EPICS high-performance control and data acquisition using synApps

Mark Rivers
GeoSoilEnviroCARS
Advanced Photon Source
University of Chicago
Outline

• Goals for Beamline Control and Data Acquisition (BCDA) systems
• Why EPICS
• What’s available in synApps
  – Basics
  – High performance
  – What needs to be done
Characteristics of an Ideal Beamline
Control and Data Acquisition System

1. Easy to use
   • Intuitive Graphical User Interface (GUI)
   • Customized for the specific experiment

2. Fast, efficient
   • Expert users don’t want to navigate GUI

3. Powerful, able to accomplish any task, even if it has never been done before

4. Able to use existing applications that have substantial development and user familiarity (GDA, Blu-Ice, SPEC, someone’s favorite EXAFS program, etc.)

5. Can use different applications to run the same equipment for different experiments (i.e. not tied into Labview or SPEC drivers)

6. Available off-the-shelf, no development required (you wish!)
My background and perspective

• Worked on beamline control and data acquisition software at NSLS from the earliest days (1983) of LeCroy MCA with built-in CAMAC and 8-bit OS, through VMS/CAMAC, to EPICS/VME.

• Started with EPICS at APS in 1995. Software for our sector (GSECARS, sector 13) but also developed modules that run on almost at all beamlines, as well as Swiss Light Source, Diamond, Australian Synchrotron, etc.

• EPICS binaries compiled at GSECARS are running 5 NSLS beamlines now. Precompiled binaries and source code for EPICS detectors running on ~6 other NSLS beamlines

• I strongly believe that the distributed nature of EPICS, and the large amount of existing working code make it the logical choice for beamline controls.

• User interfaces have been the least coordinated and need the most work

• EPICS is a very successful collaboration. Beamline controls has been APS-dominated in the past, but with new contributions from Diamond (e.g. GDA) and hopefully NSLS-II this will evolve
Ten really neat things about EPICS for Beamline Control and Data Acquisition

• There are lots of beamlines using it – NSLS, APS, AS, SLS, DLS, SSRF
• Most record types for beamline components and common operations already exist
  – Motors, monochromators, multi-channel analysers, A/D, D/A, digital I/O, 2-D detectors, scanning, save/restore, save_data, etc.
• Lots of device support and drivers for commercial devices in these classes already exists
• There are high-level applications available for free (e.g. EXAFS) or commercially (e.g. SPEC)
• There is a lot of expertise available close by because NSLS-II accelerator will use EPICS
• You can have a basic beamline up and running very quickly with supported hardware, add features as time permits
  – I converted 4 NSLS beamlines from CAMAC/VMS to EPICS in less than 1 week
Overview of synApps

• A collection of EPICS applications for synchrotron-beamline users. Collaboration between APS BCDA group and other beamlines at APS and elsewhere

http://www.aps.anl.gov/bcda/synApps/

• EPICS modules and build/configuration tools:
  – Modules:
    • Infrastructure: autosave, busy, calc, camac, modbus, sscan
    • Control: dac128V, delayGen, ip, ipUnidig, love, motor, optics, vac, vme
    • Detectors/acquisition: areaDetector, dac128V, dxp, ip330, mca, quadEM
  – >250,000 lines of code (C, SNL, databases), hundreds of medm screens

• Related clients, libraries, and visualization tools:
  • IDL and Python: mca display, ezcaIDL, ezcaScan, scan file visualizer, ez_fit, HDF translator/browser, Ascii-format plotter, image processors, etc.
**synApps modules**

- Modules contain the following kinds of support:
  - Compiled code; libraries
    - E.g., record, device and driver support
    - State-Notation-Language programs
  - EPICS databases and autosave-request files
    - A database is a *program* written in a high-level language.
    - One or more copies of a database can be run, each with its own private variables (PV’s).
    - The database designer recommends PV’s to be autosaved by naming them in a .req file; you can override with a private copy of the file.
  - MEDM-display files
    - The default user interface
  - Documentation
Other EPICS modules used by synApps

