CA Lab - LabVIEW (Realtime) + EPICS

Overview

What is CA Lab?

It is a

- user-friendly,
- lightweight and
- high performance

interface between LabVIEW™ and EPICS.

This interface uses

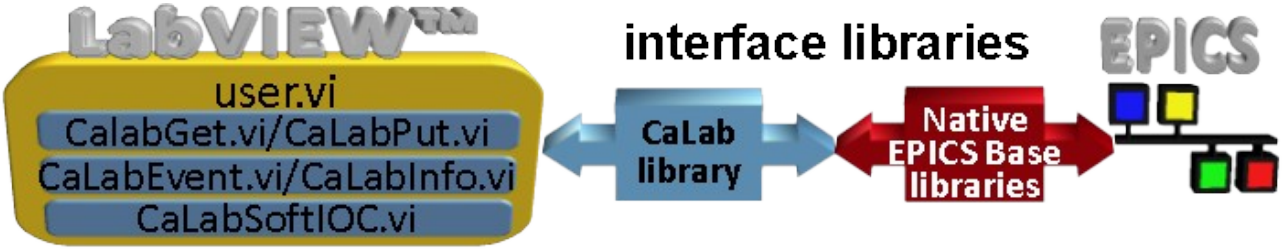
- proven EPICS BASE libraries (V7),
- a CA Lab interface library
- and polymorphic Vis

to access EPICS variables.

Creating, reading and writing EPICS variables is very simple now. Also, **user events** for EPICS variables can be implemented easily.
EPICS time stamp, status, severity, and optional PV fields (properties) are bound into a resulting data cluster. You avoid inconsistent data sets.
It's easy to create an executable of your VI.

CA Lab works with Windows®, Linux and Realtime Linux.
This interface requires only LabVIEW™.
To use this interface, it's not necessary to create any LabVIEW™ project nor to use external services. CA Lab can be used directly.

It is free! No additional licenses are required.
CA Lab is **open source** and works with all LabVIEW™ versions from 8.5 up to the current version (32bit/64bit).
It has been tested under Windows 7®, Windows 10/11® , Linux (RHEL 8.5) and NI Linux RT (2022).



schema of CA Lab interface

Any VI can use caLabGet.vi to read or caLabPut.vi to write EPICS variables.

Use caLabEvent.vi to create user events for any EPICS variables.

Call CaLabInfo.vi to get context information of the CA Lab library.

You can use CaLabSoftIOC.vi to create new EPICS variables and start them.

These CA Lab VIs call the interface library 'caLab', which uses EPICS base libraries 'ca' and 'Com' to provide Channel Access functions.

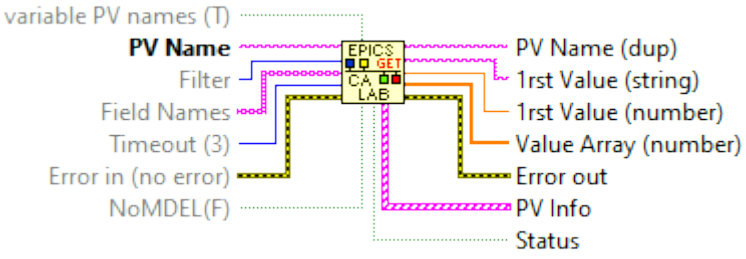
CA Lab library builds an internal PV cache and monitors PVs to improve the read and write access and reduce network traffic. Optional, you can disable caching.

CA Lab includes a EPICS Base package (caget, caput, camonitor, softIOC and more).

VIs of CA Lab

CA Lab Get VI (read values)

This is a polymorphic VI for reading values or fields of EPICS variables.



INPUT:

- *PV Name(s)*
  - Name of EPICS variable **OR** string array of EPICS variable names
  - Attention if you use an array of EPICS variable names!
    Mixed target data types arise as a result of a 2-D array with size of:
    number of names **multiplied by** *size of largest data type of the group*
- *Filter*
  - optional
  - output filter (this can save ressources and increase performance)
    Following values can be combined with OR:
    - PV indicators:
      - firstValueAsString = 1
      - firstValueAsNumber = 2
      - valueArrayAsNumber = 4
      - errorOut = 8
    - PV Info cluster:
      - pviElements = 16
      - pviValuesAsString = 32
      - pviValuesAsNumber = 64
      - pviStatusAsString = 128
      - pviStatusAsNumber = 256
      - pviSeverityAsString = 512
      - pviSeverityAsNumber = 1024
      - pviTimestampAsString = 2048
      - pviTimestampAsNumber = 4096
      - pviFieldNames = 8192
      - pviFieldValues = 16384
      - pviError = 32768
- *Field Names*
  - Optional
  - String array of static EPICS fields ("EGU", "DESC" etc.)
- *Timeout (3)*
  - Optional
  - Timeout for Channel Access requests in seconds (1-255)
  - Default value: 3 seconds
- *NoMDEL (F)*
  - optional
  - TRUE:
    - ignores EPICS PV field (parameter) MDEL
    - ignores monitor dead-band of Channel Access
    - all changes of values are visible (like caget)
    - disables caching of values -> higher network impact!
  - FALSE:
    - default
    - works with notifications (like camonitor)

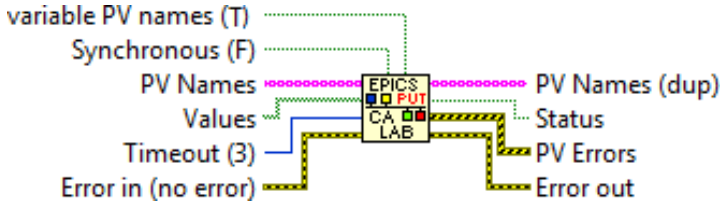
        - significant changes of values are visible (MDEL)
        - enables caching of values -> lower network impact!
- *variable PV names (T)*
    - optional
    - true: in the case of changing PV names at PV name input (performance drop)
    - false: in the case of static PV names (best performance)
- *Error in (no error)*

OUTPUT:

- *PV Names (dup)*
    - Name of EPICS variable **OR** string array of EPICS variable names
- *1st Value (string)*
    - Value of variable as String / Array of first values
    - Shows first item of value array(s) only
- *1st Value (number)*
    - Value of variable as Double / Array of first values
    - Shows first item of value array
- *Value array (number)*
    - All values of variable in double array (1D/2D)
- PV Info /PV Info array
- *Status*
    - FALSE: request(s) successful
    - TRUE: warnings or errors
- *Error out*
    - Global I/O error

CA Lab Put VI

This is a polymorphic VI for writing values to EPICS variables.



INPUT:

- PV Name
    - Name(s) of EPICS variable OR string array of EPICS variable names
- Value(s)
        - Boolean (array 1D/2D), String (array 1D/2D), Single (array 1D/2D), Double (array 1D/2D), Byte signed integer(array 1D/2D), Word signed integer(array 1D/2D), Quad signed integer(array 1D/2D)
          →2D array of values only fits together with 1D array of PV names!
        - Attention: Mixed target data types arise as a result of a 2-D array of size (number of names) multiplied by (size of largest data type)!
- Timeout (3)
    - Optional
    - Timeout for Channel Access requests in seconds (1-255)
    - Default value: 3 seconds
- Synchronous
    - Optional
    - True = writes value(s) after waiting for the end of record processing
    - False = writes value(s) without waiting for the end of record processing (DEFAULT)
    - Note: Old projects that use the deprecated "fire and forget" feature in caLabPut, are no longer compatible!
- variable PV names (T)
    - optional

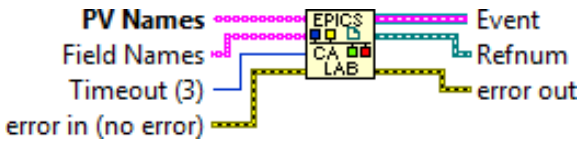
    - true: in the case of changing PV names at PV name input (performance drop)
    - false: in the case of static PV names (best performance)
- Error in (no error)

OUTPUT:

- PV Name (dup)
    - Name(s) of EPICS variable OR string array of EPICS variable names
- Status
    - FALSE: request(s) successful
    - TRUE: warnings or errors
- Errors
    - Array of errors for every PV
    - Result of Channel Access communication
    - Error status
    - Error code (value 1 stands for success because we use the EPICS nomenclature)
    - Error source (message)
- Error out
    - Global I/O error

CA Lab Event VI

This is a VI for creating user events of EPICS variables.



CA Lab Event Unregister VI should be called before terminating a VI with *CA Lab Event VI*.

INPUT:

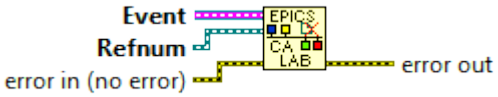
- PV Names
    - String array of EPICS variable names OR string
    - Attention if you use an array of EPICS variable names!
      Mixed target data types arise as a result of a 2-D array with size of:
      number of names **multiplied by** *size of largest data type of the group*
- Field Names
    - Optional
    - String array of EPICS fields ("EGU", "DESCRIPTION" etc.)
- Timeout (3)
    - Optional
    - Timeout for Channel Access requests in seconds (1-255)
    - Default value: 3 seconds
- error in (no error)

OUTPUT:

- Event
    - Event structure
- Refnum
    - Event registration number
- Error out
    - Global I/O error

Ca Lab EventUnregister VI

This VI should be called before terminating a VI with CA Lab User events (CA Lab Event VI). It terminates existing subscriptions and prevents unwanted data exchange.



