules

- **optics** – Supports X-ray monochromators, slits, optical tables, and other synchrotron-radiation equipment.

Other Software in synApps

- **xxx** – Provides a template for an EPICS IOC directory using synApps.
- **utils** – Provides miscellaneous software related to synApps, including support for migrating from one version of synApps to another, support for a data-file format used by synApps scan software, and support for rapid EPICS-database programming.

Software Distributed with synApps

synApps makes use of the following EPICS modules that are not part of synApps, but are distributed with it: **allenBradley**, **asyn**, **ipac, seq**, **stream**, and **vxStats**.

*The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

[#]mooney@aps.anl.gov

RUN-TIME PROGRAMMING

Because many synApps users are scientists conducting experiments, and because experimental work is typically less well understood in advance than are other activities supported by EPICS, synApps places a much greater emphasis on support for programming at run time than is typical of EPICS-application software. Most synApps IOCs load records reserved for run-time programming; and most synApps record types are supported by displays, online documentation, and autosave-request files for this purpose. Also, many synApps record types check and report to the user the states of their link fields, so that run-time link errors can be recognized promptly.

In this context, "programming" does not mean code development or scripting, but rather the configuration and linking together of EPICS records. A collection of linked EPICS records – an EPICS *database* – can be viewed as a program in a very high-level language. For example, an input record linked through a calculation record to an output record can implement a feedback loop.

An EPICS database configured at run time is not distinguishable in any essential way from a similar database configured at build time: it has the same speed and efficiency, and it can drive or be driven in the same ways. Thus, run-time-programmed databases can be layered, sequenced, event driven, or scanned, and the result for end users is an extraordinarily powerful and versatile capability to diagnose and solve problems as they arise during an experiment, and to modify solutions to those problems as they become better understood.

The principal means by which run-time programming is accomplished in EPICS is the redefinition of an EPICS link. In early versions of EPICS, links could not be changed at run time. The first programmable links were implemented by Marty Kraimer for use by the synApps *scan* and *wait* records (originally developed by Ned Arnold), and they were initially viewed as support for scans. But the *wait* record quickly came to be applied more widely for its run-time programming capability, and the result was powerful enough to motivate the development (by Marty Kraimer, Bob Dalesio, Jeff Hill, and others) of support for run-time redefinition of *all* EPICS links.

The impact of run-time-programmable links on synApps' development was profound: most synApps record types, databases, and displays came to be developed with run-time programming as an objective, and the automated saving and restoring of EPICS PV values (*autosave*, originally developed by Bob Dalesio) acquired new urgency and purpose.

Recently, the notion of run-time programming was extended to run-time development of digital hardware, in the **softGlue** module.

Rapid Prototyping

Soon after support for run-time programming became pervasive in synApps, the capability was recognized also as a rapid-prototyping tool – a way for EPICS-database developers to test and combine database fragments without rebooting. The principle defect in this development approach was the lack of a convenient way to save run-time programming in the standard form of an EPICS database file.

A wxPython program, *snapDb*, was written to address this problem. Using snapDb, a user or developer can produce a loadable EPICS database from run-time-programmed fragments simply by using MEDM's Drag-And-Drop capability to enter a PV name from each record into a list. snapDb then reads all fields of the listed records, and writes an EPICS database file. snapDb can also write an MEDM display file for the database.

DATA ACQUISITION

Synchrotron-radiation users spend a lot of time scanning – systematically varying conditions, acquiring data under those conditions, and storing the data for later analysis. The **sscan** module is dedicated almost entirely to this purpose, comprising the *sscan* record, which performs multidimensional scans; the *recDynLink* library, which manages Channel-Access connections for the *sscan* record; and the *saveData* task, which writes scan data to disk.

Other synApps modules involved heavily in data acquisition are the **areaDetector**, **mca**, **dxp**, and **std** modules. These modules support specific hardware, such as scalers, multichannel analyzers, and two-dimensional detectors, and do so in a way that permits EPICS clients, including the *sscan* record, to trigger data acquisition, wait for acquisition to complete, and collect the resulting data.

