EPICS How-Tos

Jun 01, 2020

Contents

CHAPTER 1

Installation on Linux/UNIX/DARWIN (Mac)

1.1 What is EPICS about?

We assume that you know more or less what EPICS is. Here we want to start from scratch and get to the point where we have a working server, offering some PVs for reading (caget or pvget) and writing (caput or pvput), and we read and write on them from another terminal, either on the same machine or on another one in the same network. If you are going to use two different machines, you will have to repeat the steps for installing EPICS for both of them.

1.2 Prepare your system

You need `make`, `c++` and `libreadline` to compile from source. On macOS these dependencies can be installed by using e.g. `homebrew`. On Linux you can use `apt-get install`. The *Packages required for EPICS on Centos 8* document lists all of the packages required to build EPICS base, the sequencer, synApps modules, and areaDetector.

1.3 Install EPICS

```
mkdir $HOME/EPICS
cd $HOME/EPICS
git clone --recursive https://github.com/epics-base/epics-base.git
cd epics-base
make
```

After compiling you should put the path into `$HOME/.profile` or into `$HOME/.bashrc` by adding the following to either one of those files:

```
export EPICS_BASE=${HOME}/EPICS/epics-base
export EPICS_HOST_ARCH=$(${EPICS_BASE}/startup/EpicsHostArch)
export PATH=${EPICS_BASE}/bin/${EPICS_HOST_ARCH}:${PATH}
```

EpicsHostArch is a program provided by EPICS that returns the architecture of your system. Thus the code above should be fine for every architecture.

1.4 Test EPICS

Now log out and log in again, so that your new path is set correctly. Alternatively, you can execute the three lines above beginning with export directly from the terminal.

Run `softIoc` and, if everything is ok, you should see an EPICS prompt.

```
softIoc
epics>
```

You can exit with ctrl-c or by typing exit.

Voilà.

Ok, that is not very impressive, but at least you know that EPICS is installed correctly. So now let us try something more complex, which will hopefully suggest how EPICS works.

In whatever directory you like, prepare a file test.db that reads like

```
record(ai, "temperature:water")
{
    field(DESC, "Water temperature in the fish tank")
}
```

This file defines a record instance called temperature:water, which is an analog input (ai) record. As you can imagine DESC stays for Description. Now we start softIoc again, but this time using this record database.

```
softIoc -d test.db
```

Now, from your EPICS prompt, you can list the available records with the dbl command and you will see something like

```
epics> dbl
temperature:water
epics>
```

Open a new terminal (we call it nr. 2) and try the command line tools caget and caput. You will see something like

```
your prompt> caget temperature:water
temperature:water              0
your prompt> caget temperature:water.DESC
temperature:water.DESC         Water temperature in the fish tank
your prompt> caput temperature:water 21
Old : temperature:water              0
New : temperature:water              21
your prompt> caput temperature:water 24
Old : temperature:water              21
New : temperature:water              24
your prompt> caget temperature:water
temperature:water              24
 ... etc.
```

Now open yet another terminal (nr. 3) and try `camonitor` as

```
camonitor temperature:water
```

First, have a look at what happens when you change the temperature:water value from terminal nr. 2 using caput. Then, try to change the value by some tiny amounts, like 15.500001, 15.500002, 15.500003... You will see that camonitor reacts but the readings do not change. If you wanted to see more digits, you could run

```
camonitor -g8 temperature:water
```

For further details on the Channel Access protocol, including documentation on the `caput`, `caget`, `camonitor`... command line tools, please refer to the Channel Access Reference Manual.

In real life, however, it is unlikely that the 8 digits returned by your thermometer (in this example) are all significant. We should thus limit the traffic to changes of the order of, say, a hundredth of a degree. To do this, we add one line to the file `test.db`, so that it reads

```
record(ai, "temperature:water")
{
    field(DESC, "Water temperature in Lab 10")
    field(MDEL, ".01")
}
```

MDEL stands for Monitor Deadband. If you now run

```
softIoc -d test.db
```

with the new `test.db` file, you will see that `camonitor` reacts only to changes that are larger than 0.01.

This was just a simple example. Please refer to a recent Record Reference Manual for further information.

1.5 Create a demo/test ioc to test ca and pva

```
mkdir -p $HOME/EPICS/TEST/testIoc
cd $HOME/EPICS/TEST/testIoc
makeBaseApp.pl -t example testIoc
makeBaseApp.pl -i -t example testIoc
make
cd iocBoot/ioctestIoc
chmod u+x st.cmd
ioctestIoc> ./st.cmd
#!../../bin/darwin-x86/testIoc
< envPaths
epicsEnvSet("IOC","ioctestIoc")
epicsEnvSet("TOP","/Users/maradona/EPICS/TEST/testIoc")
epicsEnvSet("EPICS_BASE","/Users/maradona/EPICS/epics-base")
cd "/Users/maradona/EPICS/TEST/testIoc"
## Register all support components
dbLoadDatabase "dbd/testIoc.dbd"
testIoc_registerRecordDeviceDriver pdbbase
## Load record instances dbLoadTemplate "db/user.substitutions"
dbLoadRecords "db/testIocVersion.db", "user=junkes"
dbLoadRecords "db/dbSubExample.db", "user=junkes"
#var mySubDebug 1
#traceIocInit
cd "/Users/maradona/EPICS/TEST/testIoc/iocBoot/ioctestIoc"
iocInit
```

(continues on next page)

(continued from previous page)

```
Starting iocInit
############################################################################
## EPICS R7.0.1.2-DEV
## EPICS Base built Mar 8 2018
############################################################################
iocRun: All initialization complete
2018-03-09T13:07:02.475 Using dynamically assigned TCP port 52908.
## Start any sequence programs
#seq sncExample, "user=maradona"
epics> dbl
maradona:circle:tick
maradona:compressExample
maradona:line:b
maradona:aiExample
maradona:aiExample1
maradona:ai1
maradona:aiExample2
... etc. ...
epics>
```

Now in another terminal, one can try command line tools like

```
caget, caput, camonitor, cainfo (Channel Access)
pvget, pvput, pvlist, eget, ... (PVAccess)
```

1.6 Add the asyn package

```
cd $HOME/EPICS
mkdir support
cd support
git clone https://github.com/epics-modules/asyn.git
cd asyn
```