INPUT:

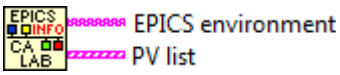
- Event
  - Event structure (coming from CA Lab Event VI)
- Refnum
  - Event registration number (coming from CA Lab Event VI)
- error in (no error)

OUTPUT:

- error out
  - Global I/O error

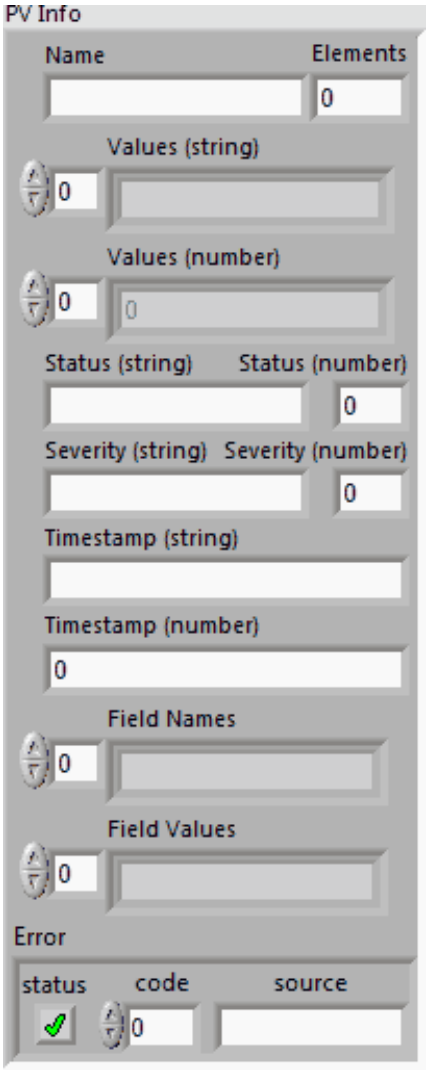
CA Lab Info VI

This VI shows the current EPICS context.



OUTPUT:

- CA Lab version
- EPICS environment
  - Environment variables of EPICS
- PV list
  - Snapshot of all currently subscribed EPICS variables



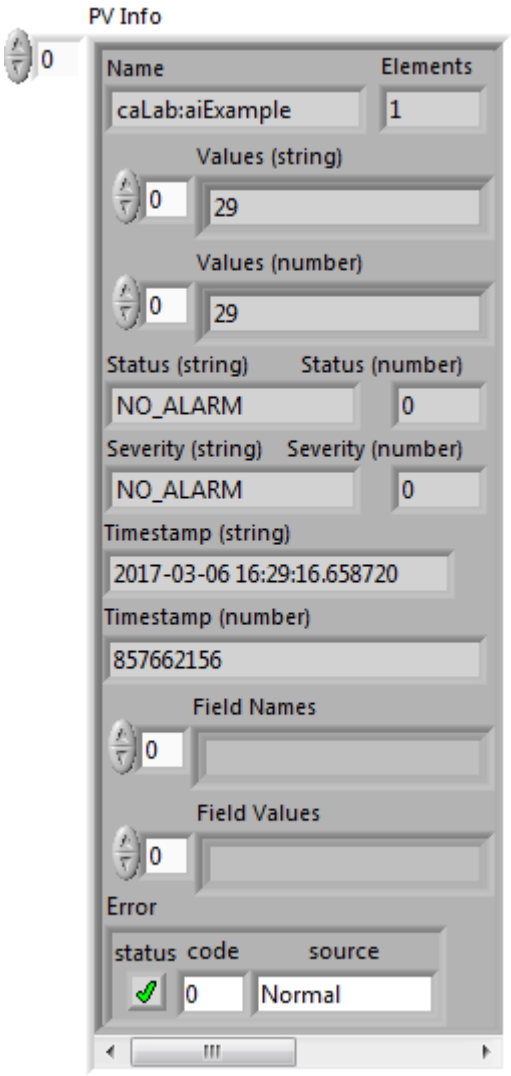
Cluster of all PV data

- Name
  - Name of EPICS variable
- Elements
  - Size of value array
- Values (string)
  - All values of variable in a string array
- Values (number)
  - All values of variable in a double array
- Elements
  - Size (elements) of value array
- Status (string)
  - Status of EPICS variable as String
- Status (number)
  - Status of EPICS variable as Long
- Severity (string)
  - Severity of EPICS variable as String
- Severity (number)
  - Severity of EPICS variable as Long
- Timestamp (string)
  - Timestamp of EPICS variable as String
- Timestamp (number)

    - Timestamp of EPICS variable as Long
- Field Names
    - String array of EPICS field names
- Field Values
    - String array of requsted EPICS field values
- Error
    - Result of Channel Access communication
    - Error status
    - Error code (value 1 stands for success because we use the EPICS nomenclature)
    - Error source (message)

PV Info Control

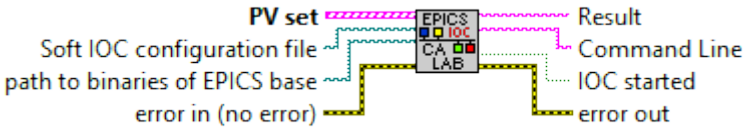
Cluster of all PV data (array)



- *Name*
    - Name of EPICS variable
- *Elements*
    - Size of value array
- *Values (string)*
    - All values of variable in a string array
- *Values (number)*
    - All values of variable in a double array
- *Elements*
    - Size (elements) of value array
- *Status (string)*
    - *Status* of EPICS variable as String
- *Status (number)*
    - *Status* of EPICS variable as Long
- *Severity (string)*
    - *Severity* of EPICS variable as String
- *Severity (number)*
    - *Severity* of EPICS variable as Long
- *Timestamp (string)*
    - *Timestamp* of EPICS variable as String
- *Timestamp (number)*
    - *Timestamp* of EPICS variable as Long
- *Filed Names*
    - String array of EPICS field names
- *Filed Values*
    - String array of requsted EPICS field values
- *Error*
    - Result of Channel Access communication
    - Error status
    - Error code (value 1 stands for success because we use the EPICS nomenclature)
    - Error source (message)

CA Lab SoftIOC

This is a VI for creating new EPICS variables and run them in a Soft IOC.
(On Linux, the app *screen* is required to run SoftIOC in the background).



INPUT:

- PV set
  - Cluster-array of names, data types and field definitions to create a new EPICS PV
- Soft IOC configuration file
  - Target of configuration (db-file)
- path to binaries of EPICS base
  - Windows: C:\Program Files\National Instruments\LabVIEW\user.lib\caLab\Lib\
  - Linux: /usr/local/epics/bin/linux-x86_64/
- error in (no error)

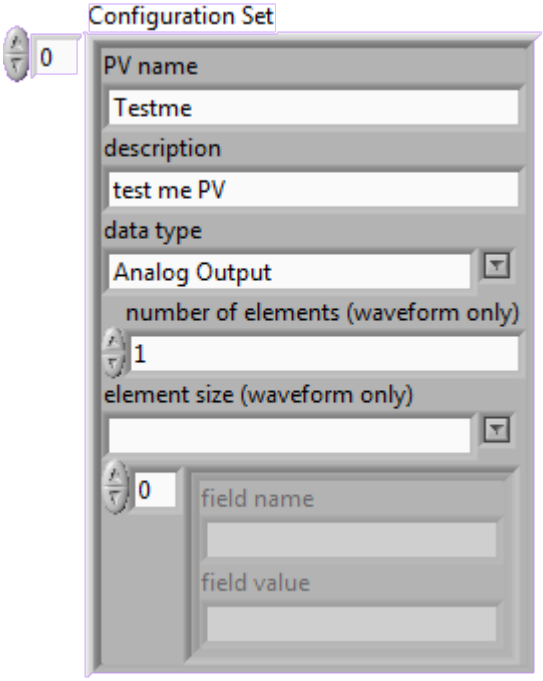
OUTPUT:

- Result
  - Initial output of Soft IOC shell (checkpoint of errors)
- Command Line
  - Command string: How the Soft IOC shell was called
- IOC started
  - TRUE: Soft IOC runs and is ok
  - FALSE: Soft IOC is in trouble (check "result")
- error out
  - Global I/O error

All created EPICS variables will be destroyed when CaLabSoftIOC stops.

Configuration Set

Cluster array of configurations of all EPICS variables.



PV Name

- Name of new EPICS variable (Please check whether this name is unique in your network!!! CaLabGet is a good tool to do that.)

Description

- User defined description of PV

Data type

- EPICS data type

Number of elements (waveform only)

- Use for data type "waveform" only!
- Size of array (elements)

Element size (waveform only)

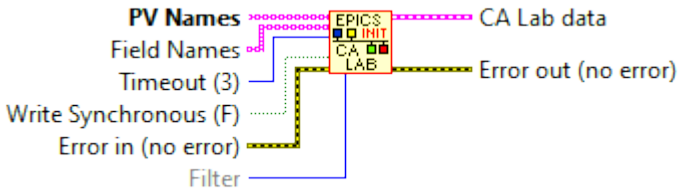
- Use for data type "waveform" only!
- Data type of array elements

Array of field definitions

- Field-value-set to define properties of EPICS variable

CA Lab Init

This is a polymorphic VI for optional initialisation of EPICS variables.



INPUT:

- *PV Name(s)*
  - Name of EPICS variable **OR** string array of EPICS variable names
  - Attention if you use an array of EPICS variable names!
    Mixed target data types arise as a result of a 2-D array with size of:
    number of names **multiplied by** *size of largest data type of the group*
- *Field Names*
  - Optional
  - String array of static EPICS fields ("EGU", "DESC" etc.)
- *Timeout (3)*
  - Optional
  - Timeout for Channel Access requests in seconds (1-255)
  - Default value: 3 seconds
- *Write Synchronous (F)*
  - optional
  - TRUE:
    - writes value(s) after waiting for the end of record processing
  - FALSE:
    - default
    - writes value(s) without waiting for the end of record processing
- *Filter*
  - optional
  - output filter (this can save ressources and increase performance)
    Following values can be combined with OR:
    - PV indicators:
      - firstValueAsString = 1
      - firstValueAsNumber = 2
      - valueArrayAsNumber = 4
      - errorOut = 8
    - PV Info cluster:
      - pviElements = 16
      - pviValuesAsString = 32
      - pviValuesAsNumber = 64
      - pviStatusAsString = 128
      - pviStatusAsNumber = 256
      - pviSeverityAsString = 512
      - pviSeverityAsNumber = 1024
      - pviTimestampAsString = 2048

                - pviTimestampAsNumber = 4096
                - pviFieldNames = 8192
                - pviFieldValues = 16384
                - pviError = 32768
- *Error in (no error)*

OUTPUT:

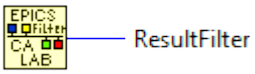
- CA Lab data
    - Cluster with preinitialized data
- *Error out*
    - Global I/O error

CaLabGet.vi and CaLabPut.vi can use "CA Lab data" cluster as input (first connector) to skip their automatically internal initialisation. Preinitialized CaLabGet.vi and CaLabPut.vi will start faster.

