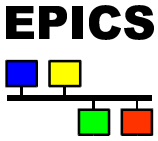
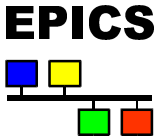


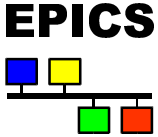
IOC Development Environment

Andrew Johnson
APS



Preamble

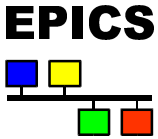
- This talk describes the IOC development environment that comes with EPICS Base R3.14.1
- Sometimes called makeBaseApp
  - After the PERL script that creates application areas
- Not the only possible approach
  - R3.12 had several different development environments
  - Sites can develop and use their own build system if desired
  - Some R3.13 sites use UAE, an environment developed by Keck/UKIRT
- It uses the same major structure and build rules as Base
- Allows independent compilation of the same source modules for multiple target architectures



Reference Documentation

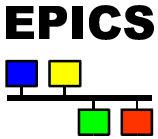
EPICS: Input / Output Controller
Application Developer's Guide
Release 3.14.1, 20 Dec 2002

- Accessible through the R3.14.1 page on the EPICS website
- Chapter 4 covers the EPICS build system in great detail, and is more recent (and accurate) than much of this talk
- The R3.14 build system has evolved quite a bit since R3.13.x



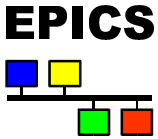
Purpose of an IOC DE

- What is an IOC development area for?
  - A place to collect files and configuration data associated with
    - one or more similar IOCs
    - device, driver and/or record support
    - other related software
  - Provides simple ways to configure and automate some very complex compilation procedures involving
    - Databases and database templates
    - Startup command files
    - Record, device & driver support
    - SNL programs
    - Other IOC code
    - Portable CA Server applications
    - Other host software



Other Features

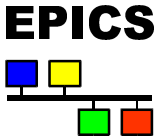
- IOC Software is usually divided into different <top> areas
  - Each <top> area is managed separately
  - A <top> may use products from other <top> areas
  - EPICS Base is mainly regarded as just another <top>
- The Gnu version of make is used to build all products
  - Almost every directory has a Makefile in it
  - Make recursively descends through the directory structure
    - Determines what needs to be [re]built
    - Invokes compilers and other tools as instructed in the Makefile
- CVS can be used for revision control of source code and configuration files



<top> Directory Structure

- The `example` directory is an application <top>
- A <top> is structured like this:

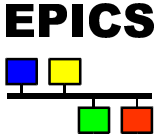
```
<top>/
        configure/          Configuration data files
        xxxApp/             All source files except startup
                src/        Source code
                xxxSrc/     There can be multiple xxxSrc directories
                Db/         Databases, templates & substitutions files
                xxxDb       There can be multiple xxxDb directories
                ...         May have others dirs too such as adl, edm, etc
        yyyApp/             Any number of *App directories
        iocBoot/            Only one iocBoot
                iocxxx      Directory for each ioc
                ...
        <install directories>
```



<top> Install Directories

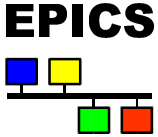
- By default, build products are installed into various subdirectories under <top>:

<top>/	
bin/	Object files and executables
linux-x86/	
vxWorks-68040/	
lib/	Object libraries
linux-x86/	etc.
include/	e.g. xxxRecord.h
dbd/	menu, recordtype, device, driver,etc
db/	record instances, templates, etc
javalib/	.jar files



<top>/configure files

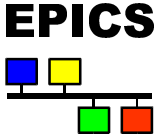
- Files meant to be modified
  - `CONFIG` - Can override make variables
    `CROSS_COMPILER_TARGET_ARCHS = vxWorks-68040`
  - `CONFIG_APP`
    Some other definitions
  - `RELEASE`
    Location of other <top> areas used in this application
    `EPICS_BASE=/usr/local/epics/R3.14.1/base`
- Other files not meant to be modified
  - `RULES*`
    Files containing (pointers to) make rules in base
  - `Makefile`
    Some configuration tasks



<top>/xxxApp/src files

- base.dbd - definitions supplied by base
  - Edit this if you don't want to load some base record or device types
  - Links to menu, recordtype, device, drivers provided by base
  - Earlier versions of base included hardware support too, not R3.14.1
  - Not fully expanded, contains many component include statements

```
include “menuGlobal.dbd”
include “menuConvert.dbd”
include “aiRecord.dbd”
...
include "waveformRecord.dbd"
device(ai,CONSTANT,devAiSoft,"Soft Channel")
...
device(waveform,CONSTANT,devWfSoft,"Soft Channel")
```



