

# EPICS 'makeBaseApp', IOC Binaries

**Kay Kasemir**

**ORNL/SNS**

[kasemirk@ornl.gov](mailto:kasemirk@ornl.gov)

**Sept. 2014**

# EPICS IOC

- **Ideally: Just Database records**

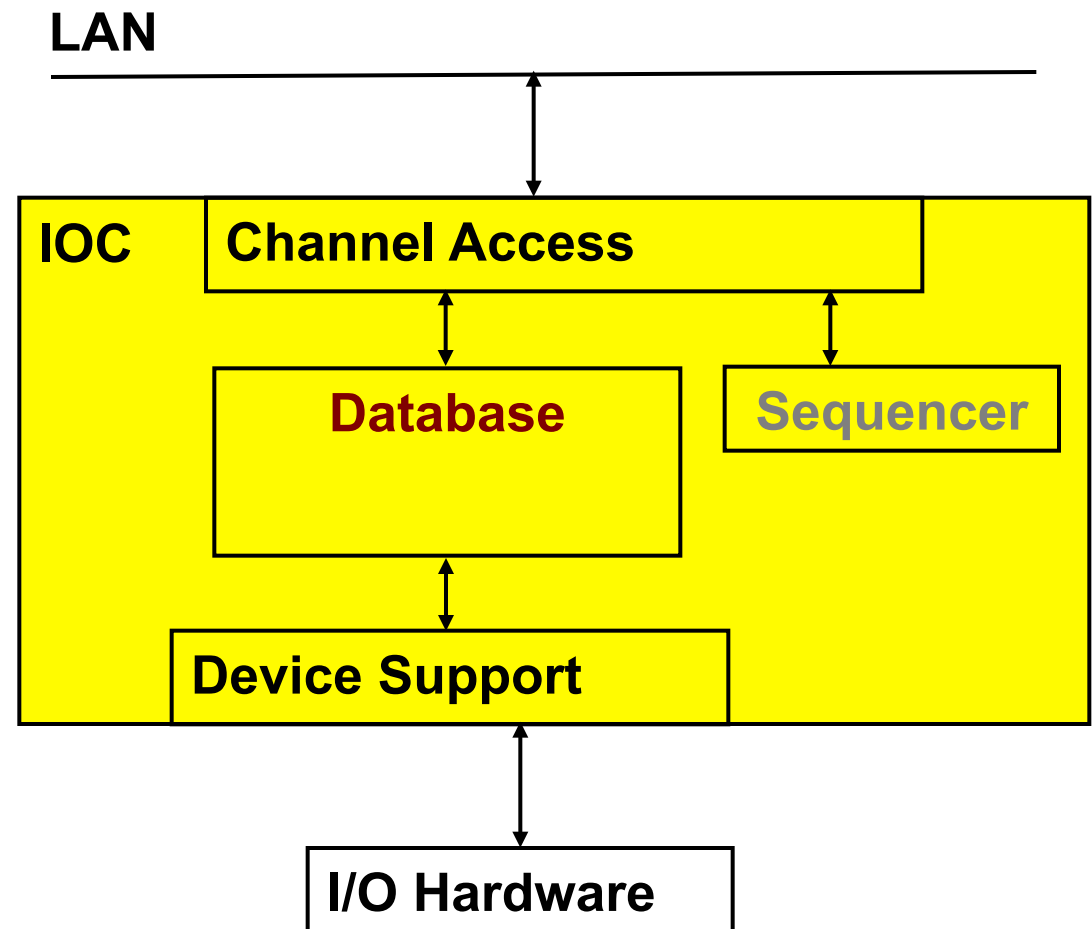
- Known & well tested building blocks
- Remote access
- Access security
- 'bumpless' reboot

- **Sometimes: Need Sequencer code**

- C(++) code, nobody else will understand it

- **Need Device Support**

- Include existing device support? Easy enough
- Have to write new device (driver) code? Running with scissors!

