COSYLAB
CONTROL SYSTEM LABORATORY

EPICS

EPICS Lecture @ KEK

IOC Overview

Takashi Nakamoto
June 25th, 2013

Based on presentation by E. Norum, APS

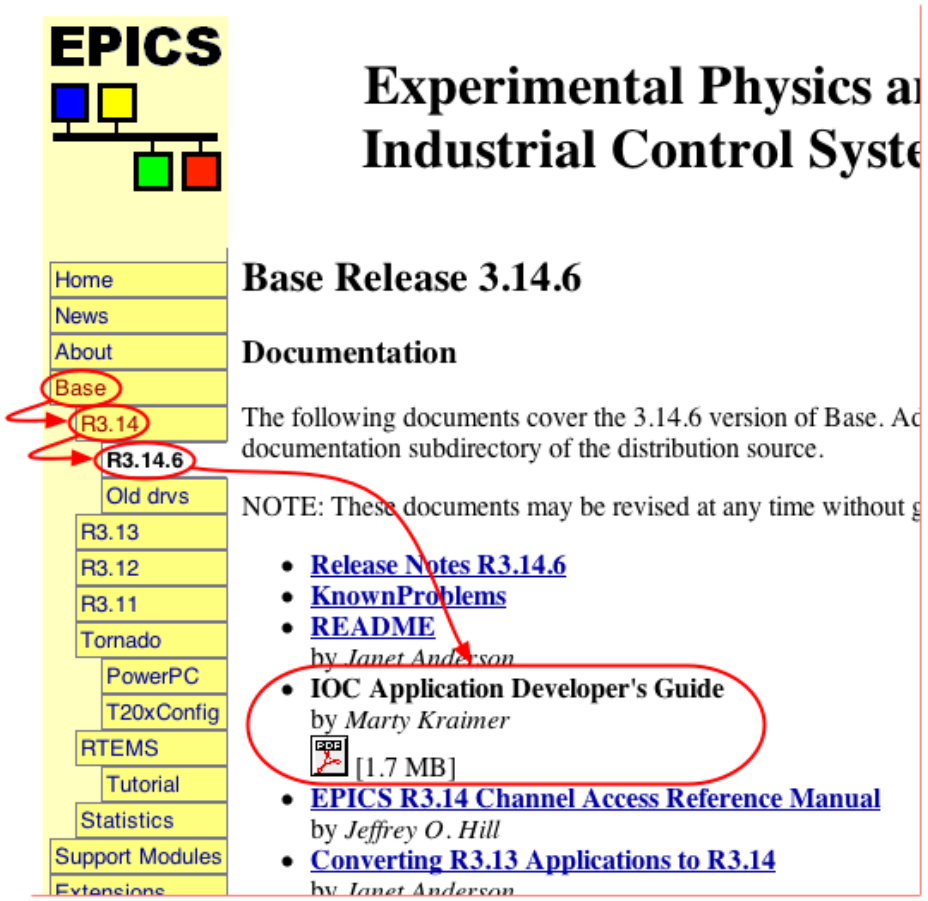
IOC Overview

- What is an EPICS Input/Output Controller
- How to create a new IOC application
- How to build an IOC application
- How to run an IOC application on various platforms
- Console interaction with an IOC application (iocsh)

