UT-BATTELLE

EPICS Channel Access Client Coding

2006

kasemirk@ornl.gov

SNS
SPALLATION NEUTRON SOURCE

06-G00576/arm

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

Channel Access

- The main CA client interface is the "C" library that comes with EPICS base
  - Internally uses C++, but API is pure C.
- Almost all other CA client interfaces use that C library
  - Exception: New pure Java CAC, which for now still has some issues.
- Full documentation of the C API: "EPICS R3.14 Channel Access Reference Manual", Jeff Hill, on APS EPICS web page.
  - See same web site for copies of previous EPICS training material that basically presents every routine in the API.
  - This time:
    - Brief makeBaseApp.pl, Matlab, Java examples.
    - Point to some fundamental issues.

makeBaseApp.pl

- Includes a template for basic CA client in C:
  - Start with this:
    ```
    makeBaseApp.pl -t caClient cacApp
    make
    ```
  - Result:
    ```
    bin/linux-x86/caExample <some PV>
    bin/linux-x86/caMonitor <file with PV list>
    ```
  - Then read the sources, compare with the reference manual, and edit/extend to suit your needs.

makeBaseApp's caExample.c

- Minimal CA client program.
  - Fixed timeout, waits until data arrives.
  - Requests everything as 'DBR_DOUBLE'.
    - … which results in values of C-type 'double'.
    - See db_access.h header file for all the DBR_… constants and the resulting C types or structures.
    - In addition to the basic DBR_<type> requests, it is possible to request packaged attributes like DBR_CTRL_<type> to get { value, units, limits, …} in one request.

Excerpt from db_access.h

```
/* values returned for each field type
…
 *      DBR_DOUBLE      returns a double precision floating point number
…
 *      DBR_CTRL_DOUBLE returns a control double structure (dbr_ctrl_double)
 */
…
/* structure for a control double field */
struct dbr_ctrl_double{
        dbr_short_t     status;                 /* status of value */
        dbr_short_t     severity;               /* severity of alarm */
        dbr_short_t     precision;              /* number of decimal places */
        dbr_short_t     RISC_pad0;              /* RISC alignment */
        char            units[MAX_UNITS_SIZE];  /* units of value */
        dbr_double_t    upper_disp_limit;       /* upper limit of graph */
        dbr_double_t    lower_disp_limit;       /* lower limit of graph */
        dbr_double_t    upper_alarm_limit;
        dbr_double_t    upper_warning_limit;
        dbr_double_t    lower_warning_limit;
        dbr_double_t    lower_alarm_limit;
        dbr_double_t    upper_ctrl_limit;       /* upper control limit */
        dbr_double_t    lower_ctrl_limit;       /* lower control limit */
        dbr_double_t    value;                  /* current value */
};
```

makeBaseApp's caMonitor.c

- Better CA client program.
  - Registers callbacks to get notified when connected ot disconnected
  - Subscribes to value updates instead of waiting.
  - … but still uses the same data type (DBR_STRING) for everything.

Ideal CA client?

- Use callbacks for everything
  - no idle 'wait', no fixed time outs.
- Upon connection, check the channel's *native* type (int, double, string, …)
  - to limit the type conversion burden on the IOC.
- … request the matching DBR_CTRL_<type> *once*
  - to get the full channel detail (units, limits, …).
- … and then subscribe to DBR_TIME_<type> to get updates of only time/status/value
  - so now we always stay informed, yet limit the network traffic.
  - *Only subscribe once*, not with each connection, because CA client library will automatically re-activate subscriptions!
- This is what EDM, archiver, … do.
  - Quirk: They don't learn about online changes of channel limits, units, …. Doing that via a subscription means more network traffic, and CA doesn't send designated events for 'meta information changed'.

