# Getting Started with EPICS
## *Applications / Special Topics*

## Introduction to synApps (v5.1)

*Tim Mooney*
*1/11/2005*

***Argonne National Laboratory***

*A U.S. Department of Energy*
*Office of Science Laboratory*
*Operated by The University of Chicago*

## What is synApps?

- **A collection of EPICS applications for synchrotron-beamline users**
  **http://www.aps.anl.gov/aod/bcda/synApps**
- **EPICS modules and build/configuration tools:**
  - Modules: *autosave, calc, camac, ccd, dac128V, dxp, ip, ip330, ipUnidig, love, mca, motor, optics, quadEM, sscan, std, vme, xxx*
  - Build/config*: config and utils directories*
- **Related clients, libraries, and visualization tools:**
  - IDL: scanSee, mca display, ezcaIDL, ezcaScan, ez_fit, HDF translator/browser, Ascii-format plotter, image processors, etc.
  - CA-Server based CCD control
  - some python support

## synApps modules

- **Modules contain the following kinds of support:**
  - Compiled code; libraries
    - *E.g., record and device 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 4.1**
- **ipac 2.8**
- **seq 2.0.8 (9?)**
- **genSub 1.6**
- **vxStats 1.7.2c**
- **allenBradley 2.1**

## 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 (synApps 5.1)**
  - Array-valued PV must be hosted by the ioc that does the restore operation. (Typically, all ioc's save/restore their own PV's.)
  - DBF_MENU, DBF_ENUM PV's are handled by number.
- **Save operation uses channel access for scalars.**
- **Restore operation uses static database access for scalars.**
- **Arrays are saved and restored with database access.**

## …autosave module

- **Three restore options for save files:**
  1) before record/device initialization
     - Motor positions must be restored at this time.
     - Arrays cannot be restored at this time. *
     - PV's that are DBF_NOACCESS before record init (e.g., genSub variable-type fields) cannot be restored at this time. *
  2) after record/device initialization
     - to override record-initialization values
     - Link fields cannot be restored at this time. *
  3) both before and after record initialization
     - The 'auto_settings.sav' file is restored at both times.
     - It's not an error to attempt to restore a PV at the wrong time.
     - If you restore a motor position at this time, you override the value read from hardware, without writing to hardware.

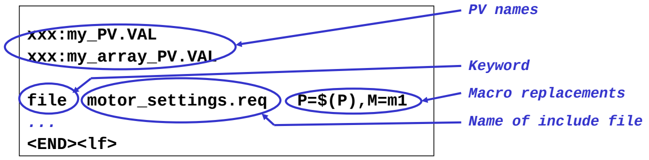
* Not illegal, just doesn't work

## ...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.**
- **Autosave can recover from file-server reboot (synApps 4.6+).**
  - Currently, only on vxWorks
- **User can choose to save redundant files (synApps 4.6+).**
- **Autosave reports status via EPICS PV's (synApps 5.1+).**

## ...autosave module

- **Sample request file**



```
xxx:my_PV.VAL
xxx:my_array_PV.VAL

file motor_settings.req P=$(P),M=m1
...
<END><lf>
```

- **Sample save file**

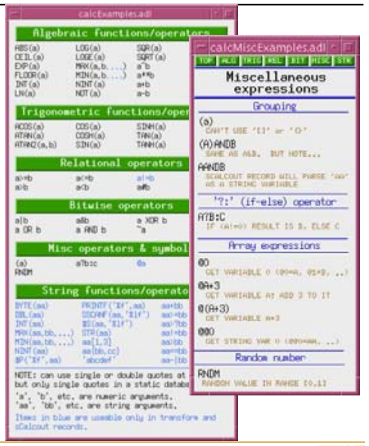
```
# save/restore V4.4 Automatically generated…
xxx:my_PV.VAL 1.0
xxx:my_array_PV.VAL @array@ { “0” “0.1” … “10.2” }
xxx:m1.DIR 0
xxx:m1.DHLM 100
xxx:m1.DLLM –100
...
```