Edit `$HOME/EPICS/support/asyn/configure/RELEASE` and set `EPICS_BASE` like

```
EPICS_BASE=${HOME}/EPICS/epics-base
```

Comment `IPAC=...` and `SNCSEQ=...`, as they are not needed for the moment. The whole file should read:

```
#RELEASE Location of external products
HOME=/Users/maradona
SUPPORT=$(HOME)/EPICS/support
-include $(TOP)/../configure/SUPPORT.$(EPICS_HOST_ARCH)
# IPAC is only necessary if support for Greensprings IP488 is required
# IPAC release V2-7 or later is required.
#IPAC=$(SUPPORT)/ipac-2-14
# SEQ is required for testIPServer
#SNCSEQ=$(SUPPORT)/seq-2-2-5
# EPICS_BASE 3.14.6 or later is required
EPICS_BASE=$(HOME)/EPICS/epics-base
-include $(TOP)/../configure/EPICS_BASE.$(EPICS_HOST_ARCH)
```

Now, run

```
make
```

1.7 Install StreamDevice (by Dirk Zimoch, PSI)

StreamDevice does not come with its own top location and `top/configure` directory. It expects to be put into an already existing top directory structure. We can simply create one with `makeBaseApp.pl`

```
cd $HOME/EPICS/support
mkdir stream
cd stream/
makeBaseApp.pl -t support
git clone https://github.com/paulscherrerinstitute/StreamDevice.git
cd StreamDevice/
rm GNUmakefile
```

Now we must edit the `$HOME/EPICS/support/stream/configure/RELEASE`. The not-commented lines must read

```
# Variables and paths to dependent modules:
MODULES = ${HOME}/EPICS/support
# If using the sequencer, point SNCSEQ at its top directory:
#SNCSEQ = $(MODULES)/seq-ver
# EPICS_BASE should appear last so earlier modules can override stuff:
EPICS_BASE = ${HOME}/EPICS/epics-base
# These lines allow developers to override these RELEASE settings
# without having to modify this file directly.
-include $(TOP)/../RELEASE.local
#-include $(TOP)/../RELEASE.$(EPICS_HOST_ARCH).local
-include $(TOP)/configure/RELEASE.local
ASYN=$(MODULES)/asyn
```

Remember that `$(NAME)` works if it is defined within the same file, but `${NAME}` with curly brackets must be used if a shell variable is meant. It is possible that the compiler does not like some of the substitutions. In that case, replace the `${NAME}` variables with full paths, like `/Users/maradona/EPICS...`.

Finally run `make` (we are in the directory `...EPICS/support/stream/StreamDevice`)

CHAPTER 2

Packages required for EPICS on Centos 8

Contents

2.1 Overview

This document describes the packages that must be installed in order to build EPICS base, synApps, and areaDetector on a new Centos 8 system. For other versions of Linux the package manager and package names may be different, but the requirements are likely to be the same.

Add the Extra Packages for Enterprise Linux (EPEL) site for the dnf package manager. This site has additional packages that are needed:

```
dnf install epel-release
```

Edit `/etc/yum.repos.d/CentOS-PowerTools.repo` to set `enabled=1` to enable that repository, because it also contains needed packages.

2.2 Packages required to build EPICS base

```
dnf install gcc
dnf install gcc-c++
dnf install gcc-toolset-9-make
dnf install readline-devel
dnf install perl-ExtUtils-Install
```

2.3 Packages required by the sequencer

```
dnf install re2c
```

2.4 Packages required by epics-modules/asyn

```
dnf install rpcgen
dnf install libtirpc-devel
```

2.5 Packages required by the Canberra and Amptek support in epics-modules/mca

```
dnf install libnet-devel
dnf install libpcap-devel
dnf install libusb-devel
```

2.6 Packages required by the Linux drivers in epics-modules/measComp

```
dnf install libnet-devel
dnf install libpcap-devel
dnf install libusb-devel
```

2.7 Packages required by areaDetector/ADSupport/GraphicsMagick

```
dnf install xorg-x11-proto-devel
dnf install libX11-devel
dnf install libXext-devel
```

2.8 Packages required by areaDetector/ADEiger

```
dnf install zeromq-devel
```

2.9 Packages required to build aravis 7.0.2 for areaDetector/ADAravis

```
dnf install ninja-build
dnf install meson
dnf install glib2-devel
dnf install libxml2-devel
dnf install gtk3-devel
dnf install gstreamer1
dnf install gstreamer1-devel
dnf install gstreamer1-plugins-base-devel
dnf install libnotify-devel
dnf install gtk-doc
dnf install gobject-introspection-devel
```

2.10 Packages required to build areaDetector/ADVimba

```
dnf install glibmm24-devel
```

2.11 Packages required to build EDM

```
dnf install giflib
dnf install giflib-devel
dnf install zlib-devel
dnf install libpng-devel
dnf install motif-devel
dnf install libXtst-devel
```

2.12 Packages required to build MEDM

```
dnf install libXt-devel
dnf install motif-devel
```

CHAPTER 3

Creation of an Input/Output Controller (IOC)

An IOC allows to talk to devices e.g. via ethernet. Create a directory for the IOCs. For example `$HOME/EPICS/IOCs`

```
cd $HOME/EPICS
mkdir IOCs
cd IOCs
```

Create a top for an IOC called `sampleIOC`

```
mkdir sampleIOC; cd sampleIOC
makeBaseApp.pl -t example sampleIOC
makeBaseApp.pl -i -t example sampleIOC
Using target architecture darwin-x86 (only one available)
The following applications are available:
sampleIOC
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name? (just return)
```

Now, by running `make`, a sample IOC like the demo/test IOC is built. Next, we want to add asyn and StreamDevice to the IOC. For this, we add the stream and asyn libraries to the Makefile. Edit `sampleIOCApp/src/Makefile` and add the block

```
#add asyn and streamDevice to this IOC production libs
sampleIOC_LIBS += stream
sampleIOC_LIBS += asyn
```