- asyn (driver/device support)
- ipac (Industry Pack support)
- seq (State Notation Compiler)
- stream (ASCII device support: serial, GPIB, TCP)
- vxStats (IOC statistics)
- allenBradley (PLC support)
asyn

- Well defined interface between EPICS device support and driver
- Standard asyn device support that can be used in nearly all cases
- In last 8 years I have written a LOT of new drivers and I have written NO device support, just use standard asyn device support
- I believe asyn should be used to write all EPICS device drivers, not just “asynchronous” drivers like serial, GPIB and TCP/IP.
  - All of my drivers (except one old one) use asyn
- There is a new C++ base class, asynPortDriver, that makes it very easy to write a new driver
  - All of my areaDetector, D/A, binary I/O, and most recently motor drivers use asynPortDriver
**autosave module**

- Records latest values of selected EPICS PVs; restores those values when the ioc restarts.
  - not an archiver; only the latest value is saved
  - not the same as saveData, which writes scan data
  - When a list list of PV’s is saved, the entire list is written, even if only one PV has changed.

- Can save/restore any scalar or array-valued PV
...autosave module

- PV lists can use include files (e.g., `<database_name>.req`), include path.
  - Database developer can supply default include file with database.
  - User can override with custom include file.
- Save triggers:
  - on change of any PV in the list
  - periodically
  - on change of a trigger PV
  - manual
- User can reload save sets.
- User can choose to save redundant files
  - Autosave reports status via EPICS PV’s
areaDetector module

• Drivers for many detectors popular at synchrotron beamlines
  – Handle detectors ranging from >500 frames/second to <1 frame/second

• Basic parameters for all detectors
  – E.g. exposure time, start acquisition, etc.
  – Allows generic clients to be used for many applications

• Easy to implement new detector
  – Single device-driver C++ file to write. EPICS independent.

• Easy to implement detector-specific features
  – Driver understands additional parameters beyond those in the basic set

• EPICS-independent at lower layers.

• Middle-level plug-ins to add capability like regions-of-interest calculation, file saving, etc.
  – Device independent, work with all drivers
  – Below the EPICS layer for highest performance
areaDetector drivers

- Simulation driver
  - Produces calculated images up to very high rates. Implements nearly all basic parameters, including color. Useful as a model for real detector drivers, and to test plugins and clients.

- Prosilica driver
  - Gigabit Ethernet cameras, mono and color
  - High resolution, high speed, e.g. 1360x1024 at 30 frames/second = 40MB/second.

- Firewire (IEEE-1396 DCAM)
  - Vendor-independent Firewire camera drivers for Linux and Windows

- Roper driver
  - Princeton Instruments and Photometrics cameras controlled via WinView
areaDetector drivers

• PVCAM driver
  – Princeton Instruments and Photometrics cameras controlled via PVCAM library
• Pilatus driver
  – Pilatus pixel-array detectors.
• marCCD driver
  – Rayonix (MAR-USA) CCD x-ray detectors
• ADSC driver
  – ADSC CCD detectors
• mar345 driver
  – marResearch mar345 online image plate
• Perkin-Elmer driver
  – Perkin-Elmer amorphous silicon detectors
areaDetector plugins

• Designed to perform real-time processing of data, running in the EPICS IOC (not over EPICS Channel Access)
• Receive NDArray data over callbacks from drivers or other plugins
• Plug-ins can execute in their own threads (non-blocking) or in callback thread (blocking)
  – If non-blocking then NDArray data is queued
    • Can drop images if queue is full
  – If executing in callback thread, no queuing, but slows device driver
• Allows
  – Enabling/disabling
  – Throttling rate (no more than 0.5 seconds, etc)
  – Changing data source for NDArray callbacks to another driver or plugin
• Some plugins are also sources of NDArray callbacks, as well as consumers.
  – Allows creating a data processing pipeline running at very high speed, each in a difference thread, and hence in multiple cores on modern CPUs.
areaDetector plugins

• NDPlugInStdArrays
  – Receives arrays (images) from device drivers, converts to standard arrays, e.g. waveform records.
  – This plugin is what EPICS channel access viewers normally talk to.