CaLabInit.vi is good if a VI uses a large number (>100) of EPICS PVs

CA Lab Result Filter VI

With this VI you can filter the output channels for the "CA Lab Get VI". If this filter is set, then only the output channels that are included in the filter are enabled.



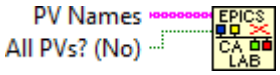
OUTPUT:

- *ResultFilter*
    - output filter (this can save ressources and increase performance)
      Following values can be combined with OR:
        - PV indicators:
            - firstValueAsString = 1
            - firstValueAsNumber = 2
            - valueArrayAsNumber = 4
            - errorOut = 8
        - PV Info cluster:
            - pviElements = 16
            - pviValuesAsString = 32
            - pviValuesAsNumber = 64
            - pviStatusAsString = 128
            - pviStatusAsNumber = 256
            - pviSeverityAsString = 512
            - pviSeverityAsNumber = 1024
            - pviTimestampAsString = 2048
            - pviTimestampAsNumber = 4096
            - pviFieldNames = 8192
            - pviFieldValues = 16384
            - pviError = 32768

CaLabInit.vi and CaLabGet.vi can use the "ResultFilter" as input (filter) to define active output connectors of CaLabGet.vi. This allows the performance and the required resources to be optimized. A typical use case are large waveform records (value arrays) and when a large number of Epics variables are queried

CA Lab Disconnect VI

This is a polymorphic VI for disconnecting a list / all subscribed EPICS variables.



INPUT:

- *PV Name(s)*
  - Name ofEPICSvariable**OR**string array ofEPICSvariable names
- *All PVs? (No)*
  - optional
  - TRUE:
    - disconnect all subscribed EPICS variables
    - all Channel Access connections will be closed
  - FALSE:
    - default
    - disconnect all EPICS variables in array of "PV Name(s)"
    - specific Channel Access connections will be closed

Environment of CA Lab

CA Lab environment variables

CALAB_NODBG

- Environment variable for redirecting debug window output to a file
- CALAB_NODBG = PATH_NAME_OF_LOG_FILE

CALAB_POLLING

- Environment variable for switching off monitoring of EPICS variables
- Useful to avoid permanent open network ports. (CompactRIO)
- CA Lab checks only if CALAB_POLLING is defined. The value does not matter.
- CALAB_POLLING = TRUE

EPICS environment variables

A good documentation can be found here:
https://docs.epics-controls.org/en/latest/sys-admin/configure-ca.html

---

What does Ca Lab library do?

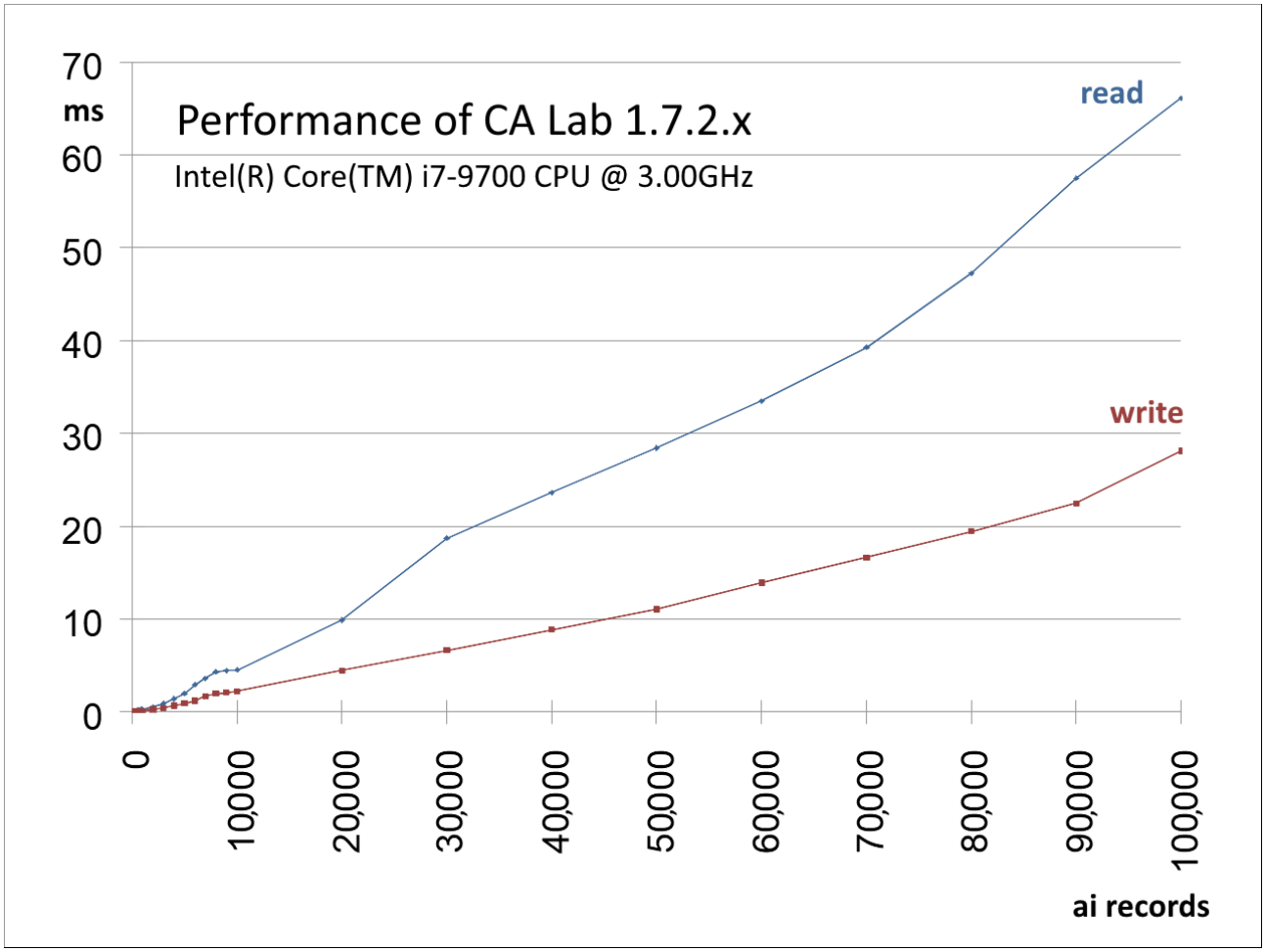
caLab.dll / libCaLab.so (interface between LabVIEW™ and EPICS)

Functions:

- Handles Channel Access activities
- Administrates list of used EPICS variables
- Monitors used EPICS variables
- Buffers Channel Access requests

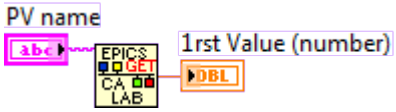
- Reports events (user events)
- Works with
    - DBR_TIME_STRING
    - DBR_TIME_SHORT
    - DBR_TIME_FLOAT
    - DBR_TIME_ENUM
    - DBR_TIME_CHAR
    - DBR_TIME_LONG
    - DBR_TIME_DOUBLE