The application must also load `asyn.dbd` and `stream.dbd` to use StreamDevice. This can be put into a generated dbd, e.g into `xxxSupport.dbd` which already gets included by the Makefile. So the `xxxSupport.dbd` now reads:

```
cat sampleIOCApp/src/xxxSupport.dbd
include "xxxRecord.dbd"
device(xxx,CONSTANT,devXxxSoft,"SoftChannel")
```

(continues on next page)

(continued from previous page)

```
#
include "stream.dbd"
include "asyn.dbd"
registrar(drvAsynIPPortRegisterCommands)
registrar(drvAsynSerialPortRegisterCommands)
registrar(vxi11RegisterCommands)
```

To find the dbd files, you have to add the paths to these files in `configure/RELEASE`:

```
...
# Build variables that are NOT used in paths should be set in
# the CONFIG_SITE file.
# Variables and paths to dependent modules:
SUPPORT = ${HOME}/EPICS/support
ASYN=$(SUPPORT)/asyn
STREAM=$(SUPPORT)/stream
# If using the sequencer, point SNCSEQ at its top directory:
#SNCSEQ = $(MODULES)/seq-ver
...
```

If `make` was done before, `make distclean` is probably required. Anyway, then `make`. The newly created IOC can be run with:

```
cd iocBoot/iocsampleIOC/
chmod u+x st.cmd
./st.cmd
```

Not very interesting yet, because there is no database file nor a protocol file.

```
ls -la sampleIOCApp/Db/
total 56
drwxr-xr-x 11 maradona  staff   374  Jun  1  16:47  .
drwxr-xr-x  5 maradona  staff   170  Jun  1  12:46  ..
-rw-r--r--  1 maradona  staff   523  Jun  1  12:46  Makefile
drwxr-xr-x  2 maradona  staff    68  Jun  1  16:47  O.Common
drwxr-xr-x  3 maradona  staff   102  Jun  1  16:47  O.darwin-x86
-rw-r--r--  1 maradona  staff  1761  Jun  1  12:46  circle.db
-rw-r--r--  1 maradona  staff  1274  Jun  1  12:46  dbExample1.db
-rw-r--r--  1 maradona  staff   921  Jun  1  12:46  dbExample2.db
-rw-r--r--  1 maradona  staff   286  Jun  1  12:46  dbSubExample.db
-rw-r--r--  1 maradona  staff   170  Jun  1  12:46  sampleIOCVersion.db
-rw-r--r--  1 maradona  staff   307  Jun  1  12:46  user.substitutions
```

Note that this is a `Db` directory and not the `db` directory that is in `./sampleIOC`. For MDOxxxx scopes by Tektronix, the database (`.db`) and protocol (`.proto`) file can look something like

```
cat MDO.db
record(stringin, $(P)$(R)idn){
    field(DESC, "Asks for info blabla")
    field(DTYP, "stream")
    field(INP, "@MDO.proto getStr(*IDN,99) $(PORT) $(A)")
    field(PINI, "YES")
}

cat MDO.proto
Terminator = LF;
```

(continues on next page)

(continued from previous page)

```
getStr{
    out "$1?";
    in "%s";
    @replytimeout {out "$1?"; in "%s";}
}
```

Now, we add to `sampleIOCApp/Db/Makefile` the information that these files must be included in the compilation. So

```
cat sampleIOCApp/Db/Makefile
TOP=../..
include $(TOP)/configure/CONFIG
#----------------------------------------
# ADD MACRO DEFINITIONS BELOW HERE
# Install databases, templates & substitutions like this
DB += circle.db
DB += dbExample1.db
DB += dbExample2.db
DB += sampleIOCVersion.db
DB += dbSubExample.db
DB += user.substitutions
DB += MDO.db
DB += MDO.proto
# If .db template is not named *.template add
# _TEMPLATE =
include $(TOP)/configure/RULES
#----------------------------------------
# ADD EXTRA GNUMAKE RULES BELOW HERE
```

Again, `make` in directory sampleIOC. Finally, we add IP port configuration, setting the Stream path and loading the database to the `st.cmd` file. The `st.cmd` should read:

```
cat st.cmd

#!../../bin/darwin-x86/sampleIOC

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

< envPaths

epicsEnvSet ("STREAM_PROTOCOL_PATH","$(TOP)/db")

cd "${TOP}"

## Register all support components
dbLoadDatabase "dbd/sampleIOC.dbd"
sampleIOC_registerRecordDeviceDriver pdbbase

## Load record instances
dbLoadTemplate "db/user.substitutions"
dbLoadRecords "db/sampleIOCVersion.db", "user=UUUUUU"
dbLoadRecords "db/dbSubExample.db", "user=UUUUUU"

#IF if the user also defines EPICS_CAS_INTF_ADDR_LIST then beacon address
#list automatic configuration is constrained to the network interfaces specified
```

(continues on next page)

(continued from previous page)

```
#therein, and therefore only the broadcast addresses of the specified LAN interfaces,
#and the destination addresses of all specified point-to-point links, will be␣
↪automatically configured.
#epicsEnvSet ("EPICS_CAS_INTF_ADDR_LIST","aaa.aaa.aaa.aaa")

# connect to the device ... IP-Address ! Port 2025 used by textronix, see manual
drvAsynIPPortConfigure("L0","bbb.bbb.bbb.bbb:pppp",0,0,0)

## Load record instances
dbLoadRecords("db/MDO.db", "P=UUUUUU:,PORT=L0,R=MDO:,L=0,A=0")

#- 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=UUUUUU"
```

In here, you have to replace *UUUUUU* with the user name that runs the EPICS IOC (you?). *bbb.bbb.bbb.bbb* is the IP of the device (e.g. the scope) and *pppp* the port on which it listens. *EPICS_CAS_INTF_ADDR_LIST* can be used if there are two network interfaces (e.g. wlan and eth0).

The following commands might be necessary with multiple network interfaces:

```
export EPICS_CA_ADDR_LIST=ccc.ccc.ccc.ccc << Broadcast address of the network
export EPICS_CA_AUTO_ADDR_LIST=NO
```

CHAPTER 4

Installation on Windows

4.1 Introduction

We assume that you know more or less what EPICS is. You can get basic idea from https://epics-controls.org/about-epics/. Here we want to start from scratch on windows system and get to the point where we have a working server, then you get on other how-tos to take you further.