Reference

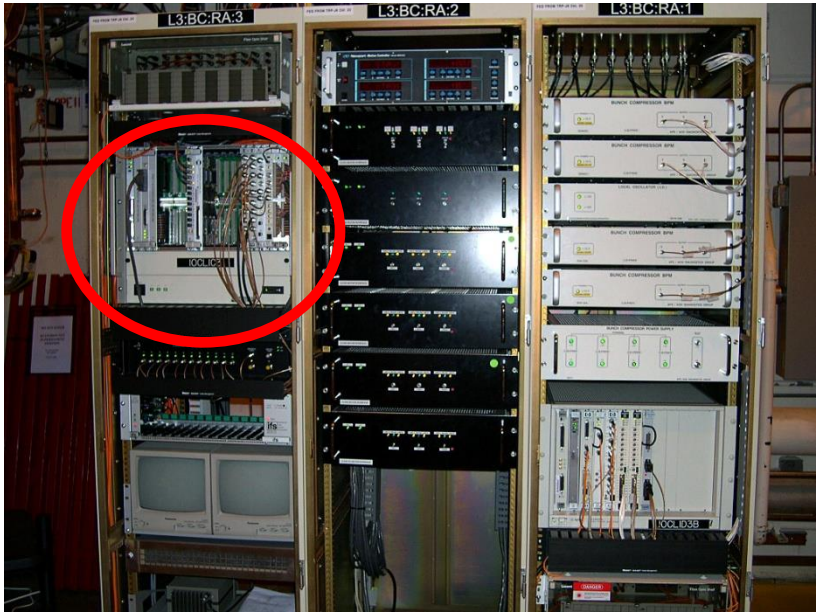
EPICS: Input/Output Controller
Application Developers Guide

Go to EPICS home page:
http://www.aps.anl.gov/epics/
then follow links, as shown



What is an Input/Output Controller?

The answer used to be easy – “A single-board computer running the vxWorks real-time operating system and installed in a VME chassis”.



What is an Input/Output Controller?

An IOC can also be an embedded microcontroller, a rack-mount server, a laptop PC or Mac, a desktop PC or Mac, or a standalone single-board computer.

It may be running on Linux, Windows, Solaris, Darwin, RTEMS, HP-UX or vxWorks

What is an Input/Output Controller?

Some definitions from the first lectures:

- A computer running *iocCore,* a set of EPICS routines used to define process variables and implement real-time control algorithms

- *iocCore* uses database records to define process variables and their behavior

What does an Input/Output Controller do?

- **As its name implies, an IOC often performs input/output operations to attached hardware devices.**
- **An IOC associates the values of EPICS process variables with the results of these input/output operations.**
- **An IOC can perform sequencing operations, closed-loop control and other computations.**

'Host-based' and 'Target' IOCs

- **'Host-based' IOC**
  - Runs in the same environment as which it was compiled
  - 'Native' software development tools (compilers, linkers)
  - Sometimes called a 'Soft' IOC
  - IOC is an program like any other on the machine
  - Possible to have many IOCs on a single machine
- **'Target' IOC**
  - Runs in a different environment than where compiled
  - 'Cross' software development tools
  - Linux, VxWorks, RTEMS
  - IOC boots from some medium (usually network)
  - IOC is the only program running on the machine

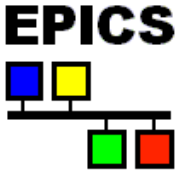
IOC Software Development Area

- **IOC software is usually divided into different <top> areas**
  - Each <top> provides a place to collect files and configuration data associated with one or more similar IOCs
  - Each <top> is managed separately
  - A <top> may use products from other <top> areas (EPICS base, for example can be thought of as just another <top>)

IOC Software Development Tools

- **EPICS uses the GNU version of make**
  - Almost every directory from the <top> on down contains a ‘Makefile’
  - Make recursively descends through the directory tree
    - *Determines what needs to be [re]built*
    - *Invokes compilers and other tools as instructed in Makefile*
  - GNU C/C++ compilers or vendor compilers can be used
- **No fancy ‘integrated development environment’ yet**

IOC Application Development Examples



The following slides provide step-by-step examples of how to:

- **Create, build, run the example IOC application on a 'host' machine (Linux, Solaris, Darwin, etc.)**
- **Create, build, run the example IOC application on a vxWorks 'target' machine**

Each example begins with the use of ‘makeBaseApp.pl’

