

Communicating with a GPIB device through a GPIB/LAN adapter and an ASYN record

Step-by-step instructions for setting up simple device control using only an ASYN record.

1. Create a new application

```
mkdir serialTest
cd serialTest
~/src/EPICS/base_R3_14_2_branch/bin/darwin-x86/makeBaseApp.pl -l
~/src/EPICS/base_R3_14_2_branch/bin/darwin-x86/makeBaseApp.pl -t ioc serialTest
~/src/EPICS/base_R3_14_2_branch/bin/darwin-x86/makeBaseApp.pl -i -t ioc serialTest
```

2. Add ASYN support

Edit configure/RELEASE and add a line specifying the path to your ASYN installation
ASYN=/home/phoebus/NORUME/src/EPICS/modules/soft/asyn

Edit serialTestApp/src/Makefile and add two lines. It should be apparent from the template where these lines are to be placed:

```
serialTest_DBD += drvVx111.dbd
serialTest_LIBS += asyn
```

Edit serialTestApp/Db/Makefile and add a line:

```
DB_INSTALLS += $(ASYN)/db/asynRecord.db
```

Edit iocBoot/iocSerialTest/st.cmd and add lines to configure the GPIB/LAN adapter and to load an ASYN record. Here's a complete st.cmd file showing how things should look when you're finished. You will, of course, have to substitute the IP address of your GPIB/LAN adapter and the address of your GPIB device. You'll likely want to choose a different value for the PV name prefix macro (P) as well. The IMAX and OMAX values should be large enough to handle the longest messages you expect.

```
#!../bin/darwin-x86/serialTest
< envPaths
cd ${TOP}
## Register all support components
dbLoadDatabase("dbd/serialTest.dbd",0,0)
serialTest_registerRecordDeviceDriver(pdbbase)
## Configure devices
vx111Configure("L0","164.54.8.129",0,0.0,"gpib0",0,0)
## Load record instances
dbLoadRecords("db/asynRecord.db","P=normum:,R=asyn,PORT=L0,ADDR=24,IMAX=100,OMAX=100")
cd ${TOP}/iocBoot/${IOC}
iocInit()
```

3. Build the application (run make from the application top directory).

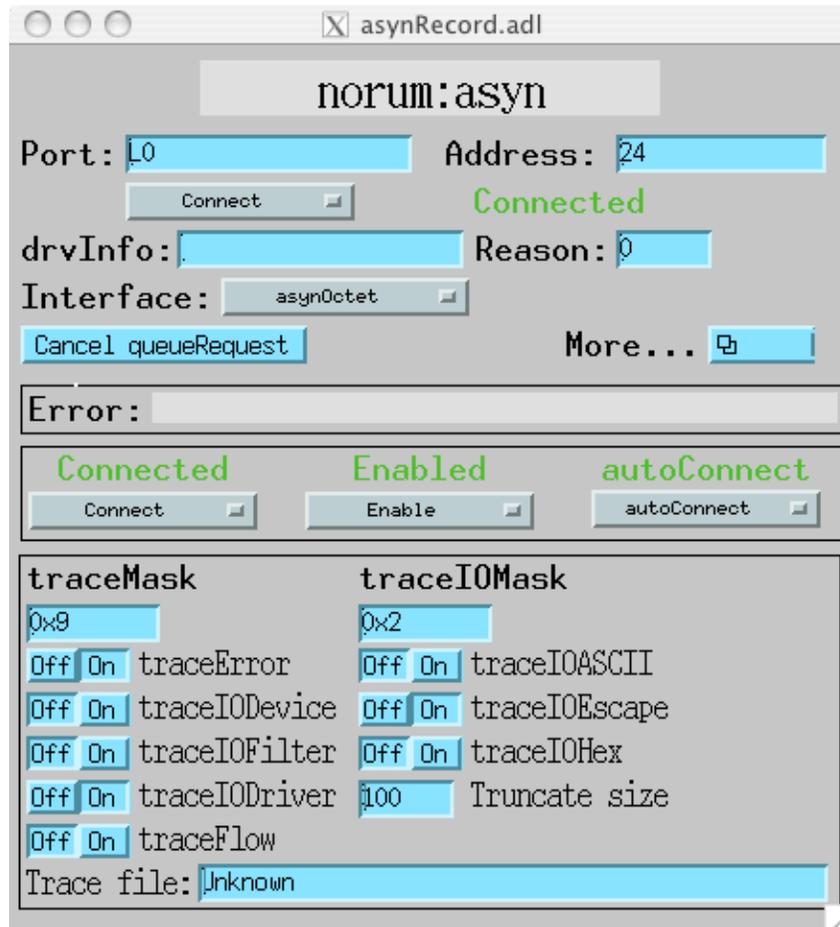
4. Start the IOC

```
cd iocBoot/iocserialTest
../bin/darwin-x86/serialTest st.cmd
```

