labCA Tutorial

Till Straumann

LCLS Control Group
strauman@slac.stanford.edu

Overview

- What is labCA
- General Concepts
  - PV s, Timestamps, Timeout, Error Handling, Help
- Basic Commands
  - lcaGet, lcaPut
- Monitors
- Examples
- Summary

What is labCA

- EPICS / Channel-Access interface for *matlab* and *scilab* (open-source, more features than *octave* but language not *matlab*-compatible)
- Builds on top of modified ezca library which has been made thread-safe. Use multi-threaded EPICS-3.14.
- read, write and monitor PVs
- relation to MCA: different history. I developed labCA for *scilab*, as it evolved *matlab* support was added

labCA 3.x Features

- handle multiple PVs in a single call
- retrieve data and time-stamp
- support for string data
- error handling
- asynchronous operation
- abort labCA with <Ctrl-C>
- on-line help and detailed documentation
- tested on linux, solaris, win, matlab, scilab, 32 and 64-bit.

General Concepts: PVs

- PVs are passed as a column vector of strings (PV names) (cell-array on matlab):

```
my_pvs = {'one_pv'; 'second_pv'; 'third_pv'};

lcaGet( my_pvs )
```

(no need for curly braces when dealing with single PV)

```
lcaGet( 'another_pv' )
```

General Concepts: Timestamps

- Timestamps have a 'quirky' format:
  - POSIX-timespec: number of nanoseconds since 0:0:0 1/1/1970 UTC is given as a *complex number*:

    full_seconds + j nanoseconds_remainder

  - Rationale:
    - I wanted to avoid more complex objects and stick with numbers
    - easy separation of seconds and sub-second parts (LCLS, SSRL embed 'pulse-ID' in nanosecond part)
    - standard POSIX format; utilities for conversion to local time available.

General Concepts: Timeout

- Synchronous lca routines block until request completes or times out
  - lcaGet, lcaPut, lcaNewMonitorWait, ...
- Total timeout is product of 'timeout' * 'retry_count'. Inspect and change with

```
lcaGetTimeout(), lcaSetTimeout( new_timeout )
lcaGetRetryCount(), lcaSetRetryCount( newcnt )
```

- Matlab is polled for <Ctrl-C> activity every 'timeout' (recommended: timeout=0.5s..1s, retry_count = total_timeout / timeout)

General Concepts: Errors

- Errors are reported by 'throwing' exceptions; use `try/catch/lasterror`.
- If an EPICS PV read with `lcaGet()` has a non-OK error status then a warning message is printed.
- If an EPICS PV has severity INVALID then lcaGet() returns NAN.
- Both of these features may be disabled and/or tuned; consult `lcaSetSeverityWarnLevel()`.

General Concepts: Help

- On-line documentation for all `lcaXXX()` calls available
- RT(F)M; available in PDF and HTML.
- latest versions always @

  www/~strauman/labca

Basic Commands: lcaGet/lcaPut

- Read a PV:

```
[ value, timestamp ] = lcaGet( <pv_name> )
```

- Write a PV

```
lcaPut( <pv_name>, <value> )
```

- Additional, optional arguments available (RTM)

Asynchronous Operation

- `lcaPut()` is synchronous, i.e., waits for completion of the write operation on the server (IOC).
  In some cases (e.g., driving stepper motor to target position) this may take a very long time.
- `lcaPutNoWait()` is asynchronous. It does not wait for completion (and you won't be notified if the operation fails). Merely sends the 'write' command and returns (AKA, 'posted-write').

Asynchronous Op.: Monitors

- `lcaSetMonitor( <pvs> )` registers a subscription (monitor) for a PV. Every time the value of the PV changes labCA is notified and a background task reads the new value into a internal buffer
- `lcaGet()` of a monitored PV reads from internal buffer
- Check if the background buffer has new data (since last `lcaGet()`):
  - `lcaNewMonitorValue( <pvs> )` returns flag vector (nonzero value flags new data)
  - `lcaNewMonitorWait( <pvs> )` blocks until all PVs have new data available.

Monitors (cont.)

- Subscription remains active until connection is released (`lcaClear( <pv> )`)

Monitors Example

- Trigger instrument and read new data (w/o monitor: how do we know if we have old or fresh data?)

```
my_pvs = {'det_1'; 'det_2' }; // helper
lcaSetMonitor( my_pvs );      // establish monitor
lcaNewMonitorWait( my_pvs );  // initial buffer fill
lcaGet( my_pvs );             // discard initial value
trigger_data_acq();           // trigger;
lcaNewMonitorWait( my_pvs );  // wait for new data
data = lcaGet( my_pvs );      // read new data
...
lcaClear( my_pvs );           // clear channel/monitor
```

Connection Management

- Background:
  - EPICS-Channel Access maintains one TCP connection for each IOC <-> client connection
    - multiple PVs on same IOC share one connection
    - multiple clients (matlab instances) to same IOC have *separate* connections
  - labCA establishes connections transparently and on-demand but never automatically releases them (except if matlab terminates). [Connection 'aging' and garbage-collection could be feature-addition.]

Connection Management (cont)

- labCA can release connections *explicitly*:
- `lcaClear()` releases all connections and monitors
- lcaClear(<pvs>) releases monitors and *channels* on named PVs – connection is released when last channel (=PV) using it is cleared.
- labCA cannot release monitor w/o clearing channel
- next time a cleared PV is used a new channel is created transparently (but takes a little bit longer).
- avoid frequent clearing/re-creation of channels; if done automatically at high frequency broadcast storms may result. Occasional clearing/re-connecting is OK.