Performance of CA Lab

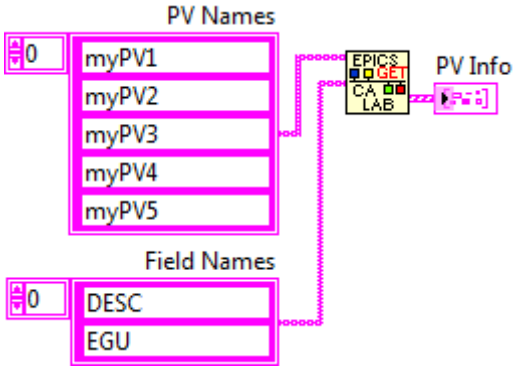


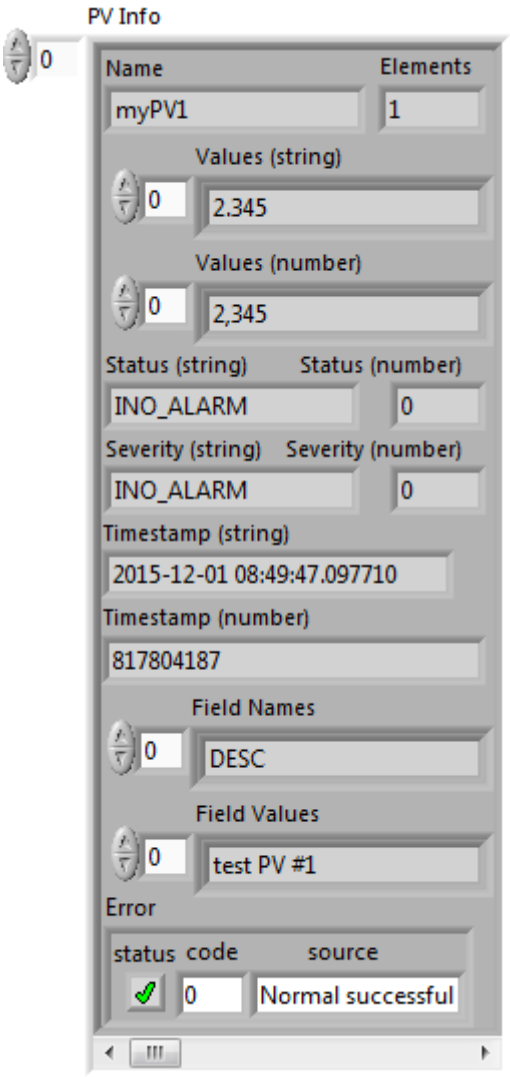
Examples of CA Lab

Example of reading one value

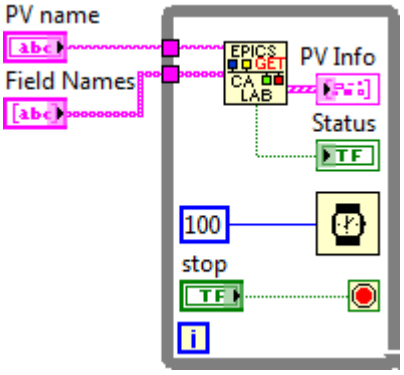


Example of reading many values

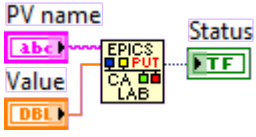




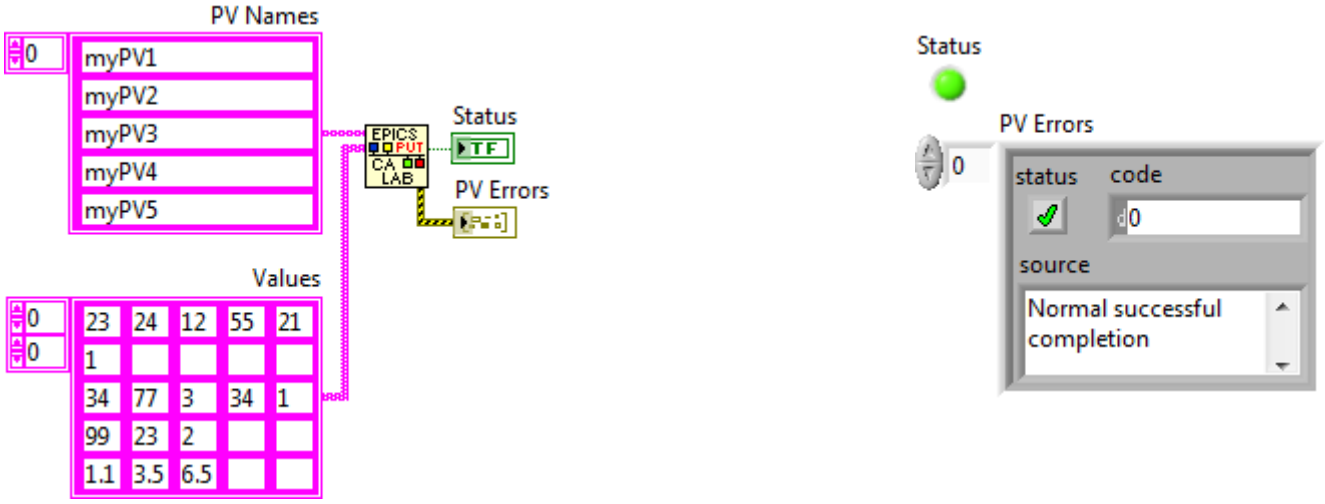
Example how to monitor a value