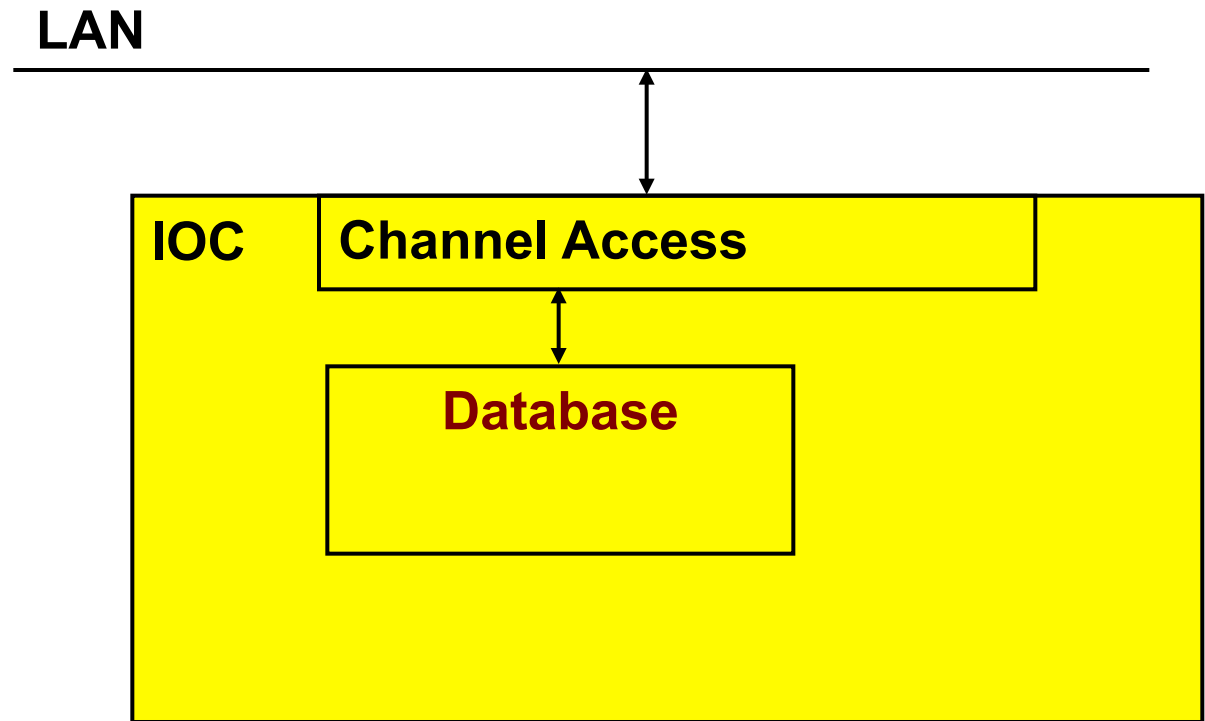


# 'softloc'

Binary with  
Database engine  
and  
Channel Access.

Run as many  
instances as  
needed.

Need sequencer, device support?  
→ Create your own IOC application binary!



# Who needs custom IOC binary?

## Accelerator: One per subsystem

- Vacuum: Support for AllenBradley PLC
  - LLRF: Support for LLRF hardware
- Different maintainers, different needs,  
then many instances per subsystem

## Beamlines: One (few) per beamline?

- CG-1D:  
One binary with support for Camera, Parker6K, 'Stream' device.  
Separate instance for Camera, Motors, ICP, Robofocus, Scan support.
  - Choppers: One binary for all choppers?
- Each beam line must have different binary  
to allow independent updates.  
Within a beam line, try to keep low number?