# 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

# ...calc module

- **Databases, medm displays for run-time programming**
  - userCalc
  - userStringCalc
  - userTransform
  - userAve
  - arrayTest
  - interpolation
- **Examples of ALL calc expressions (normal and stringCalc) can be found in synApps MEDM help displays**



```
calcExamples.adl

Algebraic functions/operators
ABS(a)          LOG(a)          SQR(a)
CEIL(a)         LOGE(a)         SQRT(a)
EXP(a)          MAX(a,b,...)    a^b
FLOOR(a)        MIN(a,b,...)    a**b
INT(a)          NINT(a)         a+b
LN(a)           NOT(a)          a-b

Trigonometric functions/oper
ACOS(a)         COS(a)          SINH(a)
ATAN(a)         COSH(a)         TAN(a)
ATAN2(a,b)      SIN(a)          TANH(a)

Relational operators
a>=b            a<=b            a!=b
a>b             a<b             a#b

Bitwise operators
a|b             a&b             a XOR b
a OR b          a AND b         ~a

Misc operators & symbol
(a)             a?b:c           @a
RNDM

String functions/operato
BYTE(aa)        PRINTF('%f',aa)     aa+bb
DBL(aa)         SSCANF(aa,'%[f')    aa>=bb
INT(aa)         $S(aa,'%lf')        aa>?bb
MAX(aa,bb,...)  STR(aa)             aa!=bb
MIN(aa,bb,...)  aa[1,3]             aa<bb
NINT(aa)        aa{bb,cc}           aa==bb
$P('%f',aa)     'abcdef'            aa-|bb

NOTE: can use single or double quotes at
but only single quotes in a static databa
'a', 'b', etc. are numeric arguments.
'aa', 'bb', etc. are string arguments.
Items in blue are usable only in transform and
sCalcout records.
```

```
calcMiscExamples.adl
TOP  ALG  TRIG  REL  BIT  MISC  STR

Miscellaneous
expressions

Grouping
(a)
  CAN'T USE '[]' or '{}'
(A)ANDB
  SAME AS A&B.  BUT NOTE...
AANDB
  SCALCOUT RECORD WILL PARSE 'AA'
  AS A STRING VARIABLE

'?:' (if-else) operator
A?B:C
  IF (A!=0) RESULT IS B, ELSE C

Array expressions
@0
  GET VARIABLE 0 (@0=A, @1=B, ..)
@A+3
  GET VARIABLE A; ADD 3 TO IT
@(A+3)
  GET VARIABLE A+3
@@0
  GET STRING VAR 0 (@@0=AA, ...)

Random number
RNDM
  RANDOM VALUE IN RANGE [0,1]
```

# camac module

- **Communication with CAMAC crate/modules**
- **Records**
  - camac – generic BCNAF/data for run-time camac control
- **Devices supported**
  - VME bus adapter
  - CAMAC crate controller
  - E500 motor controller
  - RTC-018 real-time clock
  - QS-450 quad scaler
  - DXP spectroscopy system (now in *dxp* module)

# ccd module

- **Support for area detectors (CCD’s and image plates)**
- **Supported devices**
  - MAR 165 CCD
  - MAR 345 image-plate reader
  - Roper (all WinView-supported CCD’s, including former Princeton and most former Photometrics devices)
  - Bruker SMART CCD
- **Can control, at minimum**
  - exposure time
  - file name
  - data-acquisition start
  - wait for acquisition to complete
  - much more for most devices
- **See lecture “*Detectors and Feedback.*”**

## config directory

- **Configures and builds all modules in or used by synApps**
- **MASTER_RELEASE**
  - specifies version number and file path to EPICS base, and to every module in or used by synApps
- **makeReleaseConsistent.pl**
  - Edits <module>/configure/RELEASE for every module in or used by synApps, to agree with MASTER_RELEASE
  - “gnumake release” causes this to run.
- **Makefile**
  - “gnumake <whatever>”, in config directory, does <whatever> for all modules.