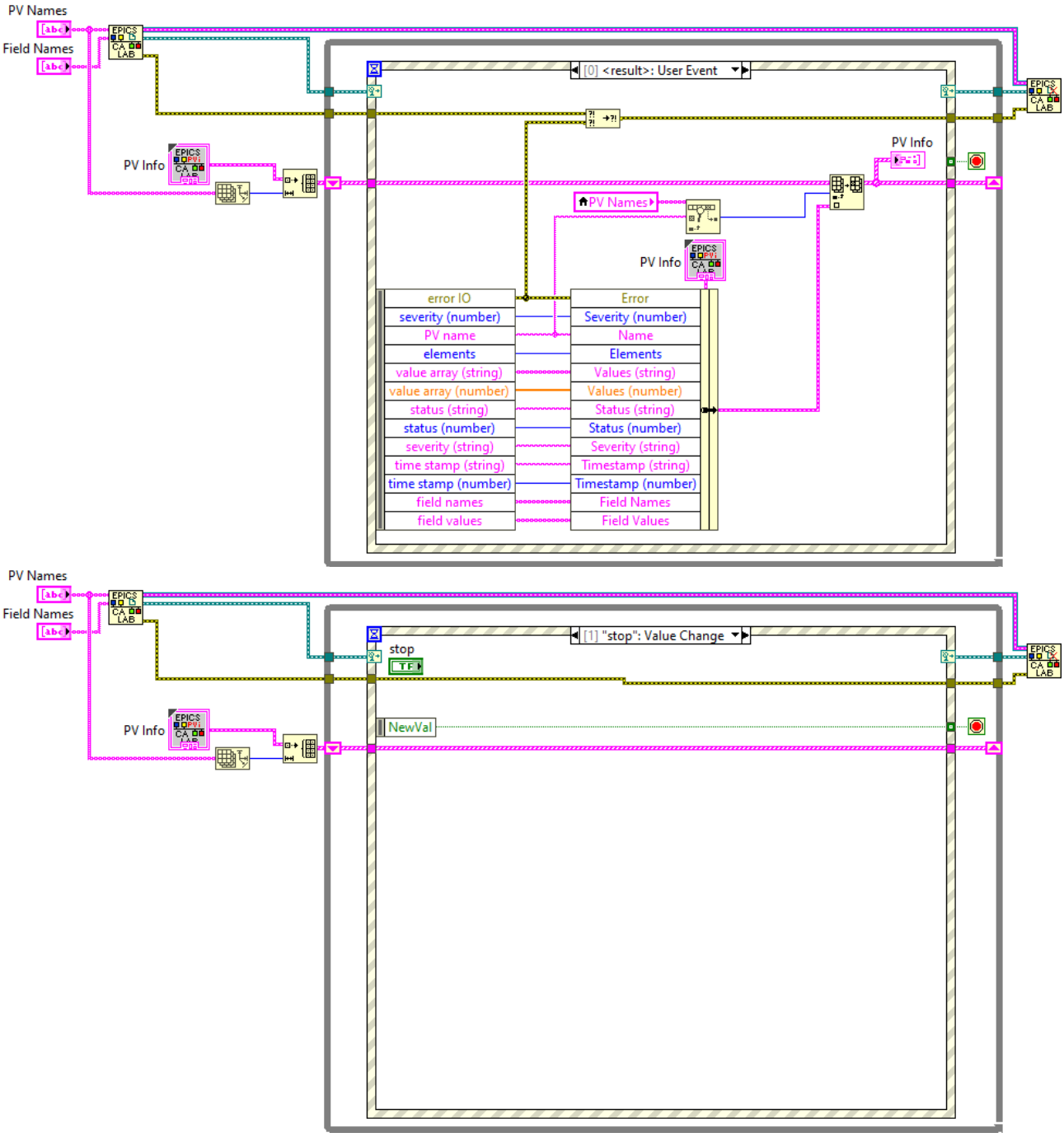
Example of writing one value



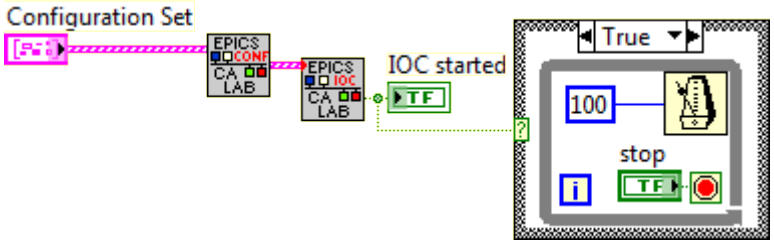
Example of writing many values



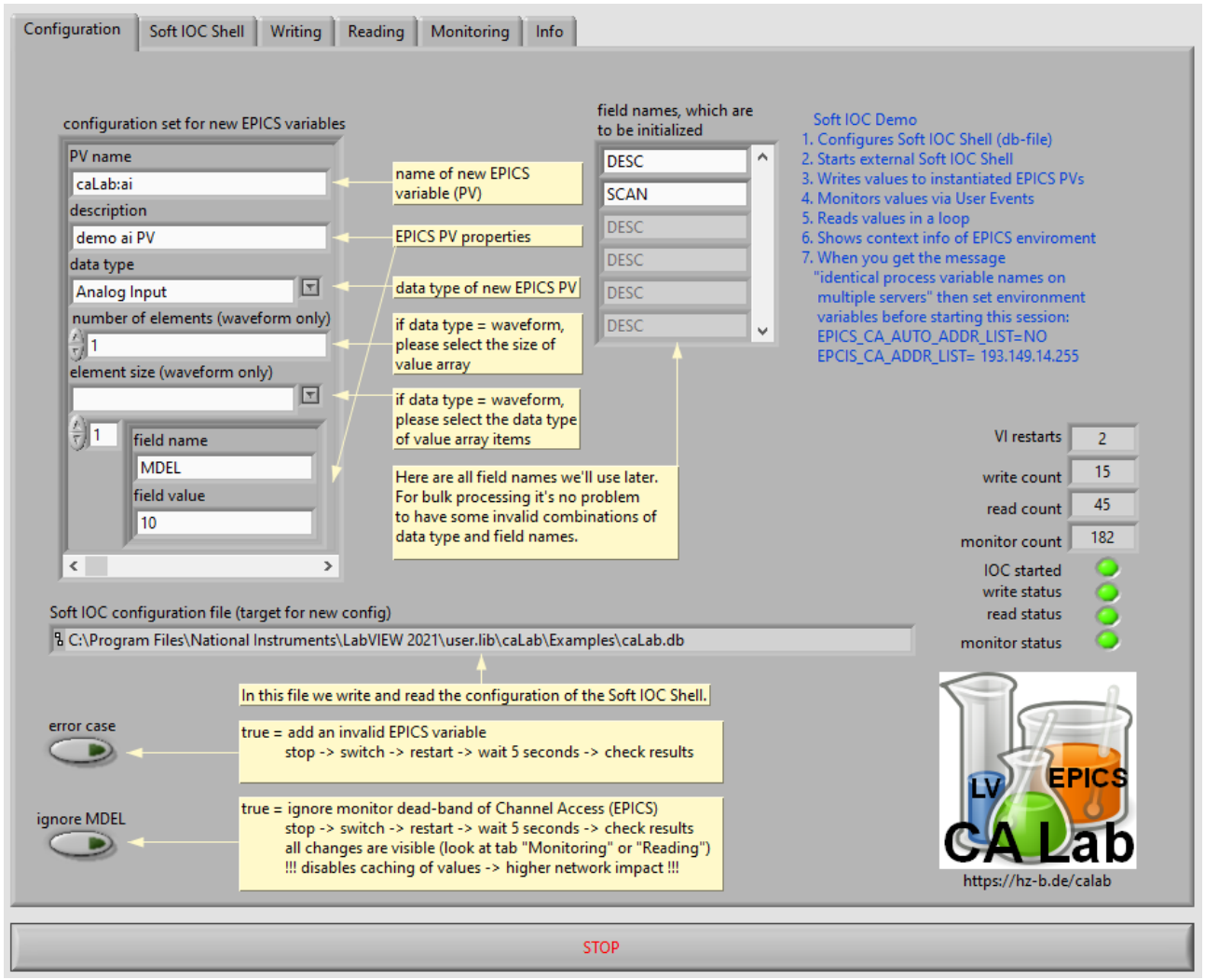
Example of User Event

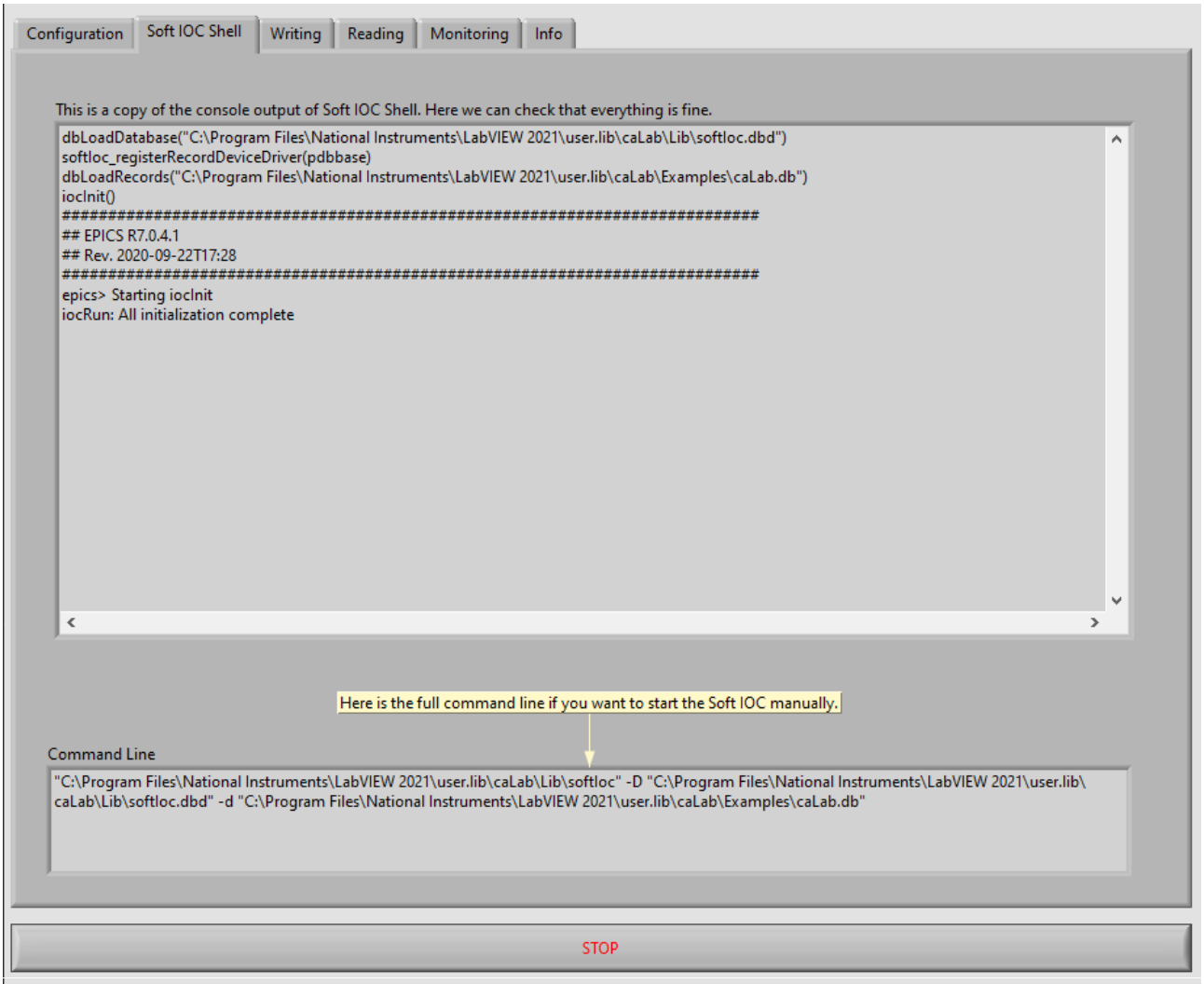


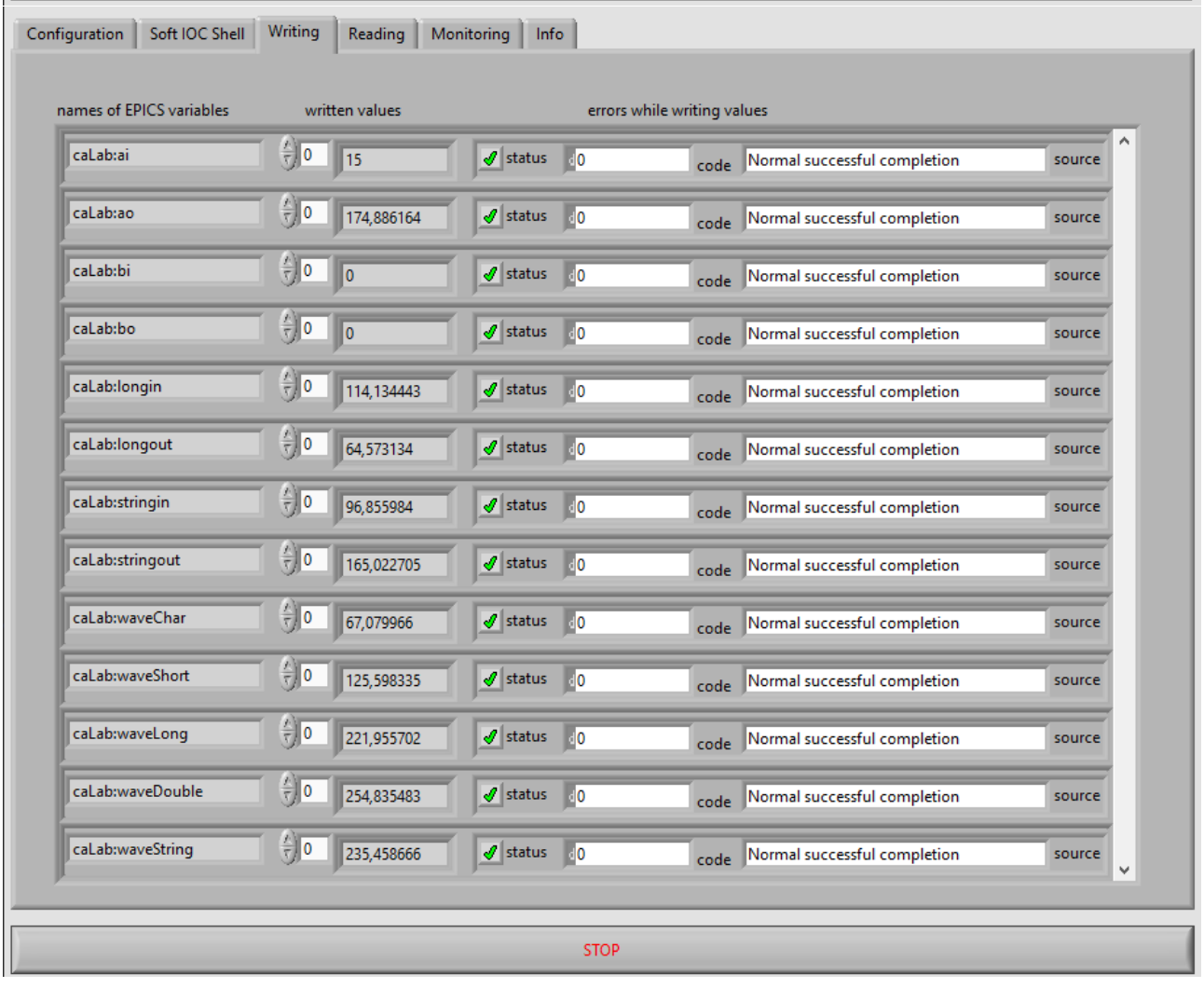
Example how to create new EPICS variables and run them in a Soft IOC

