Handout for EPICS Introduction Hands-on Training

at

EPICS Collaboration Meeting, October 2011

Dr. Elke Zimoch
elke.zimoch@psi.ch
Paul Scherrer Institute, Switzerland

Reminder: IOC System in real … step by step

1. Prepare Application (make EPICS base + driver) ✓ Done
2. Write startup script (st.cmd file)
3. Write records (db file or substitution + template files)
4. Save to boot directory (here /home/ioc)
5. Set boot parameters of VME computer ✓ Done
6. Reboot (command is “`reboot`”)
   - Load and start operating system (vxWorks)
   - Load and start st.cmd
     - Load dbd files
     - Configure hardware
     - Load records
     - Start IOC (iocInit) and create records
7. Test functionality
8. Debug files and go to step 6

Cooler Exercise

This exercise was originally developed by John Mclean from Argone National Laboratory for the ”Getting started with EPICS” lectures series.
In all files used for IOC configuration lines starting with # are interpreted as comments.
Do not use non-ASCII characters (like ä, ü, é, μ, π, …) in <u>comments</u> – this may crash the IOC.

Problem Description:

In the LINAC, we have a water chiller that must be turned ON whenever the average temperature of two temperature sensors rises above a set point. The set point is nominally 45 degrees centigrade.
I relay on your imagination for this exercise. The first two potentiometers will be your temperature sensors, to allow you full control over the ”temperatures”. The cooler will be represented by the first LED.

Content of the startup script (st.cmd)

This example is for vxWorks 5.5 on a MVME2306 CPU board. The installation directory is /home/ioc. Drivers for Hytec binary I/O and analog I/O are included into the IOC application.

```
# set environment on IOC
ARCH = "vxWorks-ppc604"
IOC  = "MTEST-VME-T1"
TOP  = "/home/ioc"

nfsMount &sysBootHost,TOP
cd TOP
cd "bin"
cd ARCH

# load and start IOC Application
ld < trainingIOC.munch

cd TOP
dbLoadDatabase("dbd/trainingIOC.dbd")
trainingIOC_registerRecordDeviceDriver(pdbbase)

cd IOC

# Binary Card
# Dim8001Configure
(slot,not_used,IVec,ILev,not_used,debounce,clock,scan,io)
  Dim8001Configure (   2,       0,   0,   0,       0,
0,    0,  10, 2)
# Analogue Card
# Hy8401Configure
(card,base_adr,ivec,ilev,en,aitype,clksrc,clkrate,inhibit,s
ample,space,trig)
  Hy8401Configure (   3,  0x1A00,   0,   0, 0,     0,
0,     14,      0,     1,    1,   0)
# Hy8402Configure (card,base_adr, 0, 0, 0, 0)   last 4
probably not used
  Hy8402Configure (   3,  0x1800, 0, 0, 0, 0)

# load records
dbLoadRecords("COOLER.db")
#dbLoadTemplate("COOLER.subs")

iocInit()
```

Content of db file:

Exchange the red one with your number (do avoid duplicate record names).

```
record (ai, "MTRT1-LI-COOL:TEMP1")
{
    field (DESC, "Sensor T1")
    field (DTYP, "Hy8401")
    field (INP,  "#C3 S0 @")
    field (EGU,  "Grad C")
    field (PREC, "1")
    field (LINR, "LINEAR")
    field (EGUF, "100")
    field (EGUL, "-100")
    field (HOPR, "100")
    field (LOPR, "0")
    field (HIGH, "51")
    field (HIHI, "52")
    field (SCAN, ".1 second")
}

record (ai, "MTRT1-LI-COOL:TEMP2")
{
    field (DESC, "Sensor T2")
    field (DTYP, "Hy8401")
    field (INP,  "#C3 S1 @")
    field (EGU,  "Grad C")
    field (PREC, "1")
    field (LINR, "LINEAR")
    field (EGUF, "100")
    field (EGUL, "-100")
    field (HOPR, "100")
    field (LOPR, "0")
    field (HIGH, "51")
    field (HIHI, "52")
    field (SCAN, ".1 second")
}
```

```
record (calc, "MTRT1-LI-COOL:COMPARE")
{
    field (DESC, "Calculation")
    field (INPA, "MTRT1-LI -COOL:TEMP1")
    field (INPB, "MTRT1-LI -COOL:TEMP2")
    field (INPC, "45")
    field (CALC, "((A+B)/2) > C")
    field (SCAN, "10 second")
    field (FLNK, "MTRT1-LI -COOL:SW")

}

record (bo, "MTRT1-LI-COOL:SW")
{
    field (DESC, "Switch for Cooler")
    field (DOL, "MTRT1-LI -COOL:COMPARE")
    field (ZNAM, "OFF")
    field (ONAM, "ON")
    field (DTYP, "Dim8001")
    field (OUT,  "#C2 S32 @")
    field (SCAN, "10 second")
    field (OMSL, "closed_loop")
}
```