• NDPluginROI
  – Performs region-of-interest calculations
  – Select a subregion. Optionally bin, reverse in either direction, convert data type, scale by a constant.

• NDPluginColorConvert
  – Convert from one color model to another (Mono, Bayer, RGB pixel, row or planar interleave)

• NDPluginMJPEG (Diamond)
  – MJPEG server that allows viewing images in a Web browser.
areaDetector plugins (new in R1-6)

- **NDPluginStats**
  - Calculates statistics on an array
  - Replaces the statistics calculations that were previously performed in the ROI plugin.
  - Adds new statistics, including the centroid position and width.
  - Computes X and Y profiles, including average profiles, profiles at the centroid position, and profiles at a user-defined cursor position.

- **NDPluginProcess**
  - Does arithmetic processing on arrays
  - Background subtraction.
  - Flat field normalization.
  - Offset and scale.
  - Low and high clipping.
  - Recursive filtering in the time domain.
  - Conversion to a different output data type.

- **NDPluginOverlay**
  - Adds graphic overlays to an image.
  - Replaces the "Highlight ROIs" function that was previously provided in the ROI plugin.
  - Much more general, and can be used to display not only ROIs, but multiple cursors, user-defined boxes, etc.
Statistics plugin

GeoSoilEnviroCARS

NSLS-II Data Acquisition and User Interfaces Workshop
April 19, 2010
Overlay plugin

Centroid of laser pointer calculated by statistics plugin
Cursor overlay X, Y position linked to centroid
Processing plugin

GeoSoilEnviroCARS
Processing plugin
30 microsec exposure time

No filtering

N=100 recursive average filter

GeoSoilEnviroCARS

NSLS-II Data Acquisition and User Interfaces Workshop
April 19, 2010
calc module

• Support for evaluation of string or numeric expressions entered at run time (or at database-configure time)

• Records
  – sCalcout – like calcout, but also supports string expressions; user can specify wait-for-completion.
  – swait – like calcout, but uses recDynLink (no “PP MS” link attributes)
  – transform – like 16 calcout records that share a PV data pool

• Other code
  – string-calc engine
  – sCalcout soft device support (with wait-for-completion option)
  – interpolation routines for genSub record
  – (yet another) averaging routine for sub record
**dac128V module**

- device support, database, and MEDM displays for dac128V IndustryPack module
  - 8-channel, 12-bit DAC
  - Uses asynPortDriver. 160 lines of C++ code.
  - Uses asyn standard device support
  - Slow or fast feedback device with EPID record using standard asyn interfaces
- Used at GSECARS for
  - Monochromator second crystal feedback
  - Furnaces and heaters: use Keithley multimeter to read temperature, analog control 1kW power supply to drive heater
  - Easy to set up in a few minutes for a new piece of equipment user brings
- Good model for D/A converter support, even if you use different hardware
**dxp module**

- Driver support, databases, and MEDM displays for XIA digital signal processing spectroscopy systems
  - **DXP4C2X:** CAMAC module for multi-element detectors. 4 detectors per CAMAC module. Obsolete, but still in use at some beamlines.
  - **Saturn:** standalone unit for single-element detectors. This is also sold in an OEM version inside the Vortex electronics from SII
  - **xMAP:** PXI module for multi-element detectors. 4 detectors per PXI module. Faster than Saturn and DXP2X, and with high-performance features.
  - **Mercury:** New 4-channel module very similar to the xMAP, but in a standalone box like the Saturn with a USB 2.0 interface.