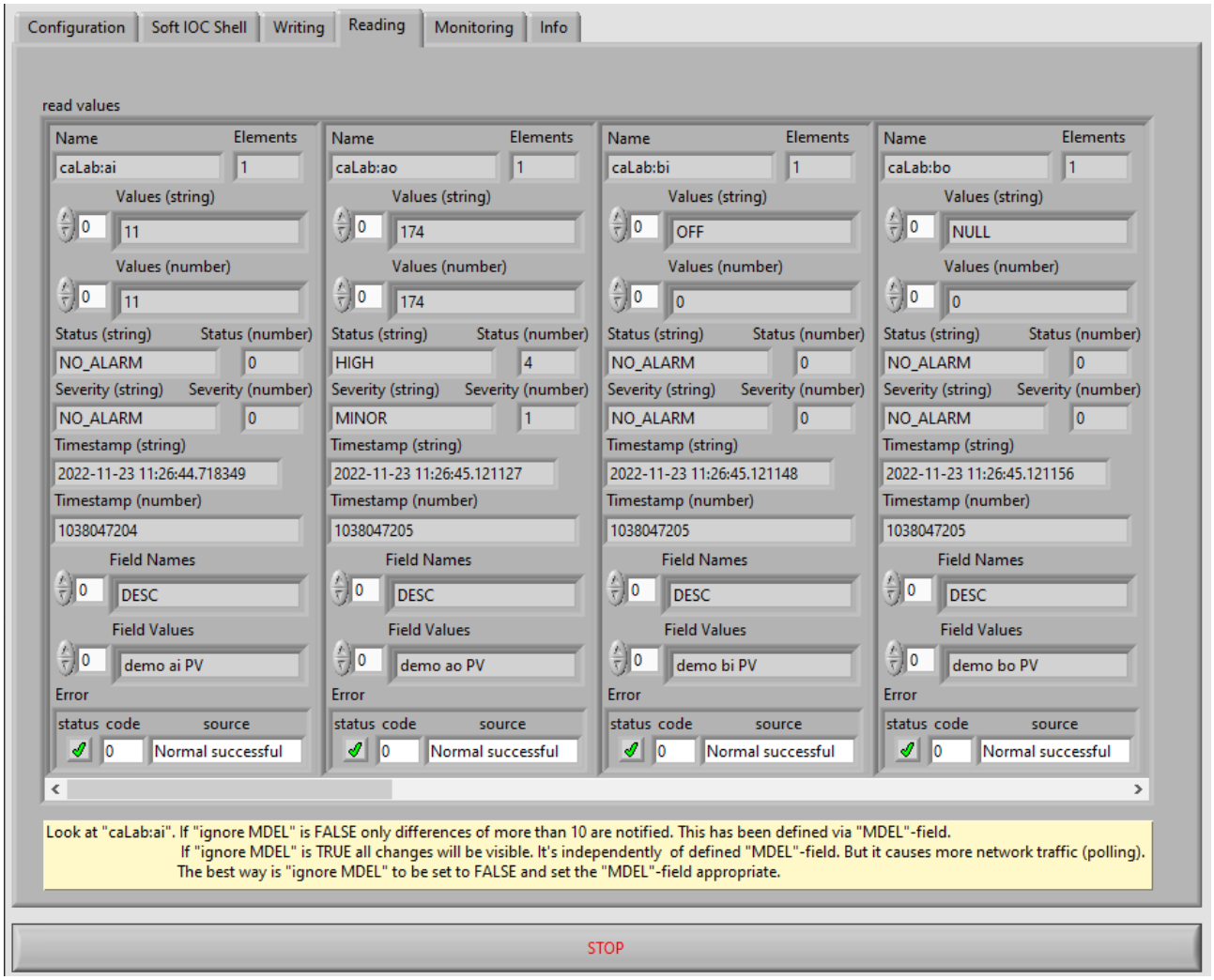


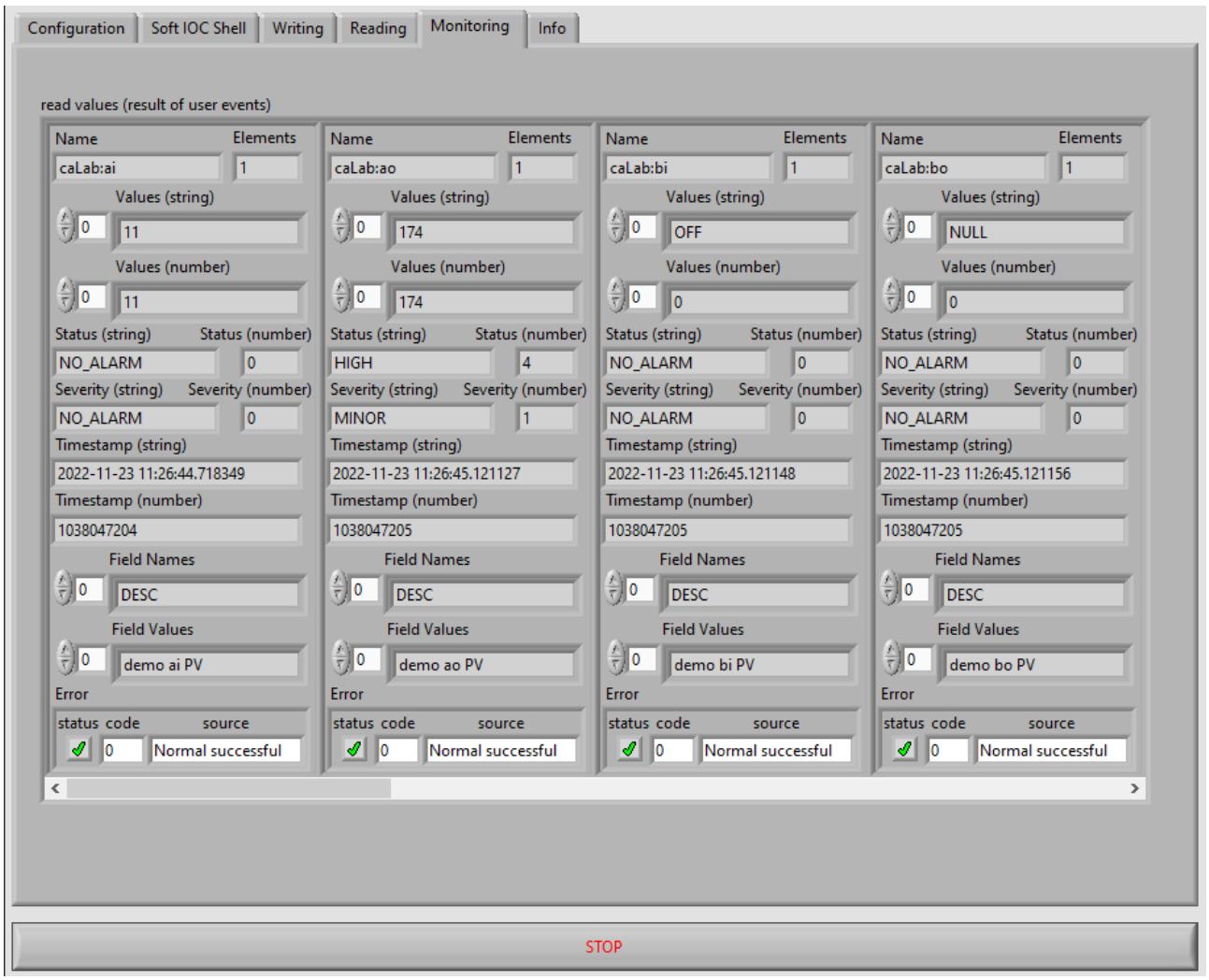
Advanced example with all functions

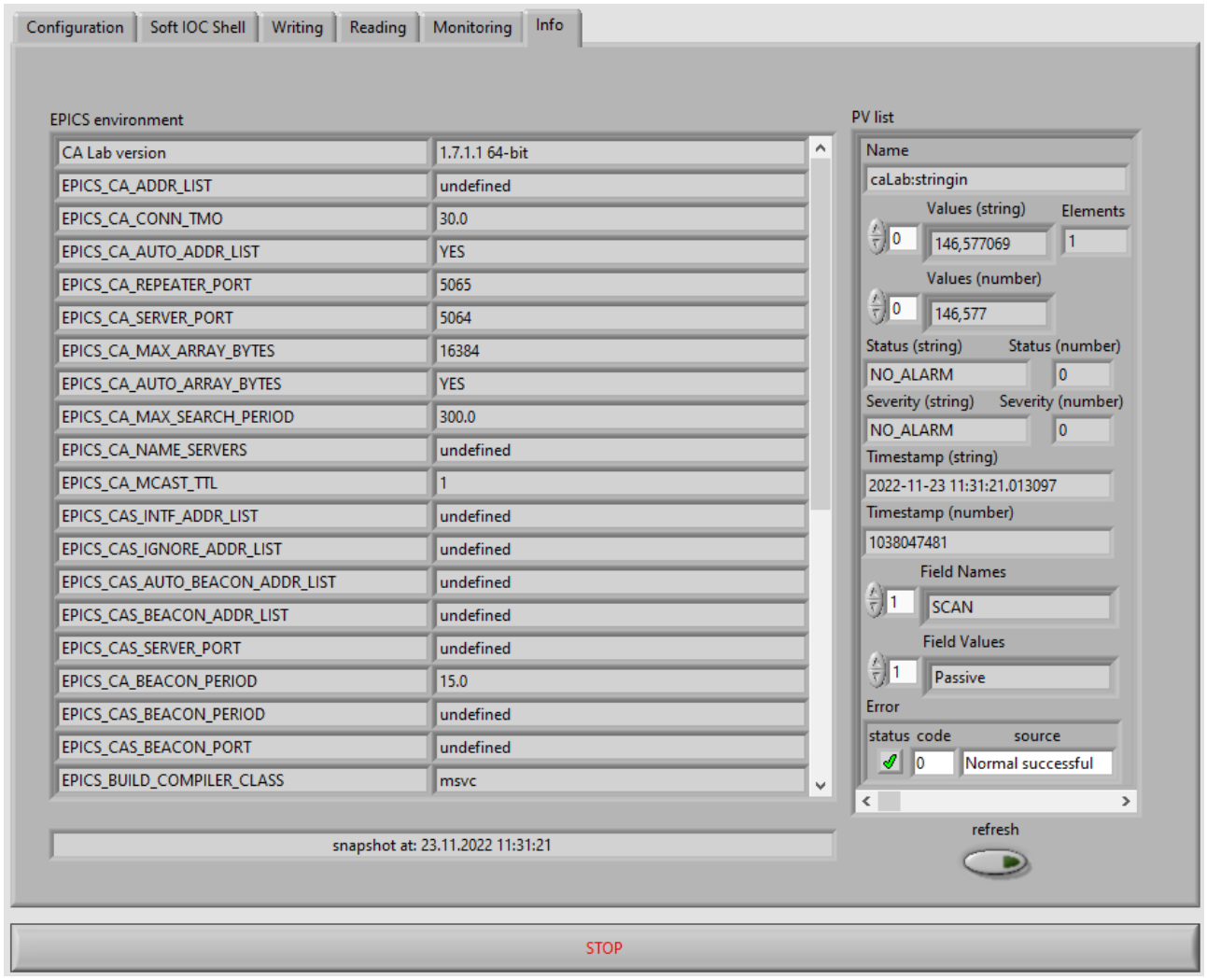


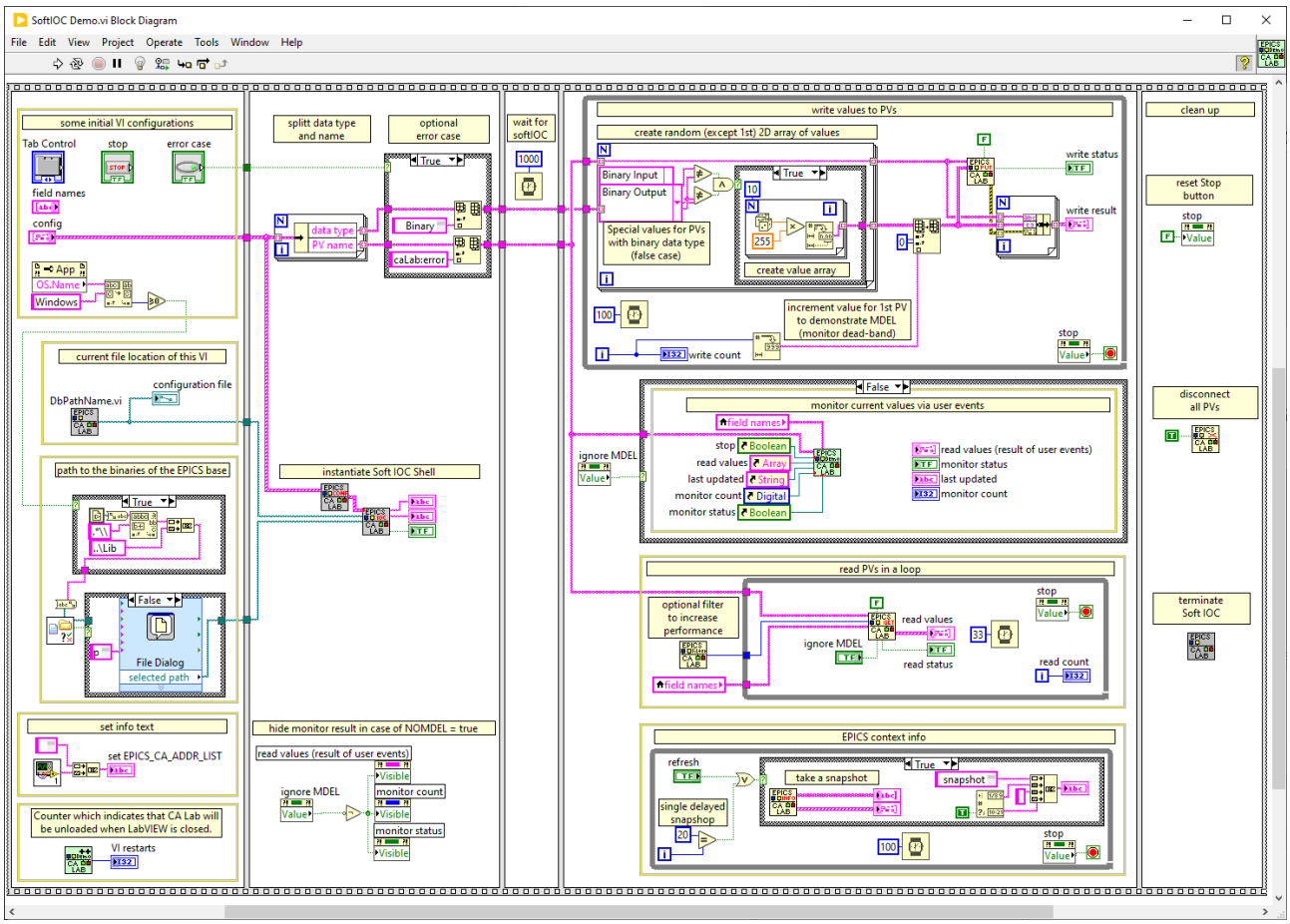

Configuration
Soft IOC Shell
Writing
Reading
Monitoring
Info
This is a copy of the console output of Soft IOC Shell. Here we can check that everything is fine.
dbLoadDatabase("C:\Program Files\National Instruments\LabVIEW 2021\user.lib\caLab\Lib\softIoc.dbd")
softIoc_registerRecordDeviceDriver(pdbbase)
dbLoadRecords("C:\Program Files\National Instruments\LabVIEW 2021\user.lib\caLab\Examples\caLab.db")
iocInit()
EPICS R7.0.4.1
Rev. 2020-09-22T17:28
epics> Starting iocInit
iocRun: All initialization complete
Here is the full command line if you want to start the Soft IOC manually.
Command Line
"C:\Program Files\National Instruments\LabVIEW 2021\user.lib\caLab\Lib\softIoc" -D "C:\Program Files\National Instruments\LabVIEW 2021\user.lib\caLab\Lib\softIoc.dbd" -d "C:\Program Files\National Instruments\LabVIEW 2021\user.lib\caLab\Examples\caLab.db"
STOP


Configuration
Soft IOC Shell
Writing
Reading
Monitoring
Info
names of EPICS variables
written values
errors while writing values
caLab:ai 0 15 status 0 code Normal successful completion source
caLab:ao 0 174,886164 status 0 code Normal successful completion source
caLab:bi 0 0 status 0 code Normal successful completion source
caLab:bo 0 0 status 0 code Normal successful completion source
caLab:longin 0 114,134443 status 0 code Normal successful completion source
caLab:longout 0 64,573134 status 0 code Normal successful completion source
caLab:stringin 0 96,855984 status 0 code Normal successful completion source
caLab:stringout 0 165,022705 status 0 code Normal successful completion source
caLab:waveChar 0 67,079966 status 0 code Normal successful completion source
caLab:waveShort 0 125,598335 status 0 code Normal successful completion source
caLab:waveLong 0 221,955702 status 0 code Normal successful completion source
caLab:waveDouble 0 254,835483 status 0 code Normal successful completion source
caLab:waveString 0 235,458666 status 0 code Normal successful completion source
STOP

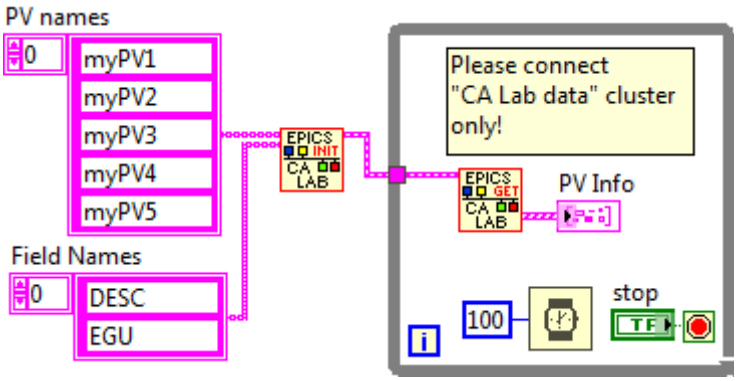

Configuration
Soft IOC Shell
Writing
Reading
Monitoring
Info
read values
Name
Elements
caLab:ai
1
Values (string)
11
Values (number)
11
Status (string)
Status (number)
NO_ALARM
0
Severity (string)
Severity (number)
NO_ALARM
0
Timestamp (string)
2022-11-23 11:26:44.718349
Timestamp (number)
1038047204
Field Names
DESC
Field Values
demo ai PV
Error
status
code
source
Normal successful
caLab:ao
174
174
HIGH
4
MINOR
1
2022-11-23 11:26:45.121127
1038047205
demo ao PV
caLab:bi
OFF
0
2022-11-23 11:26:45.121148
