lvPortDriver

a platform independent interface between LabVIEW software and EPICS device support

S. A. Baily

EPICS Collaboration Meeting

September 2016

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

GOAL: Platform Independent Interface for LabVIEW code running on an IOC

- **Currently many solutions**
  - Cosylab “shared memory” (cRIO - VxWorks)
  - SNS Shared memory (Windows)
  - Several solutions that allow LabVIEW to communicate via Channel access
- **Need a solution for Linux RT (cRIO)**
- **Solution should be useable everywhere.**
  - All operating systems that support EPICS and LabVIEW
  - Feature-rich enough to replace all existing methods.

What Is lvPortDriver?

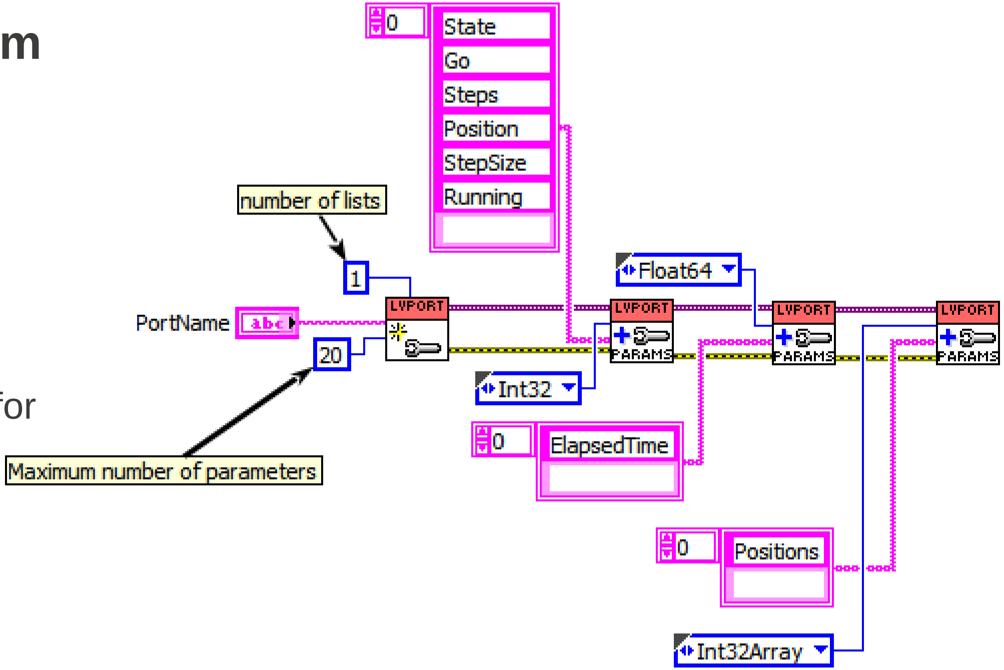
- **A C++ class and LabVIEW virtual library that allows one to write an asynPortDriver entirely with LabVIEW code.**
- **The IOC (built as a shared library) is then started by the LabVIEW code**
  - This means the EPICS IOC and the LabVIEW program run as the same process on any operating system, and don’t need to setup operating system specific shared memory.

Features

- **Currently Supported.**
  - All standard data types including Arrays.
  - Setting/getting asynStatus for a parameter.
  - LabVIEW subscribe as an asynclient to receive user events when parameters change.
    - Events include value and status.
- **Not Yet.**
  - Area Detector
    - Generic pointer is supported by the C++ class, but no LabVIEW Vis have been written.
  - Asynchronous record processing.
    - Write functions just store the value in the parameter library
    - It would be possible to do this differently, but I'm not sure how helpful this is.
  - Non-default GetBounds support
    - Could be implemented, but why not just use SLOPE conversions?
  - Waveform of Strings???

