Input/Output Controller (IOC) Development

Andrew Johnson

Slides by Eric Norum

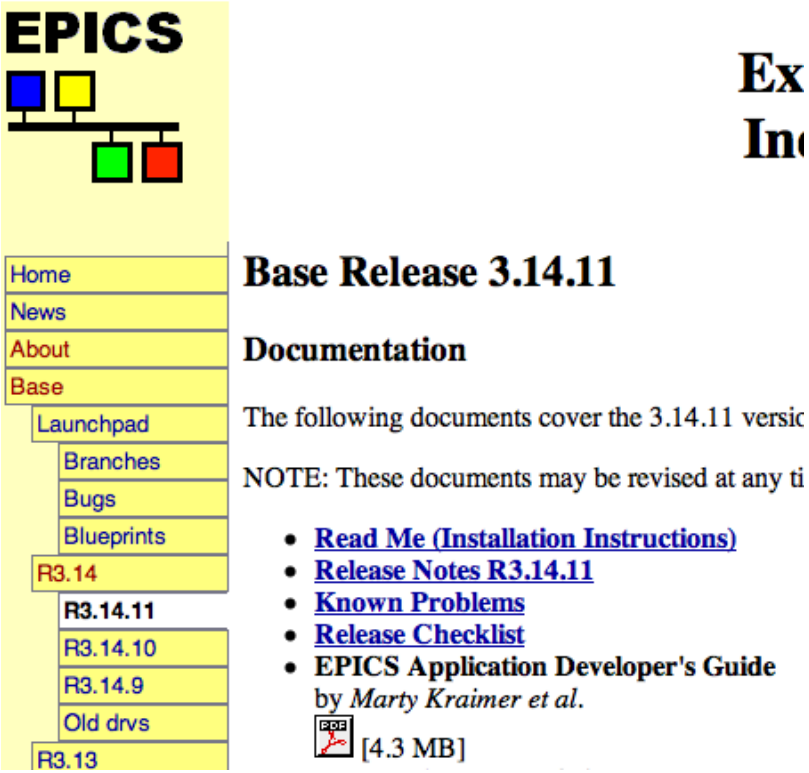
Overview

- 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

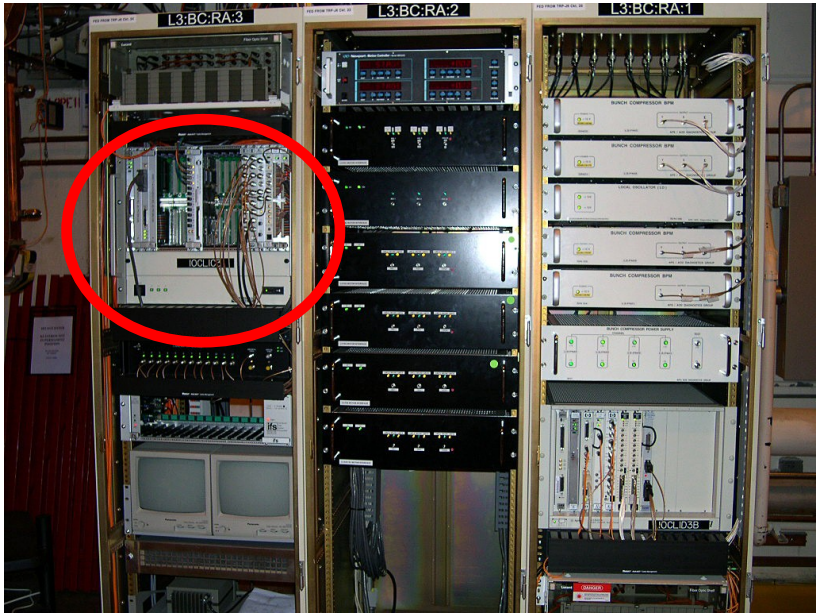
EPICS: Input/Output Controller
Application Developers Guide

From EPICS home page:
http://www.aps.anl.gov/epics/
Under the tabs Base->R3.14 click on R3.14.11 or R3.14.12. Then click on “EPICS Application Developer’s Guide” or the PDF icon immediately below it.



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?

Now an IOC can be an embedded micro-controller, a rack-mount server, a laptop PC or Mac, a desktop PC or Mac, a standalone single-board computer, or even an FPGA chip.