4.2 Prepare your system

You Need 'C++ Libraries', 'GNU make' and 'GCC' to compile from source. On Windows these dependencies can be installed by using msys2 tool. This tool is available windows 7 onwards only. Currently this procedure is verified on windows 8.1 (64 bit) and Windows 10 (64 bit). But, It should work for all the version of windows. In case we test it for other versions, We will update the document.

4.3 Install Tools

MSYS2 provides a bash shell, Autotools, revision control systems and the like for building native Windows applications using MinGW-w64 toolchains. Tool can be installed from official website <https://www.msys2.org>. Download and run the installer - "x86_64" for 64-bit, "i686" for 32-bit Windows. Currently we go for 64 bit system. Installation procedure is well explained on website.

Once installation in complete, you have three options available. Launch "MSYS MinGW 64-bit" option (MSYS2 and 32-bit option fails to compile EPICS). It shall provide you bash which resembles linux command shell. Update MSYS2 with following command

```
$ pacman -Syu
```

After finished Close the bash (do not exit). Open bash again and run the same command again to finish the updates.

`tar` is also needed to unpack the EPICS base

```
$ pacman -S tar
```

Install `perl`

```
$ pacman -S perl
```

Install `make`

```
$ pacman -S make
```

Install `mingw-gcc` for 64-bit environment

```
$ pacman -S mingw-w64-x86_64-gcc
resolving dependencies...
looking for conflicting packages...

Packages (14) mingw-w64-x86_64-binutils-2.32-3
              mingw-w64-x86_64-crt-git-7.0.0.5524.2346384e-1
              mingw-w64-x86_64-gcc-libs-9.2.0-2  mingw-w64-x86_64-gmp-6.1.2-1
              mingw-w64-x86_64-headers-git-7.0.0.5524.2346384e-1
              mingw-w64-x86_64-isl-0.21-1  mingw-w64-x86_64-libiconv-1.16-1
              mingw-w64-x86_64-libwinpthread-git-7.0.0.5522.977a9720-1
              mingw-w64-x86_64-mpc-1.1.0-1  mingw-w64-x86_64-mpfr-4.0.2-2
              mingw-w64-x86_64-windows-default-manifest-6.4-3
              mingw-w64-x86_64-winpthreads-git-7.0.0.5522.977a9720-1
              mingw-w64-x86_64-zlib-1.2.11-7  mingw-w64-x86_64-gcc-9.2.0-2

Total Download Size:    58.53 MiB
Total Installed Size:  428.12 MiB

:: Proceed with installation? [Y/n] y
```

Install `gcc`

```
$ pacman -S gcc
```

Check everything is installed properly,

```
$ pacman -Q make perl mingw-w64-x86_64-gcc gcc
make 4.2.1-1
perl 5.30.0-1
mingw-w64-x86_64-gcc 9.2.0-2
gcc 9.1.0-2
```

4.4 Install EPICS

```
$ cd $HOME
$ wget https://epics-controls.org/download/base/base-7.0.3.1.tar.gz
$ tar -xvf base-7.0.3.1.tar.gz
$ cd base-7.0.3.1
$ export EPICS_HOST_ARCH=windows-x64-mingw
$ make
```

There should be lots of warnings, but no error.

4.5 Test EPICS in Msys environment

Run `softIoc` and, if everything is ok, you should see an EPICS prompt. You need to provide whole path here, as newly executables is yet not recognised as commands by widnows. That is need to be set by windows "edit the system environment variables". After that it directly works as commands. Replace 'user' with actual windows user name folder existing in your windows installation.

```
$ /home/'user'/base-7.0.3.1/bin/windows-x64-mingw/softIoc
epics>
```

You can exit with ctrl-c or by typing exit.

Voilà.

Ok, now you know that EPICS is installed correctly.

4.6 Test EPICS in Windows

Exit or minimise Msys2 environment. Open windows command prompt. Here 'user' is windows-user/account folder name.

```
> cd c:\msys64\home\'user'\base-7.0.3.1\bin\windows-x64-mingw
> softIoc.exe -x test
    Starting iocInit
    ############################################################################
    ## EPICS R7.0.3.1
    ## EPICS Base built Apr 16 2020
    ############################################################################
    iocRun: All initialization complete
    epics>
```

Normal EPICS commands like caget, caput will still not work, as windows doesn't recognise them as valid commands. You have to add those paths in windows Environment Variable. Go to Start Manu, Type "environment" and select `Edit the system Environment Variables`.

1. Select `Advance` tab, navigate to `Environment Variables` button. That should open editable Tables of Path for Windows Environmet.
2. In `User Variable for 'user'` option, Press NEW
3. Add EPICS BASE path here. In `Variable Name`, Put "EPICS_BASE". In `Variable Path`, put `C:\msys64\home\'user'\base-7.0.3.1`
4. One more variable to describe host architecture. In `Variable Name`, put EPICS_HOST_ARCH. In `Variable Value`, put "windows-x64-mingw"
5. Now, Navigate to Variable called `Path`. Press Edit.
6. To add new Path for EPICS commands, Press New again and put `%EPICS_BASE%\bin\%EPICS_HOST_ARCH%`. Alternatively you can also put whole path as `C:\msys64\home\'user'\base-7.0.3.1\bin\windows-x64-mingw` Press ok two times and you are done.
7. Restart the Machine and check if `caget` and `camonitor` is being recognised as valid commands.

This should finish setting up EPICS environment in your windows machine. Let's test some basic commands and simple Process variable in windows `command prompt`.

prepare a file `test.db` in `C:\msys64\home\'user'\base-7.0.3.1\bin\windows-x64-mingw` that reads like,

```
record(ai, "temperature:water")
{
    field(DESC, "Water temperature in the fish tank")
}
```

This file defines a record instance called `temperature:water`, which is an analog input (ai) record. As you can imagine DESC stays for Description. Now we start softIoc again, but this time using this record database.

```
> cd cd c:\msys64\home\'user'\base-7.0.3.1\bin\windows-x64-mingw
> softIoc -d test.db
iocInit()
Starting iocInit
################################################################################
## EPICS R7.0.3.1
## EPICS Base built Apr 16 2020
################################################################################
iocRun: All initialization complete
```

Now, from your EPICS prompt, you can list the available records with the `dbl` command and you will see something like

```
epics> dbl
temperature:water
```

