




Finding, Deploying and Managing EPICS I/O Support

Andrew Johnson, ASD/Controls
11th November 2004

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

Advanced Photon Source

Outline

- ◆ Meaning of I/O Support
- ◆ Review of IOC development environment
- ◆ Development areas as components
- ◆ Distinguishing Support Modules and IOC Applications
- ◆ Managing change in Support and IOC top areas
- ◆ Different ways to support I/O devices
- ◆ Finding existing software
- ◆ Deploying support modules
 - ❖ Installing software
 - ❖ Using it in an IOC

Meaning of "I/O support"

- ◆ Software that connects one or more record types to some real hardware
 - ❖ At minimum contains device support code
 - ❖ May include record or driver support as well
 - ❖ Also covers any additional software needed by those layers
 - ✦ *Not necessarily EPICS-specific code*
- ◆ May comprise multiple separate software packages, e.g.:
 - ❖ drvIpac Octal232 serial port support
 - ❖ Asyn Serial interfacing software
 - ❖ *Device Support* Specific commands for the target serial device
- ◆ Might provide database templates, SNL programs and MEDM screens
 - ❖ SynApps does this for beamline devices
 - ❖ Not appropriate for all kinds of I/O

IOC Development Environment Review

```
<top>
  configure
    RELEASE
    CONFIG
    Other build-system files
  xxxApp
  src
    C & SNL sources, .dbd files
  Db
    .db files, templates & substitutions
  locBoot
    locxxxx
    st.cmd
  Installation directories:
    dbd, db, include, bin/<arch>, lib/<arch>
```

Development Areas as Components



- ◆ The <top> structure and Makefile rules are designed to encourage modularity
 - ❖ An IOC is built up out of many components
 - ❖ Channel Access, database access, scanning, other core libraries
 - ❖ Record, device & driver support
 - ❖ Databases
 - ❖ Sequence programs
 - ❖ Etc.
 - ❖ These components do not have to be defined in the same <top> as the IOC itself
 - ❖ Most of the IOC software comes from Base
 - ❖ Other <top> areas can provide additional components
 - ❖ Other <top> areas can override (replace) components from Base
- ◆ The configure/RELEASE file determines what other <top> areas will be searched for required components, and in what order
 - ❖ Only the installation directories of other <top> areas are searched



Support Modules vs. IOC Applications



- ◆ Most sites distinguish between <top> areas that provide commonly-used components, and those that build IOCs
 - ❖ Record, Device and driver support are usually shared by many IOCs
 - ❖ Having multiple copies of source files is a recipe for disaster
 - ❖ Different engineers maintain device support than IOCs
 - ❖ The lifecycles of the two are usually very different
- ◆ Areas that provide components are Support Modules
 - ❖ /usr/local/epics/R3.14.6/modules/...
- ◆ Areas that build IOCs are IOC Applications
 - ❖ /usr/local/epics/R3.14.6/ioc/...
- ◆ IOC areas use entries in configure/RELEASE to indicate which support apps they will use components from
- ◆ IOC areas may contain local record/device/driver support, but only for hardware that is not used in in any other IOC subsystem



Managing Change: Support Modules



- ◆ Support modules usually only change when a bug is fixed, or some new functionality is added
 - ❖ It should be a deliberate decision by the engineer responsible for an IOC subsystem to use a new version of a support module
 - ❖ Bug fixes can introduce new bugs elsewhere
 - ❖ New functionality might include changes that break old applications
- ◆ Therefore: once installed and in use by an operational IOC application, a support module's <top> area should **never** be changed
 - ❖ New versions of the module should be installed alongside the old one
 - ❖ The engineer responsible for an IOC subsystem can switch to the new version when s/he is ready for it by changing the IOC Application's <top>/configure/RELEASE file to point to the new version
 - ❖ The old version is not deleted until it will never be required again
 - ❖ It's easy to revert to a previous version if problems are found
 - ❖ Disk space is cheap
- ◆ Sites should document what changed in each update of a support module, and should notify IOC engineers that a new version is available



Managing Change: IOC Applications



- ◆ IOC applications change very frequently in small ways
 - ❖ Updating alarm limits, revised sequences, adding new I/O points etc.
- ◆ It should be relatively easy to modify the files that configure an IOC (databases, subroutines, sequence programs etc.)
 - ❖ Elaborate version control procedures make it harder to respond to change requests, so less change will happen or the procedures get ignored
- ◆ However it is important to retain the history of what application changes were made, when and by whom
 - ❖ Being able to quickly back out of recent modifications can be essential to recovering from incorrect changes
- ◆ This is very different to the requirements of managing a support application
 - ❖ Do not attempt to use the same approach to manage both