## dac128V module

- **device support, database, and MEDM displays for dac128V IndustryPack module**
  - 8-channel, 12-bit DAC
  - Support exists to run a DAC channel manually, or according to an algorithm written at run time, or as a *scan* positioner, or as part of a PID feedback loop.
- **See lecture “*Detectors and Feedback.*”**

## documentation directory

- ***TOP-level synApps documentation***
  - What synApps is
  - How to build it
  - How to make a user application from the ‘xxx’ sample module
  - How to fit the user application to a particular set of hardware
- **This presentation**

## dxp module

- **record, device support, databases, and MEDM displays for XIA DXP and Saturn spectroscopy systems**
- **dxp record for setting DXP parameters**
- **device support for the mca record**
- **See lecture “*Detectors and Feedback.*”**

## ip module

- **device support, SNL code, databases, and MEDM displays for *many* message-based devices**
  - originally, for devices supported via IndustryPack hardware
  - Note some of this support will inevitably be out of date -- pending access to hardware for testing.
- **deviceCmdReply (was serial_OI_block, GPIB_OI_block)**
  - Used to write support at run time for one command/reply message
  - sCalcout to format output string
  - asyn record to write/read device
  - sCalcout record to parse reply
- **devXxStrParm device support**
  - probably will be replaced by streams/asyn

## ip330 module

- **device support, databases, and MEDM displays for the IP330 ADC IndustryPack module**
- **16/32 channel, 16-bit ADC**
  - ip330Scan for periodic, averaged reads of ADC channels
  - ip330Sweep, with the MCA record, for using ip330 as a waveform-digitizer
  - ip330PID for using the ip330 in a fast-feedback loop
- **See lecture “*Detectors and Feedback.*”**

## ipUnidig module

- **device support, databases, and MEDM displays for the IPUnidig digital I/O IndustryPack module**
- **IP-UD-I 24-channel input/output/interrupt module**
- **DIO316I 48-bit digital I/O module**
- **See lecture “*Detectors and Feedback.*”**

## love module

- **Support for Love controllers**
  - orphaned, currently under re-development for EPICS 3.14

# mca module

- **Support for multichannel analyzers, multichannel scalers, and other array-valued detectors**
- **mca record**
- **device support**
  - Canberra 556 AIM module (MCA and ICB controller)
  - DSA-2000 Ethernet MCA
  - various Canberra-ICB modules for spectroscopy
  - SIS 3801 (Struck STR7201) MCS
  - (DXP support in dxp module)
  - (IP330 support in ip330 module)
  - (quadEM support in quadEM module)

- **See lecture "*Detectors and Feedback.*"**

# motor module

- **Motor record and device support**
  - stepper and servo motors
  - soft-motor support
    - *Put motor "face" on, e.g., a DAC channel*
    - *Drive a hard motor through a nonlinear transform*
  - user/dial/raw coordinates
  - backlash-takeout algorithm
  - pre/post move commands
  - many more features

- **See lecture "*Motors.*"**

# 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

# …optics module

- ***…Monochromators***
  - *Spherical grating*
    - Geometrical variables:
      - *1) Grating line density; radius*
      - *2) Tangent-arm length*
      - *3) Diffraction order*
      - *4) Input/output slit distances*
    - Operational modes:
      - *Use/Set*
      - *Manual/Auto*
    - Grating-stripe list
    - Manual sync to motor positions

## …optics module

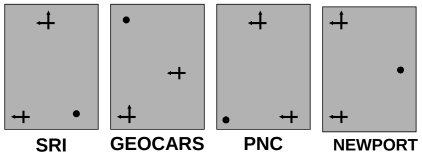
- ***…Monochromators***
  - *Dispersive double-crystal*
    - Geometries: nested, symmetric
    - Crystal species: Si, Ge, Diamond, Si (77K)
    - Miller indices, allowed reflections
    - Operational modes:
      - *Use/Set*
      - *Manual/Auto*
      - *Theta1 / Theta1&2 / Rock Theta2*
    - Accommodate incident-beam angle shift (“world offset”)
    - Automatic sync to motor positions

