areaDetector plugin lab handout

Mark Rivers, University of Chicago

January 30, 2015
Advanced Photon Source
Room E1100/1200

Overview

This lab is intended to introduce writing an areaDetector plugin using the NDPluginDriver C++ base class. The plugin computes the power spectrum of the 2-D FFT of the input array.

Setup

The development can either be done locally on a student's laptop or removely on our Linux machine called corvette.cars.aps.anl.gov.

For working on corvette each student has his/her own subdirectory where they will be working, /home/epics_class/student1, /home/epics_class/student2, etc. The assignment of student numbers will be done during the class.

You should bring a laptop to the lab with the following capabilities:

- X11 server
- ssh client configured to tunnel X11

The ssh/X11 connection should be tested prior to the class by logging into corvette using the credentials above and typing the command "xclock &" or "medm &". If you see the application then things are configured correctly.

Linux host: corvette.cars.aps.anl.gov
Username: (given out in class)
Password: (given out in class)

Copy and build the areaDetector/ADCore source code

Once you are logged in go to the appropriate subdirectory, e.g. student1.

```
$ cd student1
```

Copy the areaDetector source code for the lab from the /home/epics_class/teacher directory to this directory:

```
$ cp -rp ../teacher/areaDetector .
```

Change to the areaDetector/configure directory:

```
$ cd areaDetector/configure
```

Edit RELEASE_PATHS.local and change the location of areaDetector, changing this line:

```
AREA_DETECTOR= /corvette/home/epics_class/teacher/areaDetector
```

to

```
AREA_DETECTOR= /corvette/home/epics_class/student1/areaDetector
```

Replacing student1 with your student number.

Change to the areaDetector/ADCore directory.

```
cd ../ADCore
```

Clean the source:

```
$ make –sj clean uninstall
```

Build the source:

```
$ make –sj
```

The –s flag means that make runs silently, so you only see errors and warnings. –j means that the make is run in parallel, doing as many tasks as possible at the same time.

Customize the setup for your student number

All of the soft IOCs are running on the same machine and subnet, so each soft IOC must use a different PV prefix. It is very important that you change your setup to use the appropriate PV prefix. This is done as follows:

Change to the ADCore/iocs/simDetectorIOC/iocBoot/iocSimDetector directory:

```
$ cd iocs/simDetectorIOC/iocBoot/iocSimDetector/
```

Edit the file st.cmd and change PREFIX from SIM_1: to SIM_N:, where N is your student number, i.e. SIM_2: for student2, etc.

NDPluginFFT

You will be creating a plugin that computes the 2-D FFT power spectrum of the input array

It will implement the following 3 new parameters
OutputDataType : the data type of the output array, including Automatic (same as input)
ScaleFactor : A scale factor for the power spectrum, useful for integer output data types
FFTDirection: Forward (spatial to frequency), Reverse (frequency to spatial)

EPICS database
The database file for the plugin is in `ADCore/ADApp/Db/NDFFT.template.  It contains only the records for the 3 parameters above.`

Plugin source code
The source code for the driver is in ADCore/ADApp/pluginSrc. There are two versions of the plugin in that location: NDPluginFFT.cpp and NDPluginFFTTemplate.cpp. NDPluginFFT.cpp is a fully functional plugin, and so is an example to use. NDPluginFFTTemplate.cpp is a skeleton plugin. It contains just enough code to allow the plugin to run and for the EPICS records to connect to the plugin, but it does nothing useful. Edit the Makefile in this directory to control which version of the driver you compile.

Each student may want to approach the lab differently, depending on experience and goals. One approach would be to first start with the fully developed driver, and run the MEDM display to understand how the driver and device work.

Start the MEDM displays

```
$ medm -x –macro “P=SIM_1:, R=cam1:” simDetector.adl &
$ medm -x –macro “P=SIM_1:, R=FFT1:” NDFFT.adl &
```

The fields will all be white until you start your IOC.

Start the IOC

```
$ cd iocs/simDetectorIOC/iocBoot/iocSimDetector/
$ ../../bin/linux-x86_64/simDetectorApp st.cmd
```

Building a real driver from the skeleton driver

Once you have worked with the functional plugin, switch to building and running the skeleton plugin. Initially it will not do anything. Try adding the following features one at a time. You can try to write them yourself or copy the code from the functional driver. Make sure you understand each step, and ask questions if you don’t!