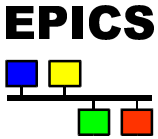
xxxApp/src continued

- Makefile defines what is to be built, from which source files

```
TOP=../..
include $(TOP)/configure/CONFIG
DBDINC += xxxRecord
DBD += example.dbd
PROD_IOC = example
example_SRCS += xxxRecord.c devXxxSoft.c exampleMain.cpp
example_SRCS += example_registerRecordDeviceDriver.cpp
example_LIBS += iocsh miscIoc rsrvIoc dbtoolsIoc asIoc dbIoc
example_LIBS += registryIoc dbStaticIoc ca Com
include $(TOP)/configure/RULES
```

- ExampleInclude.dbd

```
include “base.dbd”
include “xxxRecord.dbd”
device(xxx,CONSTANT,devXxxSoft,”Soft Channel”)
```



xxxApp/xxxDb/Makefile

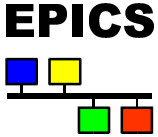
- Databases installed into <top>/db
  - `DB += xxx.db`
- Template files expanded on host
  - yyy.template has the format

```
record(ai,"$(USER):aiExample$(NO)"){
  … }
```

  - yyy.substitutions contains

```
file yyy.template {
  {USER="mrk",NO="1"}
  {USER="mrk",NO="2"}
}
DB += yyy.db
USES_TEMPLATE += yyy.template
```

- Template files to be expanded at boot time via dbLoadTemplate()
  - `DB += zzz.template zzz.substitutions`
- Support for capfast generated files
- Can also expand .dbd files



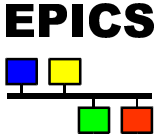
iocBoot/iocxxx

- iocBoot/iocxxx/Makefile
  Creates `cdCommands` file for vxWorks targets. Make sure that
  - `ARCH = <arch>`
  - is defined correctly in the Makefile
- For a vxWorks target the `cdCommands` output file looks like

```
startup="<full path to iocxxx>"
appbin="<full path to top/bin/arch>"
...
```

- A vxWorks `st.cmd` file looks like

```
< cdCommands
cd appbin
ld < iocCore
ld < xxxLib
cd startup
dbLoadDatabase("../../dbd/xxxApp.dbd")
dbLoadRecords("../../db/xxx.db","user=mrk")
iocInit
```



Make Targets (commands)

- It is possible to run make (gnumake) in any <top> subdirectory
- <top>
  - `make clean uninstall`
    Removes all products, leaving just the original source files
  - `make`
    Builds and installs everything that is not up to date
- configure
  - `make`
    Constructs target-specific files from contents of RELEASE
- iocBoot
  - `make`
    Same as issuing command in each iocxxx subdirectory
- iocBoot/iocxxx
  - `make`
    Creates the cdCommands file, if appropriate for the IOC's architecture

Make Targets (commands) cont.

- xxxApp
  - `make <command>`

    Same as issuing command in each subdirectory
- xxxApp/xxxSrc
  - Builds in O.<arch>; installs products in <top>/<something>
  - `make`

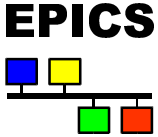
    Builds and installs all out of date components
  - `make <arch>`

    Build only for specified architecture(s), e.g.

    ```
    make vxWorks-ppc604
    ```
  - `make clean`

    Remove all O.<arch> directories
- XxxApp/xxxDb
  - `make`

    Generates and installs database instance files etc.



makeBaseApp

- To create a new application area, execute the commands

```
mkdir ~/heater
cd ~/heater
makeBaseApp.pl -t example water
makeBaseApp.pl -t example -i heater
```

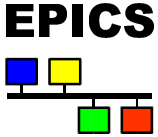
The first command creates:

```
<top>/
  Makefile
  configure/
          ...
  waterApp
    src/
          ...
    Db/
          ...
```

The second command creates:

```
<top>/iocBoot
  Makefile
  iocheater/
    …
```

makeBaseApp.pl can be run multiple times to create new subdirectories



Application Templates

- MakeBaseApp uses templates for the files it creates
- New application templates can be created by anyone
- Base supplies the following

```
<base>/templates/makeBaseApp/top/
  Makefile
  configure
  exampleApp
  exampleBoot
  simpleApp
  simpleBoot
```

- Template files undergo textual substitutions when installed