- Standard EPICS records for setting DXP parameters (peaking time, etc.)
- asyn driver support for the mca record
- Fast mapping with the PXI xMAP module
- Uses asynNDArrayDriver from areaDetector, which uses asynPortDriver from asyn
- More details in talk tomorrow
**ip, vac, delayGen modules**

- device support, SNL code, databases, and MEDM displays for many message-based devices
- Supports lots of benchtop RS-232/GPIB/TCP-IP instruments, e.g.
  - Keithley multimeters
  - Stanford Research Systems current amplifiers
  - Eurotherm and Lakeshore temperature controllers
  - Omega panel meters
- vac – vacuum device support
  - MKS ion gauge controllers
  - Physical Instruments ion pump controllers, TSP
- delayGen - Digital pulse/delay generators
  - SRS DG535, DG645
  - BNC 505 – Used at GSECARS to pulse lasers, trigger CCD after laser pulse
- Much of the support pre-dates the “stream” module, so uses other device support. Should use streamDevice for all new support where possible, and convert existing support to stream as resources permit.
**ip330 module**

- Asyn driver support, databases, and MEDM displays for the IP330 ADC IndustryPack module
  - Does not use asynPortDriver, should be converted to do do
- 16/32 channel, 16-bit ADC
  - New readings at >2kHz. Interrupt-driven callbacks to device support with new values
- Device support
  - Standard asynInt32Average for periodic, averaged reads of ADC channels
  - fastSweep, with the MCA record, puts new values in an array for using ip330 as a waveform-digitizer
  - Fast feedback device support for EPID record: use the ip330 in a fast-feedback loop, 2kHz
- Used in GSECARS for reading ion chambers & photodiodes, used with slow and fast feedback on monochromators
- Good model for A/D converter support, even if you use different hardware
ipUnidig module

- asynDriver support, databases, and MEDM displays for the IPUnidig digital I/O IndustryPack module
- Based on new asynPortDriver C++ base class in asyn
- Interrupt can call any driver, which is useful for devices that do not have interrupts themselves – used in quadEM driver
- Used at GSECARS to trigger detectors, open/close XIA shutters/filter racks, general TTL input and output
- Good model for digital I/O modules, even if you use different hardware
**mca module**

- Support for multichannel analyzers, multichannel scalers, and other array-valued detectors
- mca record
  - Adds needed features not present in EPICS waveform record
- asyn driver support
  - Canberra 556 AIM module (MCA and ICB controller)
  - DSA-2000 Ethernet MCA
  - various Canberra-ICB modules for spectroscopy
  - SIS 3801 (Struck STR7201) and SIS 3820 multichannel scalers for fast pulse counting
  - DXP digital signal processing MCAs
- Fast-sweep support (e.g. waveform digitizer) for ip330 ADC, quadEM beam position monitor, areaDetector ROIs
  - Any detector that can do callbacks with a new value can be put in array
motor module

• Motor record and device support
• Motors are the single most important component of beamline control
• Provides:
  – stepper and servo motors
    • VME, CAMAC, RS-232, GPIB, Ethernet controllers
    • More than 20 types at present
  – soft-motor support
    • Put motor “face” on, e.g., a DAC channel
    • Drive a hard motor through a nonlinear transform of 1 or more real motors
  – user/dial/raw coordinates
  – backlash-takeout algorithm
  – pre/post move commands
  – many more features
motor module

3 types of device/driver support

1. Old, pre-asyn.
   - Separate device and driver support for each controller type.
   - Very hard to add support for features beyond those supported by motor record

2. Newer, asyn-based.
   - Device independent device support, device independent asyn driver.
   - Need to write an asyn-independent driver for a new controller
   - Much better than 1) above, but still hard to add support for controller-specific features

3. Newest: asynMotorDriver C++ class, derived from asynPortDriver
   - Easy to take advantage of controller-specific features
   - Only simulation driver at present
   - I plan to convert Newport XPS and OMS MAXv
   - Can then add complex coordinated motion in a standard manner
   - This is what NSLS-II should use for any new motor drivers
Coordinated multi-axis on-the-fly scanning with EPICS