The 'makeBaseApp.pl' program

- **Part of EPICS base distribution**
- **Populates a new, or adds files to an existing, <top> area**
- **Requires that your environment contain a valid EPICS_HOST_ARCH (EPICS base contains scripts which can set this as part of your login sequence)**
  - linux-x86, darwin-ppc, solaris-sparc, win32-x86
- **Creates different directory structures based on a selection of different templates**
- **Commonly-used templates include**
  - ioc - Generic IOC application skeleton
  - example - Example IOC application

Creating and initializing a new <top>

- **Create a new directory and run makeBaseApp.pl from within that directory**
  - mkdir lectureExample
  - cd lectureExample
  - /usr/local/iocapps/R3.14.6/base/bin/linux-x86/makeBaseApp.pl -t example first

---

- Provide full path to makeBaseApp.pl script
  <*base*>/bin/<*arch*>/makeBaseApp.pl
- The template is specified with the '-t' argument
- The application name (firstApp) is specified with the 'first' argument

<top> directory structure

- **The makeBaseApp.pl creates the following directory structure in <top> (lectureExample):**

  configure/ - Configuration files

  firstApp/- Files associated with the ‘firstApp’ application

  Db/ - Databases, templates, substitutions

  src/ - Source code

- **Every directory also contains a ‘Makefile’**

<top>/configure files

- **Some may be modified as needed**
  - CONFIG

    Specify make variables (e.g. to build for a particular target):

    CROSS_COMPILER_TARGET_ARCHS = vxWorks-68040
  - RELEASE

    Specify location of other <top> areas used by applications in this <top>area.
- **Others are part of the (complex!) build system and should be left alone.**

Create a host-based IOC boot directory

- **Run makeBaseApp.pl from the <top> directory**
- **‘-t example’ to specify template**
- **‘-i’ to show that IOC boot directory is to be created**
- **‘-a <*arch*>’ to specify hardware on which IOC is to run**
- **name of IOC**

➢ /usr/local/iocapps/R3.14.6/base/bin/linux-x86/makeBaseApp.pl

-t example -i -a linux-x86 first

- **If you omit the ‘-a <*arch*>’ you’ll be presented with a menu of options from which to pick**

<top> directory structure

- **The command from the previous slide creates an additional directory in <top>:**

  iocBoot/ - Directory containing per-IOC boot directories

  iocfirst/ - Boot directory for ‘iocfirst’ IOC

Build the application

- **Run the GNU make program**
  - ‘make’ on Darwin, Linux, Windows
  - ‘gnumake’ on Solaris
- **make**

  or
- **make -w**
- **Runs lots of commands**

<top> directory structure after running make

- **These additional directories are now present in <top>**

  bin/ - Directory containing per-architecture directories

  linux-x86/ - Object files and executables for this architecture

  lib/ - Directory containing per-architecture directories

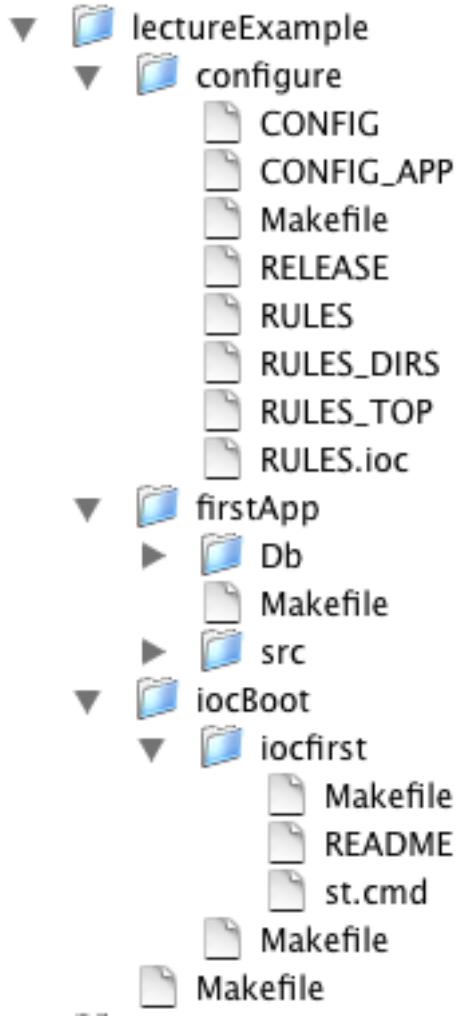
  linux-x86/ - Object libraries for this architecture

  dbd/ - Database definition files

  db/ - Database files (record instances, templates)

- **There may be other directories under bin/ and lib/, too**.