Configuring Device Support

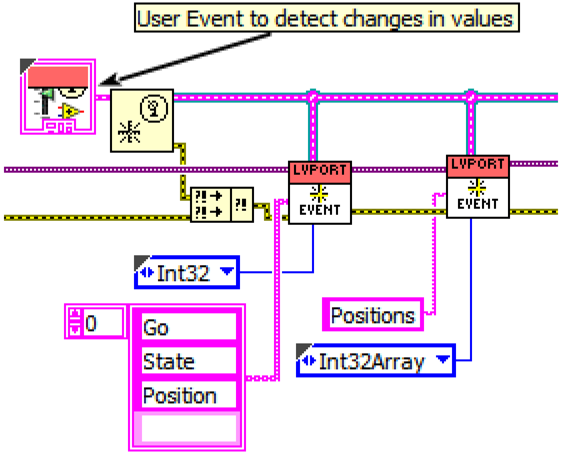
- **Configure port names, maximum number of parameters, and number of parameter lists.**
- **Then add parameters of each type.**
  - All supported asyn types.
    - LabVIEW code not yet implemented for handling generic pointer.

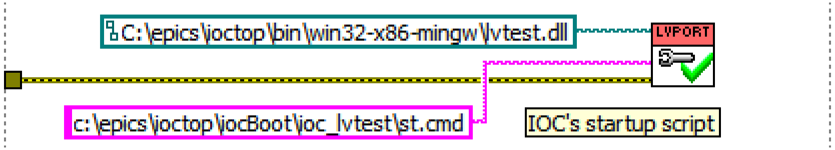


User Events and IOC startup

- **Can register for user events that occur on value change.**
  - – Uses the asyn client interface.

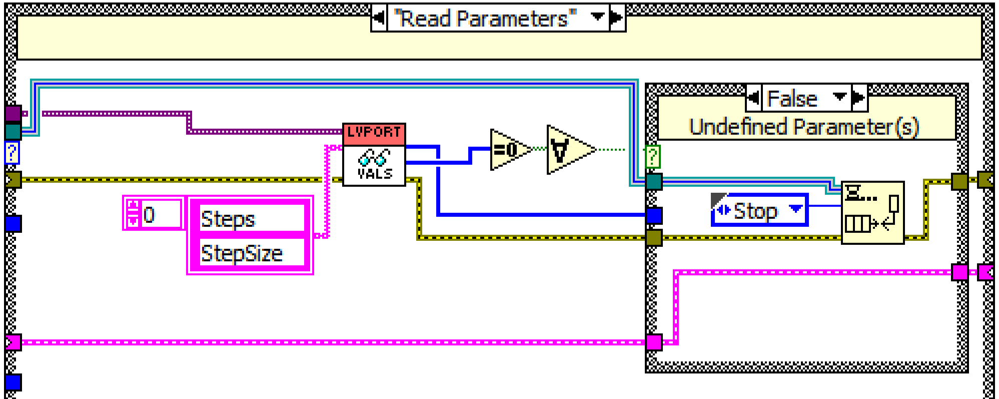
- **Start the IOC from LabVIEW**
  - – IOC built as a shared library
  - – Some changes to the main program allow access to the IOC shell.
    - Allocate a console for windows
    - Uses freopen for std I/O to connect to /dev/IOC before starting the iocshell.