Ways to support I/O devices

- ◆ **Hardware-specific device and/or record support**
 - ❖ Best solution if already available
 - ❖ Recommended if there will be multiple instances
 - ✦ *Communications protocol can be hidden from users and usually from EPICS applications developers*
 - ✦ *Much easier to handle any device peculiarities*
 - ❖ May be essential for complex devices
- ◆ **Generic support**
 - ❖ Available for general-purpose bus, serial, and network connected I/O
 - ❖ Device-specific knowledge is contained in database records
 - ✦ *Special fields in a custom record*
 - ✦ *Parm field of a hardware link address*
 - ❖ Not possible for complex devices or protocols

Finding existing software

- ◆ **Supported Hardware database on EPICS website**
- ◆ **EPICS Collaboration meetings**
- ◆ **Tech-talk mailing list**
 - ❖ Search the archives before asking
- ◆ **Google Search**

- ◆ **Most existing support software is for R3.13.x on vxWorks**
- ◆ **Porting to R3.14 is relatively easy**
- ◆ **Porting to RTEMS is slightly harder**
 - ❖ Should be straight-forward if the software uses devLib
- ◆ **Porting to Linux is inadvisable for VME or PCI-based hardware**
 - ❖ Linux Kernel drivers are hard to write and debug
- ◆ **Serial or network-based support can be made as portable as Base**
 - ❖ The Asyn framework simplifies this process

Deploying EPICS Support modules

- ◆ **Installing the software**
 - ❖ If installation instructions are provided, read & follow them!
 - ✦ *SynApps users should read the synApps README file*
 - ✦ *The synApps configuration system is slightly different*
 - ❖ Prerequisites:
 - ✦ *EPICS Base R3.14.x built for the desired architecture(s)*
 - ✦ *All other support modules needed have already been built*

Deploying: Configure the module

- ◆ **Extract the source file to the required location using tar/unzip/...**
- ◆ **Edit the configure/RELEASE file**
 - ❖ Set the EPICS_BASE variable pointing to the Base directory
`EPICS_BASE=/usr/local/epics/R3.14.6/base`
 - ❖ Set any other required paths in the file as appropriate, e.g.:
`IPAC=/usr/local/epics/R3.14.6/drvIpac-2.7`
`ASYN=/usr/local/epics/R3.14.6/asyn-3.3`
 - ❖ You may define other variables and use the Makefile \$(variable) syntax
`SUPPORT=/usr/local/epics/R3.14.6`
`EPICS_BASE=$(SUPPORT)/base`
`IPAC=$(SUPPORT)/drvIpac-2.7`
`ASYN=$(SUPPORT)/asyn-3.3`
 - ❖ Variables that name other support modules must expand to absolute pathnames beginning with a '/'; relative paths will not work
 - ✦ *'..' is allowed as a component within an absolute path*

Deploying: Build the module

- ◆ Run GNU make in the <top> directory
 - ❖ To simultaneously log the output to a file, use these commands
 - ❖ *CSH,tSH:* `gnumake |& tee build.out`
 - ❖ *sh,ksh,bash:* `gnumake 2>&1 | tee build.out`
 - ❖ This checks the configure/RELEASE file for consistency with the support modules named, and generates files in configure/O.<arch>
 - ❖ Then it descends through the other subdirectories of <top>, compiling and installing products as necessary

Using Support in an IOC Application

- ◆ Mostly very specific to the Support Module
- ◆ General steps are
 - ❖ Add a line to the IOC Application's configure/RELEASE file that points to the new module's <top> directory
 - `ASYN=$(SUPPORT) /asyn-3.3`
 - ❖ Edit the Makefile where the IOC is built
 - ❖ *Add a line for each library to be linked against*
`example_LIBS += asyn`
 - ❖ *Add a line for any .dbd files to be included*
`example_DBD += asyn.dbd`
 - ❖ *Alternatively this might become a line in exampleInclude.dbd:*
`include "asyn.dbd"`
 - ❖ Add any necessary lines to the st.cmd file
 - ❖ Create record instances, instantiate templates, etc.
 - ❖ Rebuild the IOC Application from <top>
`gnumake rebuild`