## …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

## …optics module

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



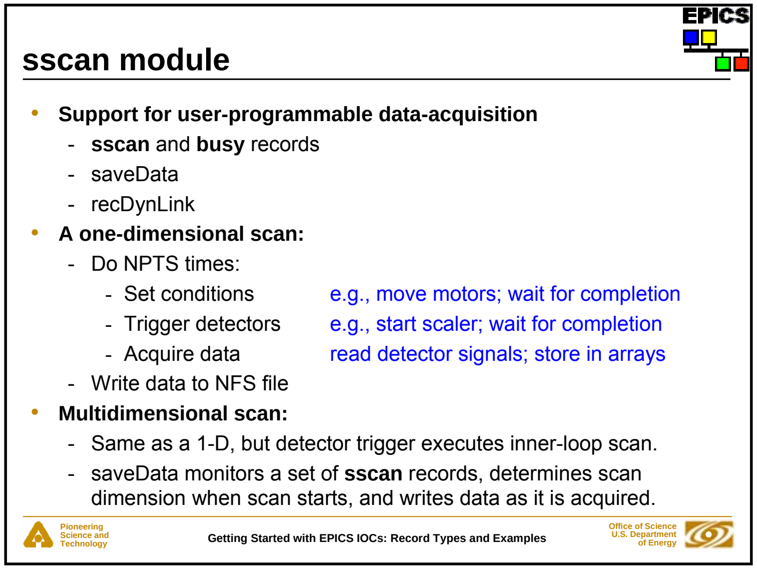
- Calibration/sync
  - Use/Set – changes to [X, Y, ..]. move table / change calibration
  - Zero – redefine current [X, Y, …] as zero
  - Sync – update [X, Y, …] from motors, honoring calibration
  - Init – clear calibration and sync to motors

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

## quadEM module

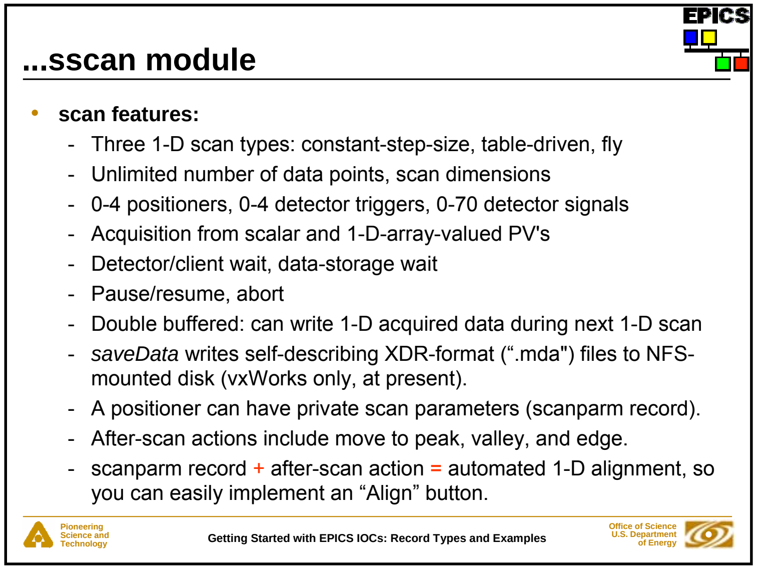
- **Support for APS Detector Group’s (Steve Ross) four-input electrometer.**