It may run on Linux, Windows, Solaris, Darwin, FreeBSD, RTEMS or VxWorks

• ‘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
  - 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 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 have EPICS_HOST_ARCH set
  - e.g. linux-x86, darwin-x86, solaris-sparc, win32-x86
  - EPICS base contains scripts which can set this as part of your login sequence
- 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
/<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 this directory structure in <*top*>

```
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_SITE
    - 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.
- Other files in <*top*>/configure 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

```
/<base>/bin/linux-x86/makeBaseApp.pl
        -t example -i -a linux-x86 first
```

  - '**`-t example`**' specifies the example template
  - '**`-i`**' asks that an IOC boot directory be created
  - '**`-a`** <*arch*>' to specify hardware on which IOC is to run
  - name of IOC

- If you omit the '**`-a`** <*arch*>' you'll be presented with a menu of architectures from which to pick

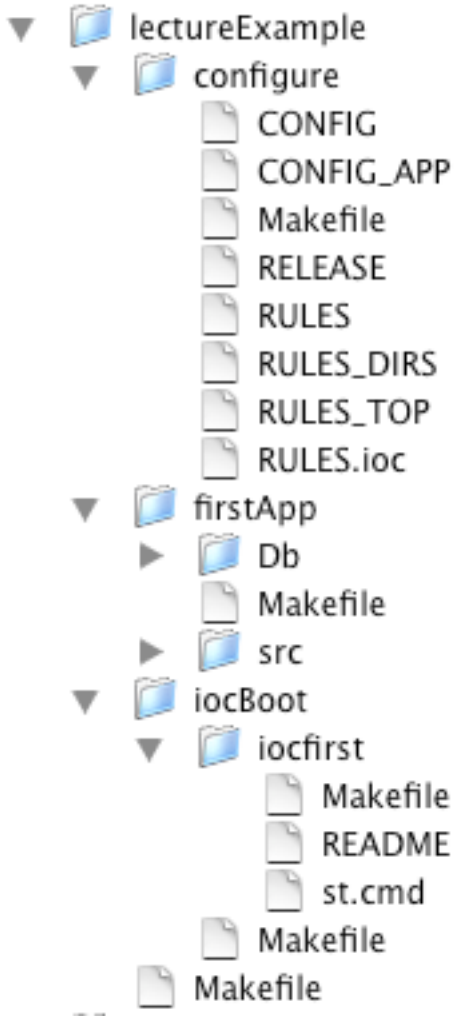
<*top*> directory structure

- The command from the previous slide creates additional directories in <*top*>

  iocBoot/ - Directory containing per-IOC boot directories

  iocfirst/ - Boot directory for ‘iocfirst’ IOC

Final <*top*> directory structure



Build the application

- Run the GNU make program
  - ‘make’ on Darwin, Linux, Windows
  - ‘gnumake’ or ‘gmake’ 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 additional directories created under bin/ and lib/

IOC startup

- IOCs read commands from a startup script
  - Typically '**`st.cmd`**' in the <*top*>**`/iocBoot/`**<*iocname*> directory
- On VxWorks the target shell runs these scripts
- On other OSs the EPICS iocsh shell runs them
  - Command syntax is similar, iocsh is more flexible
- Script is created by 'makeBaseApp.pl -i' command
- For a 'real' IOC you would add commands to configure hardware modules, start sequence programs, update log files, etc.

Example application startup script

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

## You may have to change first to something else
## everywhere it appears in this file

< envPaths

cd ${TOP}

## Register all support components
dbLoadDatabase "dbd/first.dbd"
first_registerRecordDeviceDriver pdbbase

## Load record instances
dbLoadTemplate "db/userHost.substitutions"
dbLoadRecords "db/dbSubExample.db","user=norumeHost"

## Set this to see messages from mySub
#var mySubDebug 1

## Run this to trace the stages of iocInit
#traceIocInit

cd ${TOP}/iocBoot/${IOC}
iocInit

## Start any sequence programs
#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 word '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 sets environment variables giving the IOC application’s
  - Target Architecture
  - IOC name
  - <*top*> directory
  - <*top*> directory for each component named in configure/RELEASE
- These values can be used by subsequent commands
  - epicsEnvSet(ARCH,"linux-x86")
  - epicsEnvSet(IOC,"iocfirst")
  - epicsEnvSet(TOP,"/home/NORUME/lectureExample")
  - epicsEnvSet(EPICS_BASE,"/opt/epics/iocapps/R3.14.11/base")

Example application startup script

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

- The working directory is set from the **`${TOP}`** environment variable (defined in ‘envPaths’)
- Allows subsequent commands to use relative paths

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 more information related to the database definition files

Example application startup script

```
15.  dbLoadTemplate "db/userHost.substitutions"
16.  dbLoadRecords "db/dbSubExample.db","user=norumeHost"
```

- Read the application database files
  - These define the records which this IOC will contain
  - A file may be used more than once (with different macro definitions)
  - An individual record’s field values may be spread across multiple files
    - Where a field is set more than once, the last value wins
    - All instances must specify the same record type

Example application startup script

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

- The working directory is set to the IOC’s 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

- Go to IOC startup directory (containing the st.cmd script)