Side Note: SNL just to get CAC help

- This piece of SNL handles all the connection management and data type handling:

  -
    ```
    double value;
    assign value to "fred";
    monitor value;
    ```
- Extend into a basic 'camonitor':

  -
    ```
    evflag changed;
    sync value changed;

    ss monitor_pv
    {
      state check
      {
          when (efTestAndClear(changed))
          {
              printf("Value is now %g\n", value);
          } state check
      }
    }
    ```

Quick Hacks, Scripts

- In many cases, one can get by just fine by invoking the command-line 'caget' from within bash/perl/python/php.
- Especially if you only need to read/write one value of a PV, not a subscription!
- There are more elaborate CAC bindings available for perl/python/php
  - But that means you have to find, build and later maintain these!
  - A basic p* script is portable, but you'd have to install the CAC-for-p* binding separately for Linux, Win32, MacOS…

Perl Example

```
use English;

# Get the current value of a PV
# Argumment: PV name
# Result: current value
sub caget($)
{
    my ($pv) = @ARG;
    open(F, "caget -t $pv |") or die "Cannot run 'caget'\n";
    $result=<F>;
    close(F);
    chomp($result);
    return $result;
}

# Do stuff with PVs
$fred = caget("fred");
$jane = caget("jane");
$sum = $fred + $jane;
printf("Sum: %g\n", $sum);
```

Matlab 'MCA' Extension

- Same setup & maintenance issue as for p/p/p!
  - … but may be worth it, since Matlab adds tremendous number crunching and graphing.
- Initial setup
  - Get MCA sources (see links on APS EPICS web)
  - Read the README, spend quality time with MEX.
- Assume that's done by somebody else
  - You are in the SNS control room
  - 'caget' from EPICS base works
  - Matlab works (try "matlab -nojvm -nodesktop")
- Do this once:

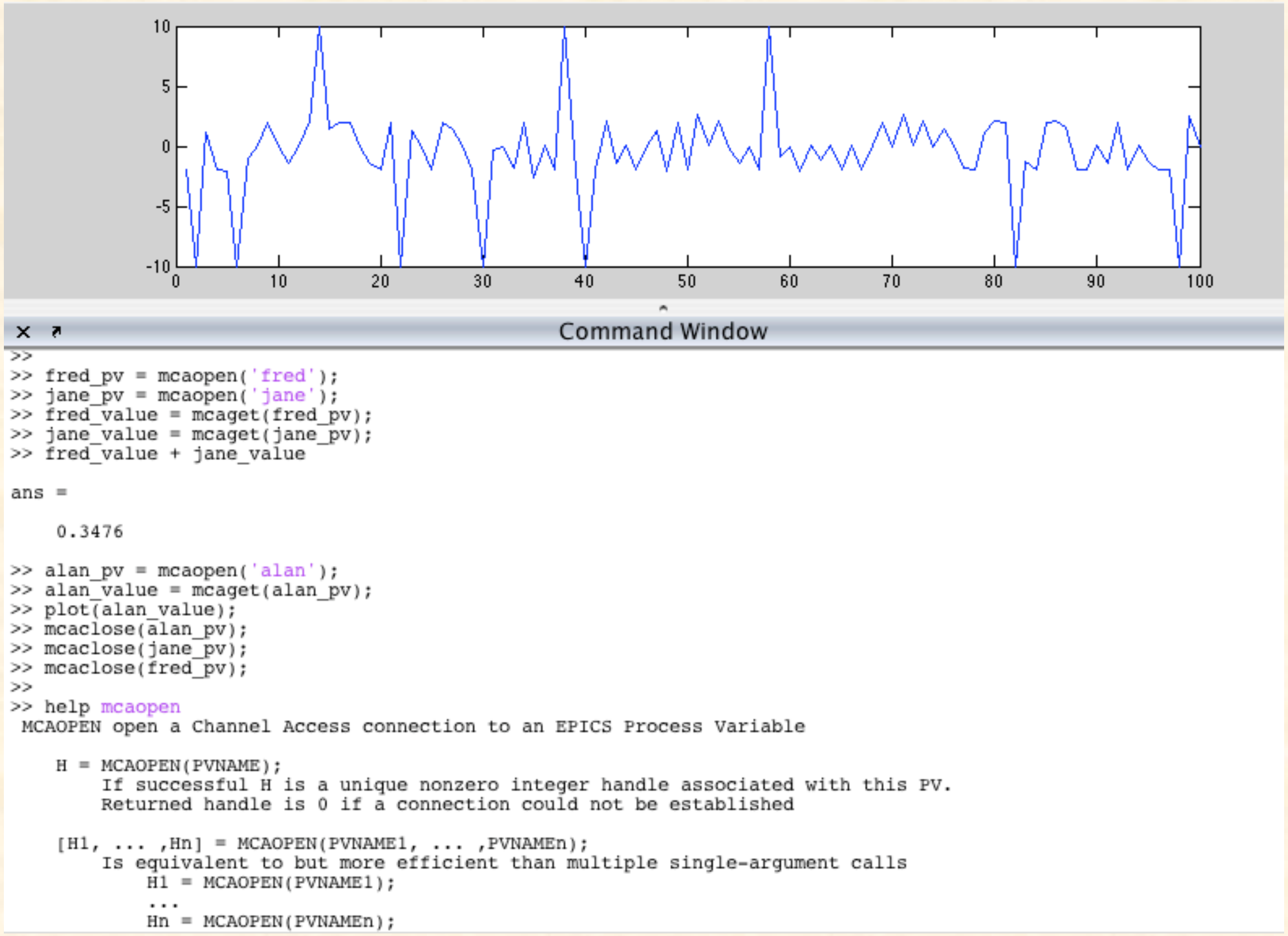
```
cd $EPICS_EXTENSIONS/src/mca
source setup.matlab
```