- **See lecture  “*Detectors and Feedback.*”**



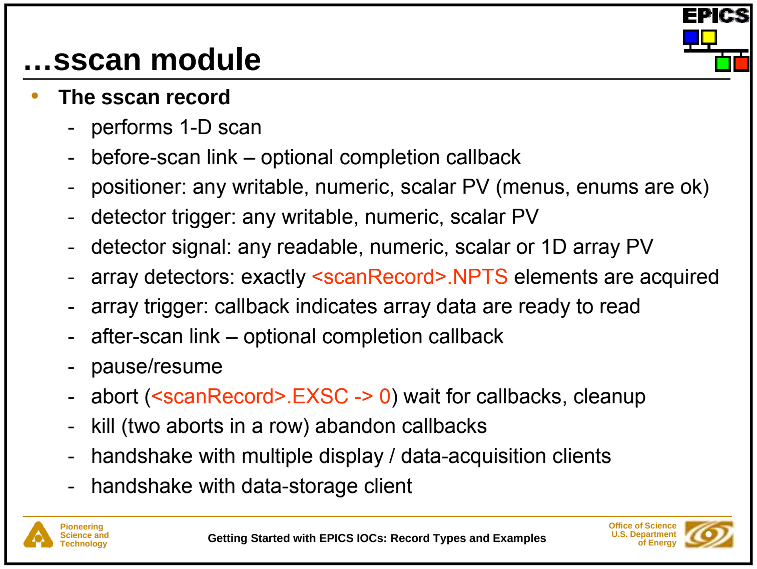
## sscan module

- **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 detectors    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.



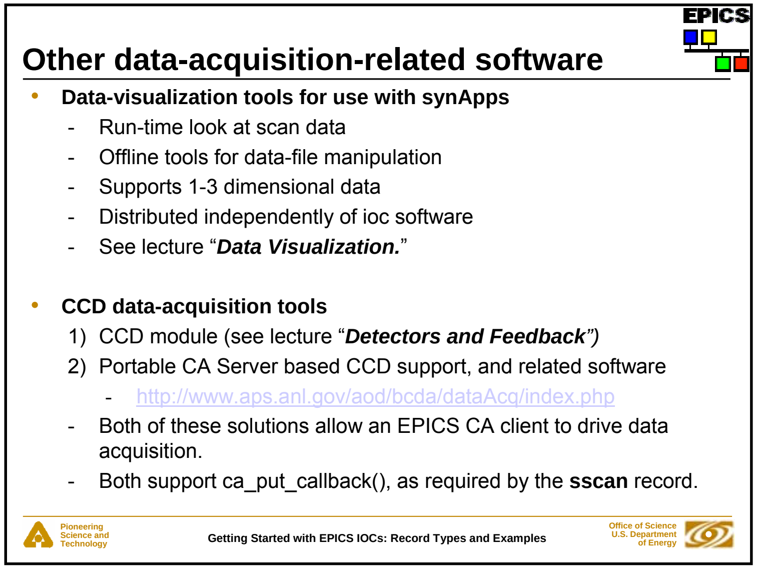
## ...sscan module

- **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.



## …sscan module

- **The sscan record**
  - performs 1-D scan
  - before-scan link – optional completion callback
  - positioner: any writable, numeric, scalar PV (menus, enums are ok)
  - detector trigger: any writable, numeric, scalar PV
  - detector signal: any readable, numeric, scalar or 1D array PV
  - array detectors: exactly <scanRecord>.NPTS elements are acquired
  - array trigger: callback indicates array data are ready to read
  - after-scan link – optional completion callback
  - pause/resume
  - abort (<scanRecord>.EXSC -> 0) wait for callbacks, cleanup
  - kill (two aborts in a row) abandon callbacks
  - handshake with multiple display / data-acquisition clients
  - handshake with data-storage client



## Other data-acquisition-related software

- **Data-visualization tools for use with synApps**
  - Run-time look at scan data
  - Offline tools for data-file manipulation
  - Supports 1-3 dimensional data
  - Distributed independently of ioc software
  - See lecture "***Data Visualization.***"
- **CCD data-acquisition tools**
  1) CCD module (see lecture "***Detectors and Feedback***")
  2) Portable CA Server based CCD support, and related software
     - http://www.aps.anl.gov/aod/bcda/dataAcq/index.php
  - Both of these solutions allow an EPICS CA client to drive data acquisition.
  - Both support ca_put_callback(), as required by the **sscan** record.