Completion Reporting

EPICS Base contains support for tracing the execution of a linked set of records (i.e., a *database*), and for signaling the completion of that execution to the client that caused it to occur. Within an IOC, tracing is performed by the EPICS *putNotify* facility. Execution spanning more than one IOC can be traced by using the Channel Access function *ca_put_callback()* to make the completion of a record in one IOC contingent on the completion of execution in another IOC.

synApps' data-acquisition strategy relies heavily on *putNotify* execution tracing, and synApps provides several record types engineered to extend *putNotify* across IOCs, in addition to serving their primary purposes:

- ***sscan*** – This record performs a one-dimensional scan. Several ***sscan*** records can be linked to perform multidimensional scans.
- ***sseq*** – This record is a variant of the EPICS ***seq*** record, which performs a programmed sequence of operations. The ***sseq*** record differs from ***seq*** in that it can read and write strings as well as numbers, and it can wait for completion between operations.
- ***swait*** – This record is an early prototype of the EPICS ***calcout*** record, and is one of the first EPICS records whose links could be modified at run time. It differs from ***calcout*** in that it uses the *recDynLink* library, and its output link waits for completion.

- ***aCalcout*** – The array calcout record is a variant of the EPICS ***calcout*** record, and differs from it by supporting array fields and expressions in addition to scalar fields and expressions. The ***aCalcout*** record also can wait for completion of execution triggered by its output link.
- ***sCalcout*** – This record is similar to the *aCalcout* record, but it supports strings, instead of arrays.
- ***busy*** – This record functions as a proxy for the execution performed by a Channel Access client. EPICS *putNotify* cannot directly trace execution by a client, so the busy record (which can be traced) pretends to be executing until the client tells it to stop.

Most of the listed record types have links that can use the Channel Access function, ca_put_callback(), to initiate execution, and that can wait for the resulting callback, which indicates that the execution has completed. The *busy* record is an exception: its purpose is to be *driven* by a ca_put_callback(), and to look busy until a client tells it to stop, whereupon its completion yields a callback indicating that the client is done.

Automated testing

The infrastructure with which synApps supports data acquisition by users is also useful to developers, for diagnostic and testing purposes. The *sscan* record, for example, has been used (with other run-time configured software) to diagnose race conditions, by systematically varying the time between the execution of application code, and a simulated response from driven equipment.

The combination of the *sscan* record and the **softGlue** module extends this diagnostic and testing capability to digital hardware.

DEPLOYMENT AT APS

The deployment of synApps at the Advanced Photon Source has evolved in response to an increasing number of beamlines, an increasing emphasis on computer security, and the similarly driven evolution of the EPICS module structure. Originally, synApps modules (called "Apps" in those days) were deployed alongside IOC directories on a file server to which beamlines had read/write access; there was not a clear distinction between support modules and application modules.

As the number of beamlines increased, and the separation between beamline subnets became more complete and more rigidly enforced, synApps was split into support modules and IOC directories. Support modules (all modules except **xxx**) are now hosted, along with EPICS Base, on a central file server, and both are distributed via *rsync* to read-only partitions on secondary servers dedicated to individual beamlines. The IOC directories are now created on read-write partitions of those secondary servers, and begin as copies of the synApps **xxx** module, which collects support from all other synApps modules and builds loadable executables and database-definition files for use by one or more IOCs.

One effect of this evolution has been the concentration of display files and autosave-request files in support modules, rather than in application directories. In turn, this concentration led to the development of an include-file capability in autosave, so that module developers could define the PVs needed to restore databases implemented in those modules, and IOC directories could simply include the request files for the databases they needed to maintain through IOC reboots.

Another effect has been an increasing reliance on MEDM's ability to build displays using *Composite Objects* – display files that can be included within other display files and customized using macro substitution.

RECENT DEVELOPMENTS

areaDetector

The **areaDetector** module provides a general-purpose interface for area (2-D) detectors in EPICS. It supports a wide variety of detectors and cameras, ranging from high-frame-rate CCD and CMOS cameras, pixel-array detectors such as the Pilatus, and large-format detectors like the MAR-345 online imaging plate.

Among recent improvements in **areaDetector** is the evolution of support for plug-ins, which provide a mechanism for device-independent real-time data analysis, such as regions-of-interest and statistics.

softGlue

The **softGlue** module provides EPICS users and developers with the capability of creating and modifying simple digital electronic circuits, connecting those circuits to external devices, and controlling or driving the circuits – all by writing to EPICS process variables.

ACKNOWLEDGMENTS

synApps is the product of a collaboration including more people than can practically be named here. Most of the EPICS developers at the Advanced Photon Source – in particular, those in the Beamline Controls and Data Acquisition Group – and many at other EPICS sites, have contributed code or ideas, tested on or ported to new architectures, improved the build software, etc. Mark Rivers is responsible for much of the recent work in synApps, and maintains a large fraction of synApps, including the **areaDetector**, **camac**, **dac128V**, **dxp**, **ip330**, **ipUnidig**, **mca**, **modbus**, and **quadEM** modules.

REFERENCES

[1] http://www.aps.anl.gov/bcda/synApps.
[2] http://www.aps.anl.gov/epics.