# 'makeBaseApp.pl'

- **Creates skeleton for custom IOC**
  - Directory structure
  - Makefiles
  - Examples: \*.db, \*.st, driver/device/record \*.c
  - IOC startup file
- **Two extremes**
  - **makeBaseApp.pl –t example**
    - Get most everything; you delete what's not needed
  - **makeBaseApp.pl –t ioc**
    - Just dirs & Makefiles; you add what's needed

# EPICS Build Facility

## Is outstanding

- **make, perl**
- **Builds on Linux, Mac, Windows**
- **..for Linux, FreeBSD, OS X, Windows, vxWorks, RTEMS, x86, x86\_64, ppc, arm, ...**
- **AppDevGuide**
- **Functioned for decades across many changes of OSs, compilers, ...**

## Is aggravating

- **“Why is it not an Eclipse, Visual C++, Kdeveloper ... project? What about CMake, GNU automake, ... ?”**
- **What’s the name of that option again?**
- **What’s causing this error now?**

# 'example' Example

```
# Go somewhere
cd ~/EPICSTraining

# Create IOC application of type 'example',
# using 'demo' in the generated names
makeBaseApp.pl -t example demo

# Create IOC startup settings of type 'example',
# call it 'demo'
makeBaseApp.pl -t example -i demo
# When prompted, use the previously created 'demo'
# application as the one that the IOC should load

# Compile everything
make

# Start IOC
cd iocBoot/iocdemo
chmod +x st.cmd
./st.cmd
```

# Directory Layout: Key Files

```
# makeBaseApp.pl -t example demo
```

```
configure/RELEASE
```

```
configure/CONFIG_SITE
```

```
demoApp/Db/*.db
```

```
demoApp/Db/*.substitutions
```

```
demoApp/Db/Makefile
```

```
demoApp/src/Makefile
```

```
# makeBaseApp.pl -t example -i demo
```

```
iocBoot/iocdemo/Makefile
```

```
iocBoot/iocdemo/st.cmd
```

- To study the skeleton, check files before the first 'make' or after a 'make distclean'



# Directory Layout: Generated Files

```
**/O.Common  
**/O.linux-x86_64  
**/O.*  
db/*  
dbd/*  
include/*  
lib/*  
bin/*
```

## Beware of difference:

- **whateverApp/Db/\***
  - Database ‘Sources’. **Edit these!**
- **db/\***
  - ‘Installed’ databases, may have macros replaced.  
**Will be overwritten** by next ‘make’!

# **\*.dbd: Database Descriptions**

**IOC record types, device support, ... are extensible**

- **Implement new record type, new device support:  
Write C/C++ code for certain interfaces, compile.**
- **Somehow ‘register’ this with core IOC code:  
\*.dbd file**

**Internals:**

**VxWorks RTOS, the original IOC target, had  
runtime loader and symbol table.**

**RTEMS, .. don’t necessarily offer this.**

**EPICS build facility generates IOC startup source  
code from \*.dbd file.**

# HowTo: Add Support Modules (Device, ...)

Example: 'Autosave'

## 1. Define path in configure/RELEASE:

```
AUTOSAVE=/home/controls/epics/R3.14.12.2/support/autosave
```

Path to the support directory is usually pulled into a macro, since you often include more than one support module:

```
SUPPORT=/home/controls/epics/R3.14.12.2/support  
AUTOSAVE=$(SUPPORT)/autosave
```

## 2. Add binary and DBD info to xyzApp/Db/Makefile:

```
YourProduct_DBD += asSupport.dbd  
YourProduct_LIBS += autosave
```

## 3. Use the support module in the IOC startup file:

```
cd ${AUTOSAVE}  
dbLoadRecords "db/save_restoreStatus.db", "P=demo"  
set_requestfile_path("/home/controls/var")
```

Details on how to use a support module depend on the specific one, including names of provided **\*.dbd**, **binary**, **\*.db**, **IOC commands**

# HowTo: Add Database files

## 1. Create xyzApp/Db/another.db

For simple database, can test via  
`softIoc -d another.db`

## 2. Add to xyzApp/Db/Makefile:

`DB += another.db`

## 3. make

Now it's under db/another.db

## 4. Add to iocBoot/iocwhatever/st.cmd

`dbLoadRecords "db/another.db", "macro=value"`

## 5. (Re-)start the IOC

# Summary

**makeBaseApp.pl creates the IOC skeleton**

## **Good practice:**

- Use `makeBaseApp.pl -t example...` for copy/paste.**
- Create empty operational setup, and only paste-in what you need.**
- Do it in small steps.**