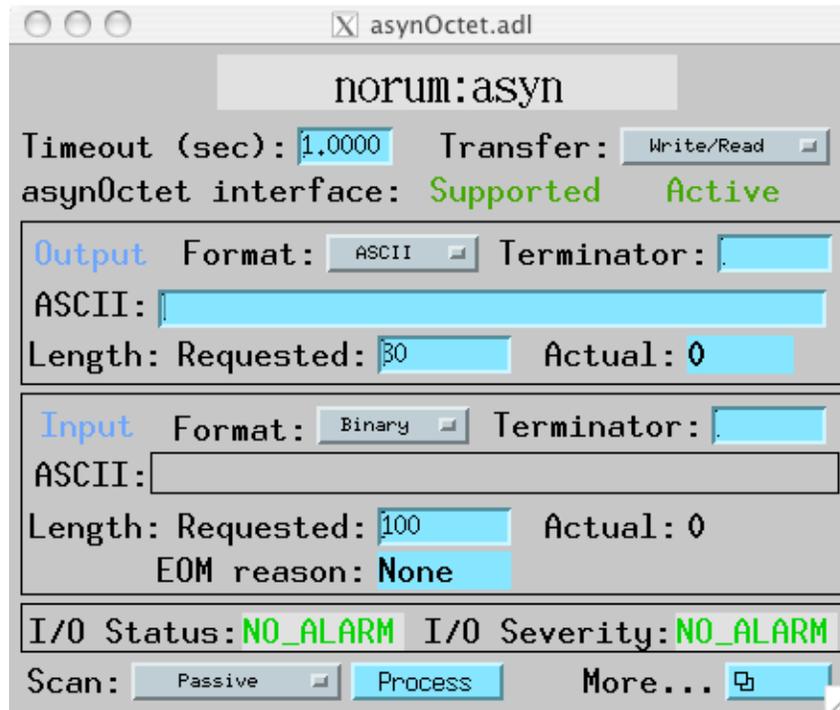
5. From another window, start MEDM. Make sure that the P and R macro values match those from st.cmd.

```
medm -x -macro "P=normum:,R=asyn" ~/src/EPICS/modules/soft/asyn/medm/asynRecord.adl
```

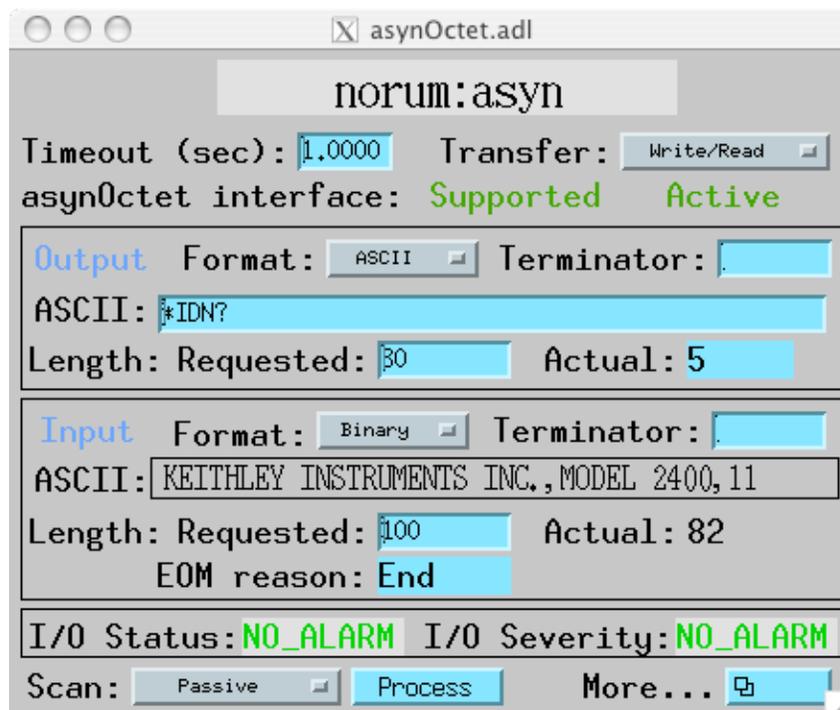
You should see something like the window shown below. I've made a few changes to the original values including increasing the trace I/O truncate size to 100 characters, enabling the traceIOEscape display and turning on traceIODriver debugging.



Click on the 'More...' button and bring up the "asynOctet Interface I/O" window. I've made some changes here as well – I've selected Binary input format and increased the requested input length to 100. If your device messages are 40 characters or less you don't have to make these changes.



Try entering some commands. A good one to start with is the SCPI Device Identification (*IDN?) command. You can see why I had to arrange for reply messages longer than the default 40 characters!



The readback display is truncated. Since I have traceIODriver enabled I can see the entire message in the IOC console window:

```
2007/09/20 09:01:15.998 L0 24 vxiWrite
*IDN?
2007/09/20 09:01:16.009 L0 24 vxiRead
```

6. If your requirements are modest, you might be done. The ASYN record may be adequate for your application. If not, you should investigate the Streams package or devGpib for writing support for your instrument.