Open one more terminal (call it t2),

```
camonitor temperature:water
```

Open a new terminal (call it t3) and try to change value of PV using `caput`. you can also readback using `caget`.

```
>caput temperature:water 23
Old : temperature:water              0
New : temperature:water              23

>caput temperature:water 24
Old : temperature:water              23
New : temperature:water              24

>caput temperature:water 27
Old : temperature:water              24
New : temperature:water              27

>caput temperature:water 28.1
Old : temperature:water              27
New : temperature:water              28.1

>caget temperature:water
temperature:water              28.1
```

Monitor changes in terminal t2,

```
temperature:water              2020-04-22 17:52:58.752021 23
temperature:water              2020-04-22 17:53:03.008201 24
temperature:water              2020-04-22 17:53:06.053267 27
temperature:water              2020-04-22 17:53:09.003619 28.1
```

This concludes EPICS installation, Windows Environment variable settings and EPICS basic testing. We can `MSYS2` for building EPICS and IOCs. Files and EPICS executable created from that process can be run in windows environment using `command prompt`.

4.7 Create a demo/test ioc

All though `softIoc` can be used with multiple instances with different db files, you may need to create your own `ioc` for any number of reasons. We will create one test ioc from existing templates using `makeBaseApp.pl` script.

Let's create one IOC, which takes value of 2 process variables and add it and store it in 3rd process variable.

We will need `MSYS2` for building `ioc`. Open `MSYS2 Mingw 64-bit`. Go to EPICS base and create a new directory `testioc` below EPICS base.

```
$ cd /home/'user'/base-7.0.3.1/
$ mkdir testioc
$ cd testioc
```

from `testioc` folder run following,

```
$ ../bin/windows-x64-mingw/makeBaseApp.pl -t ioc test
$ ../bin/windows-x64-mingw/makeBaseApp.pl -i -t ioc test
Using target architecture windows-x64-mingw (only one available)
The following applications are available:
    test
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name?
```

Accept the default name and press enter. That should generate a skeleton for your `testioc`.

```
$ ls
configure  iocBoot  Makefile  testApp
```

Now create a `db` file which describes PVs for your `IOC`. Go to `testApp\db` and create `test.db` file with following record details.

```
record(ai, "test:pv1")
{
    field(VAL, 49)
}

record(ai, "test:pv2")
{
    field(VAL, 51)
}
record(calc,"test:add")
{
    field(SCAN,"1 second")
    field(INPA, "test:pv1")
    field(INPB, "test:pv2")
    field("CALC", "A + B")
}
```

Now open `Makefile` and navigate to,

```
#DB += xxx.db``
```

Remove # and change this to `test.db` ,

```
DB += test.db``
```

Go to back to root folder for IOC `testioc`. Go to `iocBoot\ioctest`. Modify `st.cmd` file.

Change

```
#dbLoadRecords("db/xxx.db","user=XXX")``
```

to

```
dbLoadRecords("db/test.db","user=XXX")``
```

Save all the files and go back to `MSYS2` terminal,

go to ioc root folder and run `make`,

```
$ cd /base-7.0.3.1/testioc
$ export EPICS_HOST_ARCH=windows-x64-mingw
$ make
```

It should create all the files required for test ioc,

```
$ ls
bin  configure  db  dbd  iocBoot  lib  Makefile  testApp
```

Now we go back to windows. Go to `\testioc\iocBoot\ioctest` . Open `envPaths` file and change relative paths to full paths

from,

```
epicsEnvSet("IOC","ioctest")
epicsEnvSet("TOP","/home/'user'/base-7.0.3.1/testioc")
epicsEnvSet("EPICS_BASE","/home/'user'/base-7.0.3.1/testioc/..")
```

to

```
epicsEnvSet("IOC","ioctest")
epicsEnvSet("TOP","C:/msys64/home/'user'/base-7.0.3.1/testioc")
epicsEnvSet("EPICS_BASE","C:/msys64/home/'user'/base-7.0.3.1")
```

Save file.

go back to windows `command prompt`,

```
> cd C:\msys64\home\'user'\base-7.0.3.1\testioc\iocBoot\ioctest
> C:\msys64\home\'user'\base-7.0.3.1\testioc\iocBoot\ioctest>..\..\bin\windows-x64-
↪mingw\test.exe st.cmd
#!../../bin/windows-x64-mingw/test
< envPaths
epicsEnvSet("IOC","ioctest")
epicsEnvSet("TOP","C:/msys64/home/'user'/base-7.0.3.1/testioc")
epicsEnvSet("EPICS_BASE","C:/msys64/home/'user'/base-7.0.3.1")
cd "C:/msys64/home/'user'/base-7.0.3.1/testioc"
## Register all support components
```

(continues on next page)

(continued from previous page)

```
dbLoadDatabase "dbd/test.dbd"
test_registerRecordDeviceDriver pdbbase
Warning: IOC is booting with TOP = "C:/msys64/home/'user'/base-7.0.3.1/testioc"
         but was built with TOP = "/home/'user'/base-7.0.3.1/testioc"
## Load record instances
dbLoadRecords("db/test.db","user='user'")
cd "C:/msys64/home/'user'/base-7.0.3.1/testioc/iocBoot/ioctest"
iocInit
Starting iocInit
############################################################################
## EPICS R7.0.3.1
## EPICS Base built Apr 16 2020
############################################################################
iocRun: All initialization complete
## Start any sequence programs
#seq sncxxx,"user='user'"
epics>
```

Check if database `test.db` you created is loaded correctly

```
epics> dbl
test:add
test:pv1
test:pv2
```

Open other `commad prompt` (call it t2) for monitoring `test:add`. type “camonitor test:add”

```
> camonitor test:add
> test:add                       2020-04-22 18:47:59.692169 100
```

Open one `command prompt` (call it t3). using caput modify values of `test:pv1` and `test:pv2`. You shall see changes in terminal t2 accordingly

CHAPTER 5

How To Port EPICS to a new OS/Architecture

This isn't a detailed list of tasks, but is intended to show the main stages needed to add a new build architecture to EPICS. If you make use of this and find there are hints you'd like to suggest, or steps missing please add them.