```
cd iocBoot/iocfirst
```

- Run the IOC executable with the startup script as the argument

```
../../bin/linux-x86/first st.cmd
```

- The startup script lines are displayed as they are executed
- When the script has finished the IOC will display an iocsh prompt and wait for commands to be typed

```
iocInit()
…
iocInit: All initialization complete
epics>
```

Some useful iocsh commands

- The ‘help’ command, with no arguments, displays a list of all iocsh commands
  - 100 or so, plus any commands added by drivers and device support
- 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
```

Some useful iocsh commands

- Display list of records loaded 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
  - Use ‘dbgrep’ to list just the records that match a pattern

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

Terminating a host-based IOC

- Type ‘**`exit`**’ at the iocsh prompt
- Type your ‘interrupt’ character (usually control-C)
- Kill the process from another terminal/window

Create a VxWorks IOC boot directory

- Almost the same as for a host-based IOC
  - just the <*arch*> changes
- 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

```
~iocapps/R3.14.11/base/3-14-11-asd1/bin/solaris-sparc/makeBaseApp.pl
        -t example -i -a vxWorks-68040 first
```

VxWorks IOC startup script differences

- The startup script created by ‘makeBaseApp.pl -i’ for a VxWorks IOC is slightly different than for a host-based IOC
- Script interpreter
  - A host-based IOC uses the iocsh shell to run the script
  - A VxWorks IOC uses the VxWorks target shell instead of iocsh
  - The syntax accepted is subtly different
- Binary executable
  - A host-based IOC binary is a single executable program file
  - A VxWorks IOC loads the application from one or more binary files, under the control of the startup script

VxWorks IOC startup script differences

- The first few lines of the VxWorks example st.cmd script are:

```
## Example vxWorks startup file

## The following is needed if your board support package does…
## automatically cd to the directory containing its startup s…
#cd "/home/phoebus/NORUME/lectureExample/iocBoot/iocfirst"

< cdCommands
#< ../nfsCommands

cd topbin
## You may have to change first to something else
## everywhere it appears in this file

ld < first.munch
```

VxWorks IOC startup script differences

- The startup script reads commands from ‘cdCommands’ instead of from ‘envPaths’
  - This assigns values to VxWorks shell variables as well as to environment variables
- Subsequent ‘cd’ commands look like

```
cd top
```

rather than

```
cd ${TOP}
```

VxWorks IOC startup script differences

- The startup script contains command to load the binary files making up the IOC application

```
ld < first.munch
```

- VxWorks binary files have names ending in ‘.munch’

Running a VxWorks IOC

- Set up the VxWorks boot parameters

```
Press any key to stop auto-boot...
 6
[VxWorks Boot]: c
'.' = clear field;  '-' = go to previous field;  ^D = quit
boot device          : ei
processor number     : 0
host name            : phoebus
file name            : /usr/local/vxWorks/T202/mv167-asd7_nodns
inet on ethernet (e) : 192.168.8.91:ffffc00
inet on backplane (b):
host inet (h)        : 192.168.8.167
gateway inet (g)     :
user (u)             : someuser
ftp password (pw) (blank = use rsh): somepassword
flags (f)            : 0x0
target name (tn)     : iocnorum
startup script (s)   : /usr/local/epics/iocBoot/iocfirst/st.cmd
other (o)            :
```

Running a vxWorks IOC

```
host name                  : Name of your FTP server
file name                  : Path to the VxWorks image on the FTP server
inet on ethernet (e)       : IOC IP address : netmask
inet on backplane (b):
host inet (h)              : FTP server IP address
gateway inet (g)           :
user (u)                   : User name to log into FTP server
ftp password (pw) (blank = use rsh): Password for FTP account
flags (f)                  : Special BSP flags
target name (tn)           : IOC name
startup script (s)         : Path to IOC startup script on FTP server
other (o)                  :
```

- Once these parameters have been set a reboot will start the IOC

VxWorks shell

- The VxWorks shell requires that commands be entered in a slightly different form
  - String arguments must be enclosed in double quotes
  - Arguments must be separated by commas
  - There is no EPICS-specific ‘help’ command
  - Many VxWorks-specific commands are available
- For example, the ‘dbpf’ command shown previously could be entered as:

```
dbpf “norumeHost:calcExample.SCAN”,”2 second”
```

or as:

```
dbpf(“norumeHost:calcExample.SCAN”,”2 second”)
```

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
- VxWorks IOCs use the VxWorks shell, non-vxWorks IOCs use iocsh
- The EPICS Application Developer's Guide contains a wealth of information