Duplicate the cooler system

Split db file into a substitution file (<xxx>.subs) and a template file (<yyy>.template).
Define macros in the substitution file and use them in the template file.
Example of a substitution file:
Exchange the red one with your number (do avoid duplicate record names).

```
file COOLER.template {
  {
    DEVICE = MTRT1-LI1
    ADC1   = "C3 S0"
    ADC2   = "C3 S1"
    OUTPUT = "C2 S32"
  }
  {
    DEVICE = MTRT1-LI2
    ADC1   = "C3 S2"
    ADC2   = "C3 S3"
    OUTPUT = "C2 S33"
  }
}
```

```
## Alternative syntax for substitution files:
#file COOLER.template {
#  pattern
#    {DEVICE     ADC1    ADC2    OUTPUT  }
#    {MTRT1-LI1  "C3 S0" "C3 S1" "C2 S32"}
#    {MTRT1-LI2  "C3 S2" "C3 S3" "C2 S33"}
#}
```

Reuse the db file from above as template file: make a copy and rename the copy to `COOLER.template`. In the file replace the four values for macros with $(MacroName) like:

db-file:

```
  record (ai, "MTRT1-LI-COOL:TEMP1")
    field (INP,  "#C3 S0 @")
```

template-file:

```
  record (ai, "$(DEVICE)-COOL:TEMP1")
    field (INP,  "#$(ADC1) @")
```

Things to do (Exercises)

Refine the medm GUI (Graphical User Interface)

- Display both cooler sets.
- Every value displayed as a number or in a graph should have the unit attached to it. Assume the user of the panel uninformed about the things behind it. The panel should be intuitive and self explained.
- Display the alarm states of the Temperatures
- The switch state of the chiller should be given as text and with an ”LED” graphic, which is lighted yellow when the device is switched on
- Plot the temperatures with a “Strip Chart” Widget in the medm window

More records to do

- Display the average temperature in the GUI. You need to create another calc record.
- The temperature limit of 45 deg C should be accessible (read and write) using a new record. Display this record in medm.
- There should be another temperature limit: a heater should be switched on if the temperature drops below a lower limit. The hardware addresses of these heaters are “#C2 S34 @” and “#C2 S35 @”.
- The cooler (and the heater independently) should have a “always on” and “never on” option.

10 really neat Things about EPICS

1. It is free
2. It is Open Source
3. There are lots of users
4. All a client needs to know to access data is a PV name
5. You can pick the best tools out there …
6. … or build your own
7. The boring stuff is already done
8. There is a lot of expertise available close by
9. A good contribution becomes internationally known
10. It does not matter, if you need 10 or 10 million PVs
    (it is scalable)

Where to learn more?

EPICS home page:	http://www.aps.anl.gov/epics/
PSI Training:	http://epics.web.psi.ch/training/
Tech Talk:	http://www.aps.anl.gov/epics/tech-talk/index.php
Getting Started with EPICS lecture series:	
	http://www.aps.anl.gov/epics/docs/GSWE.php