  - … and from now on, Matlab should include MCA support

MCA Notes

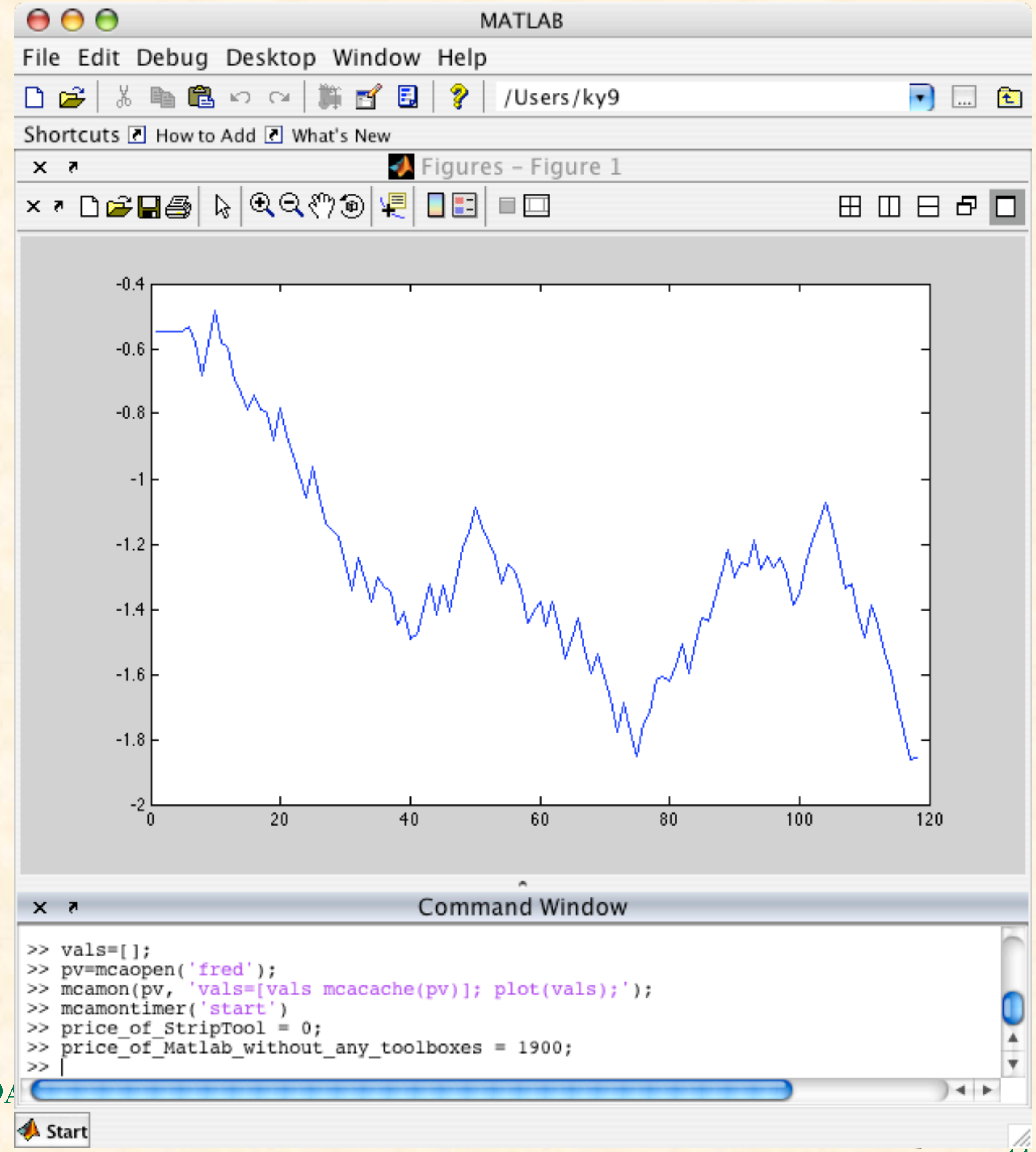
- Basically, it's a chain of
  - pv = mcaopen('some_pv_name');
  - value = mcaget(pv);
  - mcaput(pv, new_value);
  - mcaclose(pv);
- Your pv is 'connected' from ..open to ..close
  - When getting more than one sample, staying connected is much more efficient than repeated calls to 'caget'.
- Try 'mca<tab>' command-line completion to get a list of all the mca… commands
- Run 'help mcaopen' etc. to get help