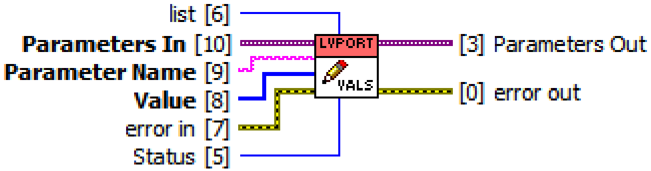




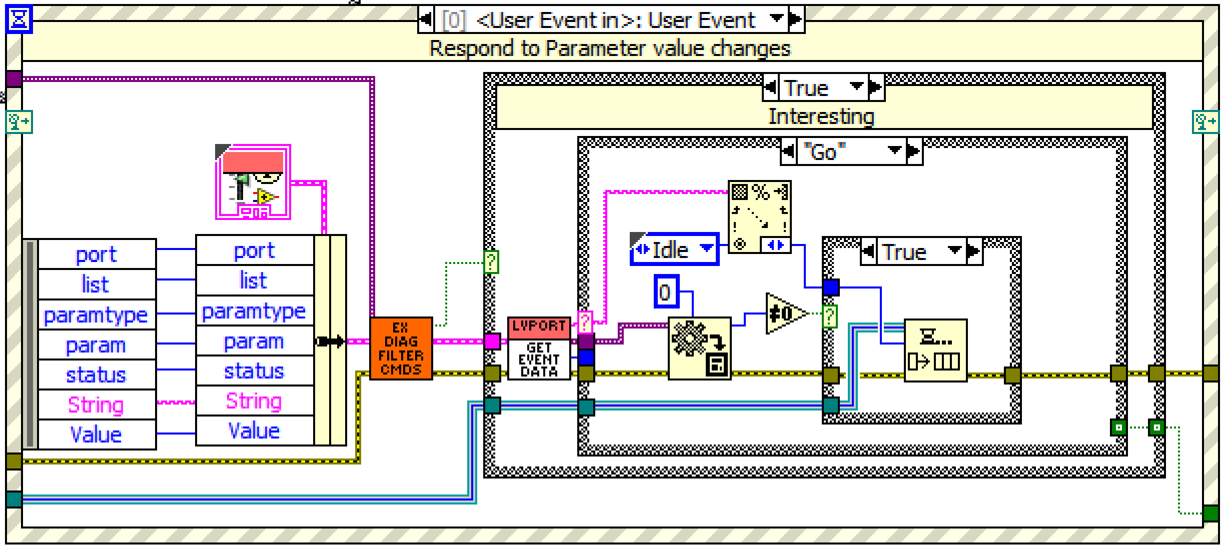
C:\epics\ioctop\bin\win32-x86-mingw\lvtest.dll
LVPORT
c:\epics\ioctop\iocBoot\ioc_lvtest\st.cmd
IOC's startup script

Using lvPortDriver (implementing device support)





- **Read**
- **Write**
  - Optional asynstatus (to indicate hardware limit, timeout, etc).
- **Respond to user events**



Where we will use this

- **Primarily for cRIO-based diagnostic systems.**
  - Our diagnostics team prefers to write in LabVIEW.
  - Programs frequently involve extensive calculations and logic implemented on the processor.
- **Someday perhaps on Linux RT (PXI)**
  - EPICS doesn’t currently run on Pharlap, and Linux RT is the future for PXI.

Where we will NOT use this

- **When a LabVIEW RT program is unnecessary.**
  - Our Industrial I/O code uses FPGA code and EPICS device support, the LabVIEW RT program only passed data between EPICS device support and the FPGA.
    - Logic is entirely within the FPGA or the EPICS database.
    - Configuration and scaling is in EPICS device support (C code).
  - I’ve modified the device support to use National Instrument’s C API for interfacing with the FPGA.

Accessing the IOC shell from Linux RT

- **Run IOC that connects stdio to /dev/IOC**
- **Create /dev/IOC**
  - socat -d -d pty,link=/dev/IOClink,raw,echo=0 pty,link=/dev/IOC,raw,mode=660,echo=0 &
- **From the Linux shell run socat - /dev/IOClink**
  - Could run this command via procserv if telnet access is desired.

Remaining Issues

- **Development environment for Windows Application can be difficult because there’s no good way to stop asyn Callback tasks once they are setup.**
  - This does not cause any problems for executable applications or for cross-platform development (where the shared libraries are not loaded).
  - Executables can call epicsExit, but in the development environment this also causes the development environment to close without warning.

Conclusion

- **A standard modern device support interface between EPICS and LabVIEW software.**
  - – Supports most asynPortDriver features.
    - Status
    - Parameter names/value/status reported by dbior.
  - – Could be extended to support additional features.
    - LabVIEW implementation of generic pointer.
    - Asynchronous processing could be done with event callbacks
  - – Will benefit from continued improvements in Asyn.
- **Implementation for a new program involves writing the LabVIEW program, and creating the EPICS database.**
  - – Not necessary to write matching C/C++ software, and keep them in synch.
  - – Easy to diagnose mismatched addresses.
- **lvPortDriver is available on github**
  - – https://github.com/sbaily/lvPortDriver