<top> directory structure after running make



IOC startup

- **IOCs read commands from a startup script**
  - Typically ‘st.cmd’ in the <top>/iocBoot/<iocname>/ directory
- **vxWorks IOCs read these scripts with the vxWorks shell**
- **Other IOCs read these scripts with the iocsh shell**
- **Command syntax can be similar but iocsh allows more familiar form too**
- **Script was created by ‘makeBaseApp.pl -i’ command**
- **For a ‘real’ IOC you’d likely add commands to configure hardware modules, start sequence programs, update log files, etc.**

Example application startup script

```
1  #!../../bin/linux-x86/first
2
3  ## You may have to change first to something else
4  ## everywhere it appears in this file
5
6  < envPaths
7
8  cd ${TOP}
9
10 ## Register all support components
11 dbLoadDatabase("dbd/first.dbd")
12 first_registerRecordDeviceDriver(pdbbase)
13
14 ## Load record instances
15 dbLoadRecords("db/dbExample1.db","user=norumeHost")
16 dbLoadRecords("db/dbExample2.db","user=norumeHost,no=1,scan=1 second")
17 dbLoadRecords("db/dbExample2.db","user=norumeHost,no=2,scan=2 second")
18 dbLoadRecords("db/dbExample2.db","user=norumeHost,no=3,scan=5 second")
19 dbLoadRecords("db/dbSubExample.db","user=norumeHost")
20
21 ## Set this to see messages from mySub
22 #var mySubDebug 1
23
24 cd ${TOP}/iocBoot/${IOC}
25 iocInit()
26
27 ## Start any sequence programs
28 #seq sncExample,"user=norumeHost”
```

Example application startup script

```
1  #!../../bin/linux-x86/first
```

- This allows a host-based IOC application to be started by simply executing the st.cmd script
- If you're running this on a different architecture the 'linux-x86' will be different
- If you gave a different IOC name to the 'makeBaseApp.pl -i' command the 'first' will be different
- Remaining lines beginning with a '#' character are comments

Example application startup script

6 < envPaths

- The application reads commands from the ‘envPaths’ file created by ‘makeBaseApp -i’ and ‘make’
- The envPaths file contains commands to set up environment variables for the application:
  - Architecture
  - IOC name
  - <top> directory
  - <top> directory of each component named in configure/RELEASE
- These values can then be used by subsequent commands

```
epicsEnvSet(ARCH,"linux-x86")
epicsEnvSet(IOC,"iocfirst")
epicsEnvSet(TOP,"/home/phoebus/NORUME/lectureExample")
epicsEnvSet(EPICS_BASE,"/usr/local/iocapps/R3.14.6/base")
```

Example application startup script

```
8  cd ${TOP}
```

- The working directory is set to the value of the ${TOP} environment variable (as set by the commands in ‘envPaths’)
- Allows use of relative path names in subsequent commands

Example application startup script

```
11  dbLoadDatabase("dbd/first.dbd")
```

- Loads the database definition file for this application
- Describes record layout, menus, drivers

Example application startup script

12 first_registerRecordDeviceDriver(pdbbase)

- Registers the information read from the database definition files

Example application startup script

```
15  dbLoadRecords("db/dbExample1.db","user=norumeHost")
16  dbLoadRecords("db/dbExample2.db","user=norumeHost,no=1,scan=1 second")
17  dbLoadRecords("db/dbExample2.db","user=norumeHost,no=2,scan=2 second")
18  dbLoadRecords("db/dbExample2.db","user=norumeHost,no=3,scan=5 second")
19  dbLoadRecords("db/dbSubExample.db","user=norumeHost")
```