demo bi PV
caLab:bo
NULL
2022-11-23 11:26:45.121156
demo bo PV
Look at "caLab:ai". If "ignore MDEL" is FALSE only differences of more than 10 are notified. This has been defined via "MDEL"-field.
If "ignore MDEL" is TRUE all changes will be visible. It's independently of defined "MDEL"-field. But it causes more network traffic (polling).
The best way is "ignore MDEL" to be set to FALSE and set the "MDEL"-field appropriate.
STOP


Configuration
Soft IOC Shell
Writing
Reading
Monitoring
Info
read values (result of user events)
Name
Elements
caLab:ai
1
Values (string)
11
Values (number)
11
Status (string)
Status (number)
NO_ALARM
0
Severity (string)
Severity (number)
NO_ALARM
0
Timestamp (string)
2022-11-23 11:26:44.718349
Timestamp (number)
1038047204
Field Names
DESC
Field Values
demo ai PV
Error
status
code
source
Normal successful
caLab:ao
174
174
HIGH
4
MINOR
1
2022-11-23 11:26:45.121127
1038047205
demo ao PV
caLab:bi
OFF
0
2022-11-23 11:26:45.121148
demo bi PV
caLab:bo
NULL
2022-11-23 11:26:45.121156
demo bo PV
STOP


Configuration
Soft IOC Shell
Writing
Reading
Monitoring
Info
EPICS environment
CA Lab version 1.7.1.1 64-bit
EPICS_CA_ADDR_LIST undefined
EPICS_CA_CONN_TMO 30.0
EPICS_CA_AUTO_ADDR_LIST YES
EPICS_CA_REPEATER_PORT 5065
EPICS_CA_SERVER_PORT 5064
EPICS_CA_MAX_ARRAY_BYTES 16384
EPICS_CA_AUTO_ARRAY_BYTES YES
EPICS_CA_MAX_SEARCH_PERIOD 300.0
EPICS_CA_NAME_SERVERS undefined
EPICS_CA_MCAST_TTL 1
EPICS_CAS_INTF_ADDR_LIST undefined
EPICS_CAS_IGNORE_ADDR_LIST undefined
EPICS_CAS_AUTO_BEACON_ADDR_LIST undefined
EPICS_CAS_BEACON_ADDR_LIST undefined
EPICS_CAS_SERVER_PORT undefined
EPICS_CA_BEACON_PERIOD 15.0
EPICS_CAS_BEACON_PERIOD undefined
EPICS_CAS_BEACON_PORT undefined
EPICS_BUILD_COMPILER_CLASS msvc
snapshot at: 23.11.2022 11:31:21
PV list
Name
caLab:stringin
Values (string)
Elements
146,577069
Values (number)
146,577
Status (string)
Status (number)
NO_ALARM
Severity (string)
Severity (number)
Timestamp (string)
2022-11-23 11:31:21.013097
Timestamp (number)
1038047481
Field Names
SCAN
Field Values
Passive
Error
status code
source
Normal successful
refresh
STOP


SoftIOC Demo.vi Block Diagram
File Edit View Project Operate Tools Window Help
some initial VI configurations
splitt data type and name
optional error case
wait for softIOC
write values to PVs
create random (except 1st) 2D array of values
clean up
reset Stop button
disconnect all PVs
terminate Soft IOC
current file location of this VI
path to the binaries of the EPICS base
instantiate Soft IOC Shell
monitor current values via user events
read PVs in a loop
EPICS context info
set info text
hide monitor result in case of NOMDEL = true
Counter which indicates that CA Lab will be unloaded when LabVIEW is closed.