- Download a tarfile for the latest release of EPICS Base, or the snapshot from the R3.14 branch (not the trunk), and unpack it.
- If you're not already familiar with EPICS, at least skim chapter 4 of the IOC Application Developers Guide (hereafter known as the AppDevGuide; our build system is different to the usual "./configure && make" approach.
- Build your <base> on a linux-x86 or solaris-sparc system so you know what a fully built system actually looks and acts like. You can build multiple architectures simultaneously in the same tree, which makes for easier comparisons. On linux the build instructions should be as simple as

```
export EPICS_HOST_ARCH=linux-x86
cd <base>
make
```

- On the new system system, setenv EPICS_HOST_ARCH to the name for your new architecture, which usually takes the form <osname>-<cpufamily>, for example solaris-sparc, linux-x86, windows-x86
- In the <base>/configure/os directory, create these files by copying and editing the files from an existing architecture:

```
CONFIG.Common.<arch>
CONFIG.<arch>.Common
CONFIG_SITE.<arch>.Common
```

- I would suggest looking at the darwin-ppc or linux-x86 versions to start with; for a Unix-like OS you should be able to make use of the UnixCommon and/or GnuCommon files to provide most of the definitions and rules.
- If you have to cross-compile then there's more work you have to do and these instructions are probably not sufficient to get you there.

CHAPTER 6

Common Database patterns

6.1 Pull Alarm Status w/o Data

This is useful to bring alarm status in without affecting the value stored in a record. In the following example the alarm status of **$(P):set** is fetched by **$(P):rbv** when it is processed, but the value is read from a different record.

```
record(bo, "$(P):set") {
    field(OSEV, "MAJOR")
    field(FLNK, "$(P):rbv")
}
```

```
record(bi, "$(P):rbv") {
    field(SDIS, "$(P):set NPP MS")
    field(DISV, "-1")
    field(INP , "$(P):some:other:record")
}
```

6.2 Combined Setting and Readback

Use when a single PV is desired. Could be used to show the results of rounding in a float to fixed precision conversion.

In the following example the value read from a 'bi' is inverted so that the associated 'bo' acts as a toggle.

```
record(bi, "$(P):rbv") {
    field(DTYP, "...")
    field(INP , "...")
    field(PINI, "YES")
    field(FLNK, "$(P):inv")
}
```

```
record(calcout, "$(P):inv")
    field(CALC, "!A")
    field(INPA, "$(P):rbv")
    field(OUT , "$(P) NPP")
}
```

```
record(bo, "$(P)") {
    field(DTYP, "...")
    field(OUT , "...")
    field(FLNK, "$(P):rbv")
}
```

The important point is the NPP modifier on output link of the 'calcout' record. This updates the VAL field of the 'bo' record (and posts monitors) without processing it.

CHAPTER 7

How to run an EPICS Collaboration Meeting

This page is intended for "lessons learned" by sites who have run collaboration meetings, as hints to help future meetings run smoothly.

7.1 Organization

The EPICS Council now decides where meetings will be held, so there are usually 2 meetings a year, circulating between the Americas, Europe and the Asia/Pacific region. Future meeting locations that have already been fixed are usually listed here.

Collaboration meetings are usually 3 days long from Tuesday to Thursday or Wednesday to Friday, allowing for training and smaller working groups on the other days of the week. You'll need a big room for the full meeting, and some smaller meeting rooms for any workshops and training sessions you host.

Also somewhere near the refreshment location a room for any exhibitions; past meetings have succeeded in attracting companies who will a small sum to put up a display table at these meetings. Providing large-screen TVs or projectors in the exhibition space allows EPICS-related projects to demonstrate their software.

Recent attendance has been between 80-130 people, but this depends on the number of users from the hosting site and nearby. It can be difficult for government-funded workers (especially from US Department of Energy labs) to get approval for travel to exotic locations though, so expect attendance to vary. Workshops can be around 30 people, but may vary depending on the topic. If someone wants to run a workshop you can ask them for estimates on numbers.

To include some hands-on training you may may provide PCs capable of running a virtual machine for students to use, or ask them to bring their own laptops (which some will do anyway). Training is still possible without these, but would consist of basic lectures and demo's only. You would need to decide what kind of training you want, and organize some people in the community to give it.

Topics for workshops should generally be aligned with interests of the hosts. It will take support on the host side to make sure there is sufficient interest and attendance. Subgroups of developers such as the EPICS Core, CS-Studio and AreaDetector groups may ask to hold a private developers meeting adjacent to the main meeting, which will usually require providing a 10-15 seat meeting room with WiFi for each group for the period of their meeting.

7.2 Communications

If possible, start creating a website for the meeting before it is first announced to on tech-talk; some people will want to be able to find out more about the location when they first hear about it, so the announcement should have a link to that website.

Attendees from some countries such as China may have to navigate a quite long approvals process (internal, government and visa) to be able to attend, and may need a letter of invitation from you to get those approvals. Provide enough information so they know who to ask well in advance (3-4 months or more is advisable).

Eventually you'll most probably need online registration of attendees and possibly speakers, as well as to provide information about local hotels, transportation to/from nearby airports, and local tourist agencies for attendees' partners. Arranging a "partner programme" is usually unnecessary.

7.3 Facilities

For the main meeting, a hall with LCD projector(s) for computer connection and a PC to display the presentation files. Most people will bring talks in MS PowerPoint, Adobe PDF and/or LibreOffice Impress formats (installing LibreOffice on this machine is advised, but not essential if you give people notice). It saves time and confusion at these international meetings if the PC can be configured with English language settings (Windows menus etc.). Providing a remote control for the presentation program and a laser pointer is helpful.

Some presenters may want to use their own laptops for live demonstrations, or if they're using a less common presentation program (e.g. Apple's Keynote).

In a large hall speakers should use a microphone if a PA is available; a radio-microphone is preferred.

The main hall (and ideally the other meeting rooms too) should have reliable, high-bandwidth WiFi internet access since most people will bring a laptop or notebook PC and want to connect up during the meeting.

Laptop batteries don't last more than a few hours, so there will be demand for power sockets in the hall too. If these are not available at every seat, providing extension leads spread about the room is generally a good idea and is much appreciated by attendees.