- Newport XPS (and older MM4005) motor controllers support complex coordinated motion in 8-axis space
- Put out synchronization pulses at specified intervals along the trajectory
  - Used to trigger detectors (e.g. SIS multi-channel scaler, Pilatus, XIA xMAP, etc.) for data collection
- Can read back from the controller the actual positions of every encoder when each pulse was output so actual motor positions are known even if these is a following error.
Use with Large Newport Diffractometers

- 5 installed at APS (GSECARS (2), sectors 2, 33, 34)
- Large goniometers have long setting times (>0.5 sec) for short moves
- On-axis encoders and XPS permit rapid on-the-fly scanning with <0.001 degree position errors
- Many other stages at GSECARS use XPS controllers now, beginning to use XPS to drive x-ray fluorescence maps and other applications.
EPICS Trajectory Scanning Software

- Common database for any controller
- Define the total number of trajectory elements.
  - Controller interpolates between trajectory elements. Simple linear motion thus requires only 2 trajectory elements, no matter how many “points” are being measured.
- Define the absolute or relative position of each axis for each point in the trajectory.
- Define the time for each element of the trajectory, or alternatively the total execution time with equal time per trajectory element.
- Define the total number of output synchronization pulses
- Define the trajectory elements where pulse outputs begin and end.
- Build the trajectory, checking for completion and errors.
- Execute the trajectory, checking for completion and errors. This can be done repeatedly without rebuilding if the only changes are in the start position.
- Read back the actual position of each axis when each synchronization pulse was output.
- All documented at
  - http://cars.uchicago.edu/software/trajectoryScan.html
medm screen
SPEC, IDL or Python client defines actual trajectory
EPICS Implementation

• A database file, `trajectoryScan.db`.
  - Contains almost no “logic” with no links between records in the database. The records are simply variables which channel access clients and the State Notation Language (SNL) program use. Waveform records define motor positions in trajectory.

• SNL program, `MM4005_trajectoryScan.st`, `XPS_trajectoryScan.st`.
  - Implements all of the logic for communicating with the MM4005 and XPS, and with channel access clients via the database.

• MEDM screens, `trajectoryScan.adl`, `trajectoryScanDebug.adl`, `trajectoryPlot.adl`.
  - Used to control the building, execution, readback, debugging and plotting of trajectory scans.

• Goes “behind the back” of the motor record driver to talk directly to the controller.
  - Not the right way to do it, it should be a function that motor controllers can optionally support
  - Need to define the API for coordinated motion
SPEC Interface

• SPEC macros have been written to allow SPEC to utilize trajectory scanning via EPICS interface
• Low level interface, all of SPEC's standard scans can be done "on-the-fly" with trajectory scanning software. Replacement macros for:
  – _ascan # Used by all ascan and dscan macros
  – Mesh
  – hklscan # Used by hscan, kscan and lscan
  – _hklemesh
  – _hklline # Used by hkcircle, hlcircle, klcircle, hkradial, hlradiial and klradiial
  – _scanabort resume
  – _loop
• Enormous improvement in performance, while keeping complete compatibility with SPEC step-scanning
• Can do a 1000-point alignment scan with a point detector or a Pilatus in under 20 seconds. Compare to many minutes step-scanning
Simple example – 2 motors, sinusoidal motion

Actual motor positions read back from controller for each output trigger pulse

Position errors (theory – actual) for each axis
Performance Example with Pilatus driver

• SPEC used to collect 1000 points using trajectory scanning mode with the Newport XPS motor controller. Hardware trigger of Pilatus from XPS.
• Relative scan of the chi axis from -2 degrees to +2 degrees with 1000 points at .02 seconds/point
• Coordinated motion of the phi, kappa and omega axes.
• Theoretical time 20.0 second, actual time 20.8 seconds
• Includes time to save all 1000 images to disk (366 MB), Pilatus driver to read each file, correct bad pixels and flat field, compute ROIs, and post the ROIs and 1000 images to EPICS.
First Results with xMAP MCA Mapping Mode
Matt Newville, 13-ID-C