- Read the application database files
  - These define the records which this IOC will maintain
  - A given file can be read more than once (with different macro definitions)

Example application startup script

```
24  cd ${TOP}/iocBoot/${IOC}
```

- The working directory is set to the per-IOC startup directory

Example application startup script

25 iocInit()

- Activates everything
- After reading the last line of the ‘st.cmd’ script the IOC continues reading commands from the console
  - Diagnostic commands
  - Configuration changes

Running a host-based IOC

- Change to IOC startup directory (the one containing the st.cmd script)
  - **cd iocBoot/iocfirst**
- Run the IOC executable with the startup script as the only argument
  - **../../bin/linux-x86/first st.cmd**
- The startup script commands will be displayed as they are read and executed
- When all the startup script commands are finished the iocsh will display an ‘epics> ’ prompt and wait for commands to be typed.

```
iocInit()
#########################################################################
###  EPICS IOC CORE built on Jun 23 2004
###  EPICS R3.14.6 $R3-14-6$ $2004/05/28 19:27:47$
#########################################################################
Starting iocInit
## Start any sequence programs
#seq sncExample,"user=norumeHost"
iocInit: All initialization complete
epics>
```

Some useful iocsh commands

- Display list of records maintained by this IOC

epics> **dbl**

norumeHost:aiExample

norumeHost:aiExample1

norumeHost:aiExample2

norumeHost:aiExample3

norumeHost:calcExample

norumeHost:calcExample1

norumeHost:calcExample2

norumeHost:calcExample3

norumeHost:compressExample

norumeHost:subExample

norumeHost:xxxExample

- **Caution – some IOCs have a lot of records**

Some useful iocsh commands

- Display a record

epics> **dbpr norumeHost:aiExample**

ASG: DESC: Analog input DISA: 0 DISP: 0

DISV: 1 NAME: norumeHost:aiExample RVAL: 0

SEVR: MAJOR STAT: HIHI SVAL: 0 TPRO: 0

VAL: 9

epics> **dbpr norumeHost:aiExample**

ASG: DESC: Analog input DISA: 0 DISP: 0

DISV: 1 NAME: norumeHost:aiExample RVAL: 0

SEVR: MINOR STAT: LOW SVAL: 0 TPRO: 0

VAL: 4

- **dbpr <recordname> 1** prints more fields
- **dbpr <recordname> 2** prints even more fields, and so on

Some useful iocsh commands

- Show list of attached clients

epics> **casr**

Channel Access Server V4.11

No clients connected.

- **casr 1** prints more information
- **casr 2** prints even more information

Some useful iocsh commands

- Do a ‘put’ to a field

epics> **dbpf norumeHost:calcExample.SCAN "2 second"**

DBR_STRING: 2 second

- Arguments with spaces must be enclosed in quotes

Some useful iocsh commands

- The ‘help’ command, with no arguments, displays a list of all iocsh commands
  - 90 or so, plus commands for additional drivers
- With arguments it displays usage information for each command listed

epics> **help dbl dbpr dbpf**

dbl 'record type' fields

dbpr 'record name' 'interest level'

dbpf 'record name' value

Terminating a host-based IOC

- Type '**exit**' to the iocsh prompt
- Type your 'interrupt' character (usually control-C)
- Kill the process from another terminal/window

Review

- IOC applications can be host-based or target-based
- The makeBaseApp.pl script is used to create IOC application modules and IOC startup directories
- <top>/configure/RELEASE contents specify location of other <top> areas used by this <top> area
- <top>/iocBoot/<iocname>/st.cmd is the startup script for IOC applications
- The EPICS build system requires the use of GNU make
- The EPICS Application Developer's Guide contains a wealth of information