Refreshments should be available at the breaks mid-morning and mid-afternoon. If your institution can't afford to fund these itself it is acceptable to charge attendees a fee at registration to pay for them (exhibitors funds are also helpful here). People will expect to pay for their own lunches, and also to pay to attend the conference dinner which is usually held on the evening before the last day.

7.4 Agenda

Setting an initial program structure gives presenters an idea of what topics the hosting institution may be particularly interested in, but isn't necessary.

In recent years the hosting site has been responsible for soliciting and collecting speakers names and talk titles; an Indico website can perform much of the clerical work involved automatically, but the talks will still have to be manual scheduled into the Agenda. Individual talks are usually allotted 15 minutes plus 5 minutes for questions, but some topics may be given additional time. Recent meetings have also introduced "Lightning talks" which are 5 minutes long with no question time, and these have proven popular and a good way to cover many topics in a short time.

The community will usually submit a number of submissions of presentations, but the hosts should expect to have to do some additional solicitation to fill out the program. This can (should) be informed by topics the host institute is interested in. Communications to the whole community about submitting talks should be sent to tech-talk.

CHAPTER 8

How to avoid copying arrays with waveformRecord

This page describes how to use the array field memory management feature to be introduced in EPICS 3.15.1 (not yet released). This allows array data to be moved into and out of the value (aka BPTR) field of the waveform, aai, and aao types.

Making use of this feature involves replacing the pointer stored in the BPTR field with another (user allocated) pointer. The basic rules are:

1. BPTR, and the memory it is currently pointing to, can only be accessed while the record is locked.
2. NELM may not be changed.
3. BPTR must always point to a piece of memory large enough to accommodate the maximum number of elements (as given by the NELM field).

Rule #1 means that it is only safe to read, write, or de-reference the BPTR field from a device support function, or after manually calling dbScanLock(). Rule #3 means that BPTR can never be set to NULL, and when replacing BPTR, the replacement must be allocated large enough for the worst case. An external client may put an array of up to NELM elements to the field at almost any time.

8.1 Example

```
/* Demonstration of using custom allocation for waveformRecord buffers.
 *
 * Requires EPICS Base with the array field memory management patch
 * https://code.launchpad.net/~epics-core/epics-base/array-opt
 *
 * This example makes inefficient use of malloc() and
 * free().  This is done to make clear where new memory appears.
 * In reality a free list should be used.
 *
 * Also be aware that this example will use 100% of the time of one CPU core.
 * However, this will be spread across available cores.
 *
```

(continues on next page)

(continued from previous page)

```
 * To use this example include the following in a DBD file:
 *
 * device(waveform,CONSTANT,devWfZeroCopy,"Zero Copy Demo")
 *
 * Also include a record instance
 *
 * record(waveform, "$(NAME)") {
 *  field(DTYP, "Zero Copy Demo")
 *  field(FTVL, "SHORT")
 *  field(NELM, "100")
 *  field(SCAN, "I/O Intr")
 * }
 */

#include <errlog.h>
#include <initHooks.h>
#include <ellLib.h>
#include <devSup.h>
#include <dbDefs.h>
#include <dbAccess.h>
#include <cantProceed.h>
#include <epicsTypes.h>
#include <epicsMutex.h>
#include <epicsEvent.h>
#include <epicsThread.h>
#include <menuFtype.h>
#include <dbScan.h>

#include <waveformRecord.h>

static ELLLIST allPvt = ELLLIST_INIT;

struct devicePvt {
    ELLNODE node;

    /* synchronize access to this structure */
    epicsMutexId lock;
    /* wakeup the worker when another update is needed */
    epicsEventId wakeup;
    /* notify the scanner thread when another update is available */
    IOSCANPVT scan;

    /* the next update */
    void *nextBuffer;
    epicsUInt32 maxbytes, numbytes;
};

static void startWorkers(initHookState);

static long init(int phase)
{
    if(phase!=0)
        return 0;
    initHookRegister(&startWorkers);
    return 0;
}
```

(continues on next page)

(continued from previous page)

```
static long init_record(waveformRecord *prec)
{
    struct devicePvt *priv;
    if(prec->ftvl!=menuFtypeSHORT) {
        errlogPrintf("%s.FTVL must be set to SHORT for this example\n", prec->name);
        return 0;
    }

    /* cleanup array allocated by record support.
        * Not necessary since we use calloc()/free(),
        * but needed when allocating in other ways.
        */
    free(prec->bptr);
    prec->bptr = callocMustSucceed(prec->nelm, dbValueSize(prec->ftvl), "first buf");

    priv = callocMustSucceed(1, sizeof(*priv), "init_record devWfZeroCopy");
    priv->lock = epicsMutexMustCreate();
    priv->wakeup = epicsEventMustCreate(epicsEventFull);
    scanIoInit(&priv->scan);
    priv->maxbytes = prec->nelm*dbValueSize(prec->ftvl);

    ellAdd(&allPvt, &priv->node);

    prec->dpvt = priv;
    return 0;
}

static void worker(void*);

static void startWorkers(initHookState state)
{
    ELLNODE *cur;
    /* Don't start worker threads until
        * it is safe to call scanIoRequest()
        */
    if(state!=initHookAfterInterruptAccept)
        return;
    for(cur=ellFirst(&allPvt); cur; cur=ellNext(cur))
    {
        struct devicePvt *priv = CONTAINER(cur, struct devicePvt, node);
        epicsThreadMustCreate("wfworker",
                              epicsThreadPriorityHigh,
                              epicsThreadGetStackSize(epicsThreadStackSmall),
                              &worker, priv);
    }
}

static void worker(void* raw)
{
    struct devicePvt *priv=raw;
    void *buf = NULL;
    epicsUInt32 nbytes = priv->maxbytes;

    while(1) {

        if(!buf) {
            /* allocate and initialize a new buffer for later (local) use */
```

(continues on next page)