- SII quad Vortex detector
- Sample stage driven with Newport XPS motor controller running trajectory scanning software, continuous stage motion
- Bi-directional stage motion
- XPS puts out a trigger pulse at each pixel
- XPS captures actual stage position when each trigger pulse is output
- Trigger pulse goes to channel advance on SIS multichannel scaler to capture \( I_0 \) from ion chamber & V/F converter
- SIS output pulse triggers xMAP trigger input
- Current version of software collects 1 row of image in xMAP buffer and writes to netCDF file
  - Could do an entire image into a single file to lower overhead.
  - Need to see if another process can read the file for display update
- Python software reads file, converts to an older format that can be displayed by Matt’s Python collection software.
  - Adds additional overhead, but will be replaced with a new system Matt is designing
XRF Fast Mapping Mode example 1

Maps of XRF intensity in sediment sampled near zinc smelter.

Data collection: 201 x 801 pixels (pixel: 5μm x 5μm) collected at 25ms per pixel

Time per Row = 5.025sec collection + ~2 sec overhead per line
Total Time = 1:37:10  (would be 1:13:47 if done as 801 x 201!!)

At 0.5sec per pixel (previous max rate), total collection time would be 22:21:41

GeoSoilEnviroCARS

G. Morin, F. Juillot  Univ Paris VI

GeoSoilEnviroCARS
Future Work

• Change architecture from SNL program to driver function
  – Make motor record and trajectory scanning work with less possibility of interference
  – Need to define an API that will work on a large number of controllers

• Support other controllers
  – Delta Tau PMAC
  – Galil
  – OMS MAXv
  – …
optics module

• Slits and mirrors
  – Four virtual positioners; two real motors
  – Automatic sync to motor positions
  – Completion reporting

• Monochromators
  – Nondispersive double-crystal
    • Geometries: (Y1, Z2), (Y2, Z2)
    • Crystal species: Si, Ge, Diamond, Si (77K)
    • Miller indices, allowed reflections
    • Operational modes:
      – Use/Set
      – Manual/Auto
    • Managing the vertical beam offset
    • Automatic sync to motor positions
  – Spherical grating
…optics module

- **Optical table**
  - *Table* record supports a six-degree-of-freedom optical table.
  - User/client can write either to \((x, y, z, \theta_X, \theta_Y, \theta_Z)\), or to underlying motor records.
  - Table rotates about user-specified point.
  - Table database includes a list of rotation points, selected by menu.
  - Can recover table position from motor positions
  - Partial support for fewer than six degrees of freedom

- **Calibration/sync**
  - Use/Set – changes to \([X, Y, ..]\). move table / change calibration
  - Zero – redefine current \([X, Y, …]\) as zero
  - Sync – update \([X, Y, …]\) from motors, honoring calibration
  - Init – clear calibration and sync to motors

- Table record sets motor speeds so that motors start/stop together.

GeoSoilEnviroCARS
NSLS-II Data Acquisition and User Interfaces Workshop
April 19, 2010
...optics module

- Optical table
  - Geometries
    - SRI
    - GeoCARS
    - Newport
    - PNC

- Calibration/sync
  - Use/Set – changes to [X, Y, ..]. move table / change calibration
  - Zero – redefine current [X, Y, …] as zero
  - Sync – update [X, Y, …] from motors, honoring calibration
  - Init – clear calibration and sync to motors

- Table record sets motor speeds so that motors start/stop together.
sscan module

• Support for user-programmable data-acquisition
  – sscan and busy records
  – saveData
  – recDynLink

• A one-dimensional scan:
  – Do NPTS times:
    • Set conditions e.g., move motors; wait for completion
    • Trigger detector e.g., start scaler; wait for completion
    • Acquire data read detector signals; store in arrays
  – Write data to NFS file