## std module

- **Epid record**
  - Extended PID record – see "***Detectors and feedback***" lecture
- **Scaler record**
  - Controls a set of counters with a common clock, gate, and trigger
- **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
- **Soft-motor database (Jonathan Lang)**
  - *Run-time programmable* soft-motor/transform/hard-motor database
  - Quick solution for driving a motor through a nonlinear transform
- **Timestamp record [SLAC]**
  - needed by SNS' vxStats; currently not available in a module
- **4-step database**
  - Up to four steps of (set condition; read data) with an end calculation
  - Originally developed for dichroism experiments

## utils directory

- **changePrefix**
  - Global search and replace of EPICS PV prefix within a copy of the xxx module
- **copyAdl**
  - Find all MEDM-display files buried in a file tree; copy to specified directory.

## vme module

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

## 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

## For developers: features of synApps

- **extended-processing records**
  - records that are neither synchronous nor asynchronous, as these terms are described in the *EPICS Application Developer's Guide*
- **completion reporting**
  - All databases behave correctly when written to by ca_put_callback().
- **recDynLink links**
  - Similar to standard EPICS links, but no "PP NMS" attributes
- **GUI standards**
  - Default colors for menus, PV values, links, etc.
- **coordinated motions**
  - Many of the databases in synApps (especially in 'optics') involve coordinated motion of several motors.
- **initialization of complex databases**
  - Some common EPICS initialization problems are handled in various synApps databases.

## Coordinated motions

- **Simple cases: database (transform records)**
  - Slits, mirrors, spherical-grating monochromator
- **More complicated cases: SNL code**
  - Multiple-crystal monochromators
- **Very complicated cases: custom record**
  - Optical table, scan
- **Criteria a useful coordination should meet:**
  - Report completion to ca_put_callback()
  - Share control of base positioners with CA clients
  - Recover state from the states of base positioners

## Completion reporting

- **Simple prescription for databases contained within a single ioc:**
  - Use only PP links and forward links in execution chain.
- **Database operations spanning more than one ioc:**
  - Use records with put_callback links to span iocs:
    - ***calcout*** *with asynchronous device support*
    - ***sscan*, *swait***
    - ***sseq*** *or* ***sCalcout*** *(with .WAIT* = "Wait")*
- **Cases in which a CA client performs part of the operation:**
  1) Database sets a **busy** record via PP or put_callback link.
  2) CA client clears the **busy** record when operation is done.
- **Cases in which part of the operation is driven by a CP link:**
  - Not different from above; a CP link is a CA client

## Initialization of complex databases

- **Initial values: .VAL vs. .DOL**
  - Most records allow .VAL field to be set in the database.
  - Note that .DOL cannot be used for constant strings.
- **Save-restore and interaction with record/device initialization**
  1) save-restore pass 0
  2) record/device initialization → *device support can use pass-0 value*
  3) save-restore pass 1 → *pass-1 overrides record/device-init value*
- **.PINI (Process at INIt) uses and limitations**
  - This is the normal mechanism for database initialization.
  - What if you need a value from some other .PINI-initialized record, and that record hasn't processed yet?
  - Note .PHAS is not considered in .PINI processing.

# ...Initialization of complex databases

- **Contending with link alarms**
  - If you have an input link to a record with .UDF=1, you get a link alarm.
  - .UDF=1 until a record processes. (In 3.14.1+, database can specify .UDF)
  - ➢ The transform record can abort execution on a link alarm (or not).
- **Initialization problems with CP links**
  - You have a CP link to a field that is a calculation result.
  - If the calc result is the same as the field's initial value, you'll have the right value, but you won't *know* that you have the right value, and you won't know for how long to wait to be sure.
  - ➢ The transform record *always* posts its initial calculation result.
- **Programmatically initializing link fields**
  - Link field must be written with a CA link (because lock-set recalc).
  - .PINI processing occurs *before* CA is running (EPICS 3.13.5+).
  - ➢ Can't use .PINI; Drive init from a scan task; set init record to "Passive" when init is done.