(continued from previous page)

```
            size_t i;
            epicsInt16 *ibuf;
            buf = callocMustSucceed(1, nbytes, "buffer");
            ibuf = (epicsInt16*)buf;
            for(i=0; i<nbytes/2; i++)
            {
                ibuf[i] = rand();
            }
        }

        /* wait for Event signal when record is scanning 'I/O Intr',
            * and timeout when record is scanning periodic
            */
        if(epicsEventWaitWithTimeout(priv->wakeup, 1.0)==epicsEventError) {
            cantProceed("worker encountered an error waiting for wakeup\n");
        }

        epicsMutexMustLock(priv->lock);

        if(!priv->nextBuffer) {
            /* make the local buffer available to the read_wf function */
            priv->nextBuffer = buf;
            buf = NULL;
            priv->numbytes = priv->maxbytes;
            scanIoRequest(priv->scan);
        }

        epicsMutexUnlock(priv->lock);
    }
}

static long get_iointr_info(int dir, dbCommon *prec, IOSCANPVT *scan)
{
    struct devicePvt *priv=prec->dpvt;
    if(!priv)
        return 0;
    *scan = priv->scan;
    /* wakeup the worker when this thread is placed in the I/O scan list */
    if(dir==0)
        epicsEventSignal(priv->wakeup);
    return 0;
}

static long read_wf(waveformRecord *prec)
{
    struct devicePvt *priv=prec->dpvt;
    if(!priv)
        return 0;

    epicsMutexMustLock(priv->lock);

    if(priv->nextBuffer) {
        /* an update is available, so claim it. */

        if(prec->bptr)
            free(prec->bptr);
```

(continues on next page)

(continued from previous page)

```
        prec->bptr = priv->nextBuffer; /* no memcpy! */
        priv->nextBuffer = NULL;
        prec->nord = priv->numbytes / dbValueSize(prec->ftvl);

        epicsEventSignal(priv->wakeup);
    }

    epicsMutexUnlock(priv->lock);

    assert(prec->bptr);

    return 0;
}

static
struct dset5 {
    dset com;
    DEVSUPFUN read;
} devWfZeroCopy = {
{5, NULL,
    &init,
    &init_record,
    &get_iointr_info
},
    &read_wf
};

#include <epicsExport.h>

epicsExportAddress(dset, devWfZeroCopy);
```

CHAPTER 9

How to find which IOC provides a PV

This process is for IOCs running on Linux servers.

9.1 Find Host and TCP port

The `cainfo` command will tell you which host is serving a particular PV, and which TCP port number on that host is used.

```
$ cainfo LN-TS{EVR:1A-SFP}Pwr:RX-I
LN-TS{EVR:1A-SFP}Pwr:RX-I
State:            connected
Host:             10.0.152.111:5064
Access:           read, write
Native data type: DBF_DOUBLE
Request type:     DBR_DOUBLE
Element count:    1
```

Here we see that the PV "LN-TS{EVR:1A-SFP}Pwr:RX-I" is served from port number 5064 of 10.0.152.111.

```
$ cainfo LN-RF{AMP:1}Amp-Sts
LN-RF{AMP:1}Amp-Sts
State:            connected
Host:             linacioc01.cs.nsls2.local:36349
Access:           read, write
Native data type: DBF_ENUM
Request type:     DBR_ENUM
Element count:    1
```

Here is another example where the hostname is shown instead of an IP address. Also this server has more than one IOC, and the one in question is using port 36349.

9.2 Find which process is using a TCP port (Linux only)

Super-user (root) permission is required to find which Linux process is bound to a particular TCP port.

To continue the example from above. On the server linacioc01.cs.nsls2.local we run:

```
$ sudo netstat -tlpn | grep 36349
tcp        0      0 0.0.0.0:36349           0.0.0.0:*               LISTEN      4627/
↪s7ioc
```

This tells us that TCP port 36349 is bound by process ID (PID) 4627, which has the process name of 's7ioc'.

9.3 Find information about a process (Linux only)

The `ps` command can give some information, including the command used to start the process. This often contains enough information to identify where the IOC's files can be found.

```
$ ps aux|grep 4627
softioc   4627  1.5  0.0  93748  6616 pts/23   Ssl+ Jan07 744:18 ../../bin/linux-x86/
↪s7ioc /epics/iocs/RF-CONTROL/iocBoot/iocrf-control/st.cmd
```

There are several pieces of information available under `/proc` which are useful. The entry `/proc/<pid>/cwd` is a symbolic link to the current working directory of the process. There is also `/proc/<pid>/exe` which links to the executable.

```
$ sudo ls -l /proc/4627/cwd
lrwxrwxrwx 1 softioc softioc 0 Feb 10 11:49 /proc/4627/cwd -> /epics/iocs/RF-CONTROL
$ sudo ls -l /proc/4627/exe
lrwxrwxrwx 1 softioc softioc 0 Jan  7 09:58 /proc/4627/exe -> /epics/iocs/RF-CONTROL/
↪bin/linux-x86/s7ioc
```

9.4 Additional: Finding the procServ/screen running an IOC (Linux only)

The `ps` command can also tell us the PID of the parent of the IOC process. The techniques of step 3 can also be applied to the parent.

```
$ ps -eo pid,ppid,user,cmd|grep 4627
4627  4566 softioc  ../../bin/linux-x86/s7ioc /epics/iocs/RF-CONTROL/iocBoot/iocrf-
↪control/st.cmd
```

The parent PID in the second column is 4566.

```
$ ps aux|grep 4566
softioc   4566  0.0  0.0   3452   592 ?        Ss   Jan07   2:18 /usr/bin/procServ -q␣
↪-c /epics/iocs/RF-CONTROL/iocBoot/iocrf-control -i ^D^C^] -p /var/run/softioc-RF-
↪CONTROL.pid -n RF-CONTROL --restrict --logfile=/var/log/softioc-RF-CONTROL.log 4057␣
↪/epics/iocs/RF-CONTROL/iocBoot/iocrf-control/st.cmd
```

And to complete the circle, and get access to the IOC console, we find which TCP port this procServ instance is bound to.

```
$ sudo netstat -tlpn|grep 4566
tcp        0      0 127.0.0.1:4057          0.0.0.0:*               LISTEN      4566/
↪procServ
$ telnet localhost 4057
epics> dbpr LN-RF{AMP:1}Amp-Sts
ASG:                DESC: Ampl.500 MHz E-Source             DISA: 0
DISP: 0             DISV: 1             NAME: LN-RF{AMP:1}Amp-Sts
RVAL: 16            SEVR: NO_ALARM      STAT: NO_ALARM      SVAL: 0
TPRO: 0             VAL: 1
```