areaDetector

Writing Drivers

Mark Rivers

University of Chicago

Advanced Photon Source

Writing an areaDetector Driver

- Study the vendor API documentation
- Run the camera with the vendor GUI to understand how it works
- Does the vendor provide a simulation camera? If so you can start development before the hardware arrives
- Find an existing areaDetector driver to use as a model. Spend some time to figure out which existing driver is most similar to the one you will be writing
- Determine what features are most important to implement first
- Start with a simple version of the driver, add complexity one piece at a time

Detector Drivers with C/C++ Vendor Library

Driver	C/C++ Library	OS	Interface
ADsimDetector	Internal	All	Internal
ADProsilica	PvAPI	Win32, Linux, OSX	GigE
ADPointGrey	FlyCap2	Win32, Linux	GigE, Firewire, USB-3.0
aravisGigE	Aravis	Linux	GigE
ADPerkinElmer	XISL	Win32	Proprietary
ADAndor	Andor SDK2	Win32, Linux	USB, PCI
ADAndor3	Andor SDK3	Win32, Linux	PCI to CameraLink
ADADSC	Detcon	Linux	PCI?
ADFireWireWin	Carnegie Mellon	Win32	Firewire
firewireDCAM	dc1394	Linux	Firewire
ADPvCAM	PvCam	Win32, Linux	PCI, USB
ADQImaging	QCamDriver	Win32	USB, Firewire, other

Detector Drivers with Socket Server

Driver	Socket Server	OS	Interface
ADPilatus	camserver	Linux (all)	Proprietary
ADmarCCD	marccd	Linux (all)	Proprietary
ADmar345	mar345dtb	Linux (all)	Ethernet
ADBruker	BIS	Win32 (all)	USB, other?
ADPixirad	In camera	Win32 (all)	Ethernet
ADPSL	PSLViewer	Win32 (all)	USB, GigE, other?

Detector Drivers with GUI Application Automation

Driver	Application	OS	Interface
ADRoper	WinView	Win32	PCI, USB, other?
ADLightField	LightField	Win32	USB, GigE, other?

Guidelines and Rules for Writing an areaDetector Driver

- Drivers will generally implement one or more of the writeInt32(), writeFloat64() or writeOctet() functions if they need to act immediately on a new value of a parameter.
  - For many parameters it is normally sufficient to simply have them written to the parameter library, and not to handle them in the writeXXX() functions. The parameters are then retrieved from the parameter library with the getIntParam(), getDoubleParam(), or getStringParam() function calls when they are needed.
- If the writeInt32(), writeFloat64() or writeOctet() functions are implemented they **must** call the base class function for parameters that they do not handle and whose parameter index value is less than the first parameter of this class, i.e. parameters that belong to a base class.
- Drivers will need to call the createParam() function in their constructor if they have additional parameters beyond those in the asynPortDriver or ADDriver base classes.

Guidelines and Rules for Writing an areaDetector Driver

- Drivers will generally need to create a new thread in which they run the acquisition task.
  - Some vendor libraries create such a thread themselves, and then the driver must just implement a callback function that runs in that thread (the Prosilica and Perkin Elmer are examples of such drivers).
- The acquisition thread will typically monitor the acquisition process and perform periodic status update callbacks.
  - The details of how to implement this will vary depending on the specifics of the vendor API. There are many existing detector drivers that can be used as examples of how to write a new driver.
- If the detector hardware does not support fixed-period acquisition or muliple-image acquisition sequence (ADNumImages parameter) then these should be emulated in the driver.
  - The simDetector, marCCD and other drivers can be used as examples of how to do this.

Guidelines and Rules for Writing an areaDetector Driver

- If NDArrayCallbacks is non-zero then drivers must do the following:
  - Set the uniqueID in the NDArray
  - Set the timeStamp (double) in the NDArray
  - Set the epicsTS (epicsTimeStamp) in the NDArray
  - Call asynNDArrayDriver::getAttributes to attach any attributes defined for this driver to the current array.
  - Call doCallbacksGenericPointer() so that registered clients can get the values of the new arrays. Drivers must release their mutex by calling this->unlock() before they call doCallbacksGenericPointer(), or a deadlock can occur if the plugin makes a call to one of the driver functions.

Driver examples

- Look in detail at 3 drivers
- simDetector
  - Implements most features in simulation
- Mar345
  - Relatively simple socket server
- Perkin Elmer
  - Vendor C API