• Multidimensional scan:
  – Same as a 1-D, but detector trigger executes inner-loop scan.
  – saveData monitors a set of sscan records, determines scan dimension when scan starts, and writes data as it is acquired.
sscan module

GeoSoilEnviroCARS

NSLS-II Data Acquisition and User Interfaces Workshop
April 19, 2010
...sscan module

- scan features:
  - Three 1-D scan types: constant-step-size, table-driven, fly
  - Unlimited number of data points, scan dimensions
  - 0-4 positioners, 0-4 detector triggers, 0-70 detector signals
  - Acquisition from scalar and 1-D-array-valued PV's
  - Detector/client wait, data-storage wait
  - Pause/resume, abort
  - Double buffered: can write 1-D acquired data during next 1-D scan
  - `saveData` writes self-describing XDR-format (".mda") files to NFS-mounted disk (vxWorks only, at present).
  - A positioner can have private scan parameters (scanparm record).
  - After-scan actions include move to peak, valley, and edge.
  - scanparm record + after-scan action = automated 1-D alignment, so you can easily implement an “Align” button.
std module

• EPID record
  – Enhanced PID record for feedback
  – Can use soft device support with standard EPICS links for readback and output
  – Also has asyn fast device support, gets callbacks from an input and writes directly to
    an output without going through record support. > 2kHz no problem.

• Scaler record
  – Controls a set of counters with a common clock, gate, and trigger
  – Now has asyn device-independent device support

• String-sequence record
  – Like the seq record in base, but works for strings and numbers
  – Can choose to wait for completion after each step in sequence
**vme module**

- Device support for VME hardware, includes:
  - Joerger scaler
  - APS bunch-clock generator
  - APS machine-status interface
  - Heidenhain encoder interpolator
  - Generic A32 VME interface
  - HP Laser interferometer
  - VMI4116 16-bit DAC
  - Acromag 9440 16-bit digital input

- VME record
  - Provides run-time access to VME bus
  - Great for testing hardware
  - Run-time programmed control of an unsupported VME board
**xxx module**

- **Prototype user directory**
  - Builds everything in synApps into a load module
  - Contains command files to load/configure everything in synApps
  - Contains sample top-level MEDM-display file
  - Contains sample script to set environment variables and start up the sample user interface
  - Contains table of recommended address/interrupt configuration for selected VME and IndustryPack hardware

- **Two ways to use this module**
  1) Make copies; run changePrefix; build; customize; run a beamline
     - this is the recommended use
     - detailed instructions in support/documentation
  2) Reference/grab bag
User Interfaces at GSECARS

• SPEC
  – Used to control 2 large Newport diffractometers
  – Only software we have that understands reciprocal space and the geometry of the diffractometers
  – It is very familiar to the scattering users
  – But they understand its limitations as a programming language

• Tomography
  – IDL GUI that talks directly to CCD detector and to EPICS motors, etc.
  – Will replace CCD interface with areaDetector
  – Simple, easy for users to learn
  – IDL program acts as a server via EPICS PVs, so scripts on other computers can order it to collect a data set while they change energy, temperature, etc.

• XRF microprobe and EXAFS
  – Python GUI and data collection code by Matt Newville
  – No unifying user interface beyond medm

• What we’d like:
  – A common GUI and common scripting language that can do all of the above and more
  – GDA?
Some Advice

• Put the functionality in the server, resist the temptation to put it in the client if possible
  – Then any client can use it, not just your favorite client *du jour*

• Use asyn to write drivers

• Purchase devices that already have support if they will meet the need. Resist temptation to save a few $$ on hardware if it means debugging a new device.
  – Check on tech-talk@aps.anl.gov to see if the work has already been done
Conclusions

• NSLS-II has clearly recognized the importance of beamline controls and data acquisition software early

• Assembled a great team of developers!

• Looking forward to working together to create a common set of tools and user interface to make life easier for developers and for users, many of whom will do experiments at both APS and NSLS-II

• Thanks for your attention!