Matlab/MCA Examples



```
>>
>> fred_pv = mcaopen('fred');
>> jane_pv = mcaopen('jane');
>> fred_value = mcaget(fred_pv);
>> jane_value = mcaget(jane_pv);
>> fred_value + jane_value

ans =

    0.3476

>> alan_pv = mcaopen('alan');
>> alan_value = mcaget(alan_pv);
>> plot(alan_value);
>> mcaclose(alan_pv);
>> mcaclose(jane_pv);
>> mcaclose(fred_pv);
>>
>> help mcaopen
 MCAOPEN open a Channel Access connection to an EPICS Process Variable

    H = MCAOPEN(PVNAME);
        If successful H is a unique nonzero integer handle associated with this PV.
        Returned handle is 0 if a connection could not be established

    [H1, ... ,Hn] = MCAOPEN(PVNAME1, ... ,PVNAMEn);
        Is equivalent to but more efficient than multiple single-argument calls
            H1 = MCAOPEN(PVNAME1);
            ...
            Hn = MCAOPEN(PVNAMEn);
```

MCA Value Subscription



```
>> vals=[];
>> pv=mcaopen('fred');
>> mcamon(pv, 'vals=[vals mcacache(pv)]; plot(vals);');
>> mcamontimer('start')
>> price_of_StripTool = 0;
>> price_of_Matlab_without_any_toolboxes = 1900;
>> |
```

OA

Java

- There is actually a JNI and a pure Java binding.
  - Only difference in initialization, then same API.
  - Usage very much like C interface, "real programming" as opposed to Matlab, but in a more forgiving Java VM.
- See Docs/Java CA example.

Acknowledgements

- Channel Access on every level in detail:
  - Jeff Hill (LANL)
- makeBaseApp.pl
  - Ralph Lange (BESSY) and others
- MCA
  - Andrei Terebilo (SLAC) is the original author,
  - Carl Lionberger maintained it for a while (then SNS)
- Java CA
  - Eric Boucher is the original author (then APS),
  - Matej Sekoranja maintains it;
    he added the pure java version (Cosylab)