Example for preinitialized monitoring of EPICS PVs



Notes and FAQ

Notes:

LabVIEW™ user events with CA Lab must be closed with the new "CaLabEventUnregister" VI, otherwise there is a risk that the app will crash.

Please do not define EPICS variables with the same name but different number or order of field names.

Whenever you use invariant PV names at runtime, set the "variable PV names" input to FALSE. This will increase the performance significantly.

After the start of a CA Lab VI, the EPICS events for all EPICS variables used continue to run in the background. This applies to the entire session of the LabVIEW Runtime. If this is not wanted, then this can be terminated with the CA Lab Disconnect VI.

Linux users should read the README.linux for the setup of EPICS and CA Lab.

FAQ:

Q: I get only correct results for first PV but all others have same/wrong value.
A: When you use several PV names as input series, set input "variable PV names" to TRUE, please.

Q: How can I do a quick check that CA Lab installation works properly?
A: Use the full installion and run "start Demo" from Start menu (Windows® only).

Q: I get a lot of warnings on first start of your VI. What's the problem?
A: The CA Lab VIs need the absolute path to caLab.dll. This absolute path varies from computer to computer. LabVIEW™ looks for the right path at first run and reports that library paths have been changed. You should save these changes.

Q: CA Lab VIs aren't executable, what can I do? (Windows®)
A: Check following:

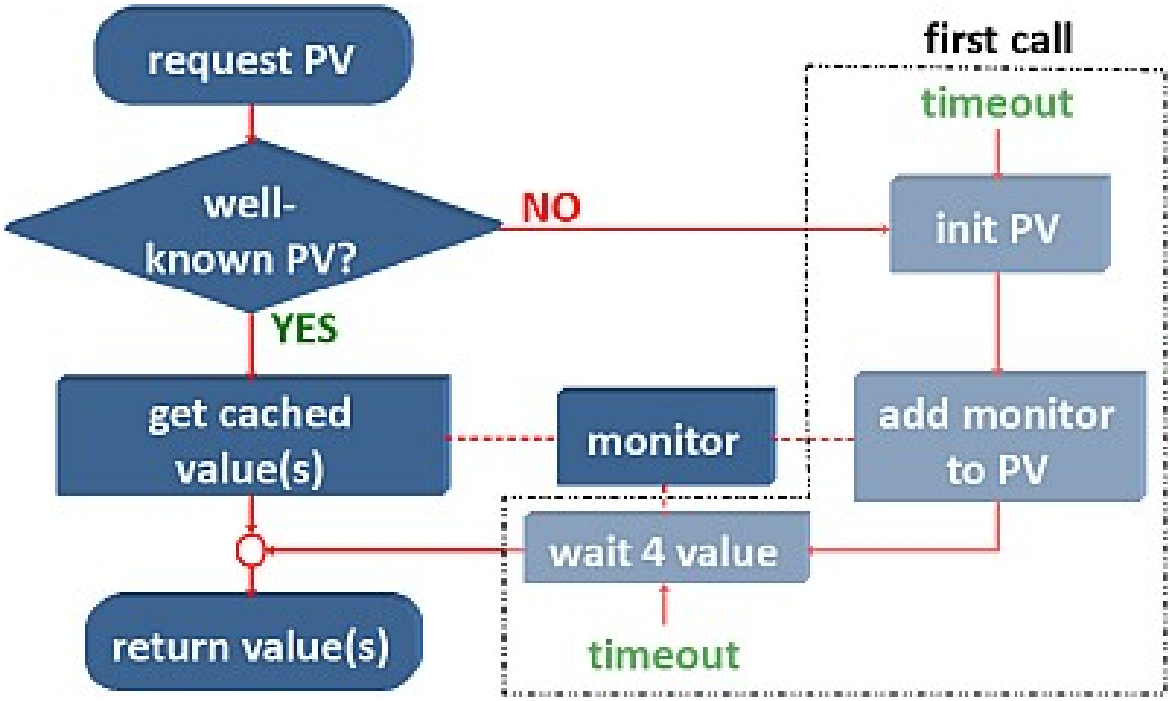
- Search your system for ca.dll, com.dll and carepeater.exe. If you can find several versions of them, then rename all outside of your installation path.
- Check whether Microsoft Visual C++ 2015-2022 Redistributable is installedsuitable for LabVIEW Runtime (32 or 65 bit).
- Open a command line and type "carepeater". You should get no error message. Terminate it with CTRL-C. If you get any error message you should reinstall CA Lab.
- Open caLabGet.vi/caLabPut.vi/caLabEvent.vi and search for text "calab.dll". Double click on founds and check if text field "Library name or path" has same content as "path.txt" in your installation path.

Q: What is the maximum number of PVs monitored, tested up to now?
A: We have simultaneously monitored up to 100.000 PVs. But we're sure the limit is higher. A large number of PVs take significant time to initialize when the PVs are passed separately. It's better to pass the PVs as an array. Reading, writing and monitoring after initialisation is no problem.

Q: Can CA Lab generate its own EPICS channels, or only connect to already defined channels?
A: CA Lab is a pure client application. If you want to generate your own channels, you can use "softIoc.exe". You can find an example in the demo application of caLab. Look at \DemoIOC\db\ demo.db for variable definitions.
UPDATE: Yes, it can do. Install CA Lab 1.5.0.0 or later.

Q: Will caLab work with any version of LabVIEW™ 64bit?
A: Yes, CA Lab does since V1.3.0.3. Please be sure to download the 64 bit version.

Q: I'm wondering what the timeout parameter does.
A: The timeout is the maximum time for initializing and for retrieving the value for the FIRST call. After first call of any PV a background task monitors PVs and every call will be served by cached values from this task.
If you're using "synchronous" for caLabPut.vi, timeout defines how long CA Lab waits for response.

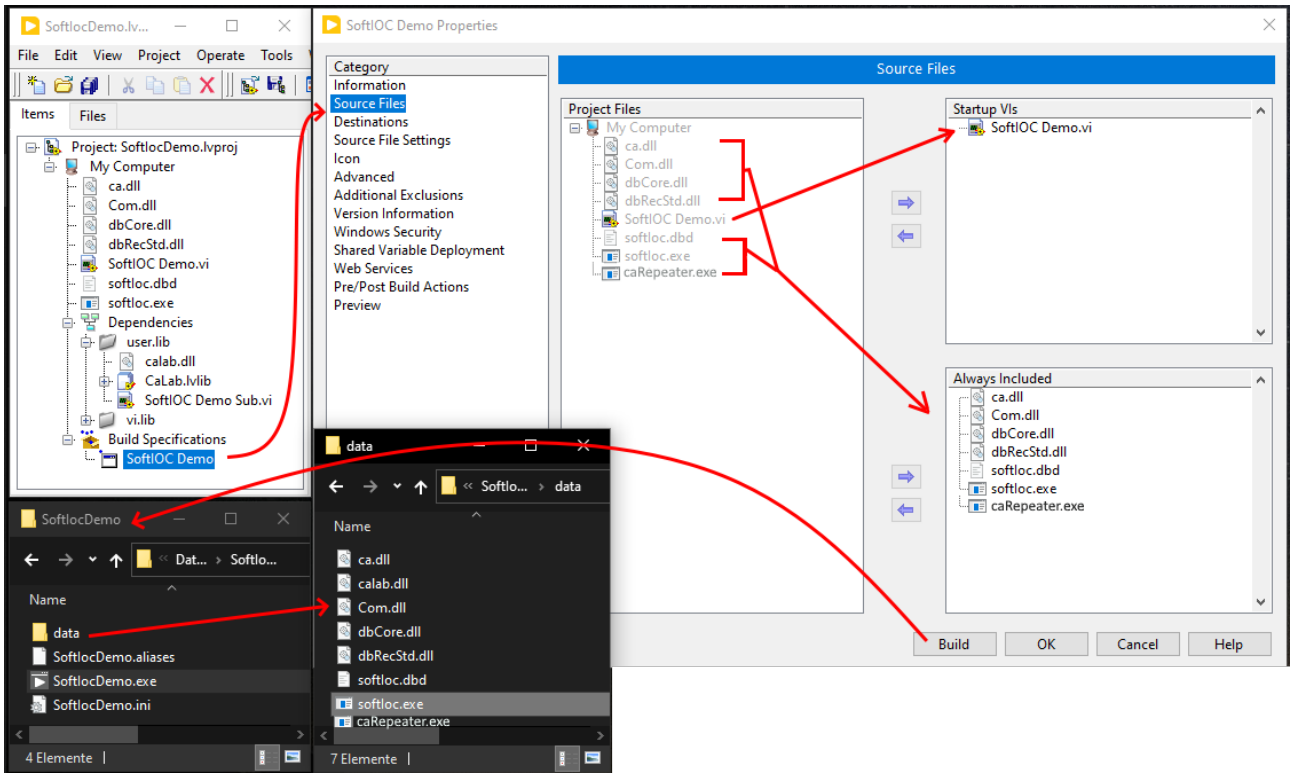


Q: When I use a control (instead of constant) for the field names, changing the field names during program execution doesn't change the returned field values.
A: It is not intended that the field names can be changed during runtime. Please put all required field names into a constant string array and select the data you are interested in in the following code. If the field names are only valid for a part of the variables, this does not matter.

Q: I use caLabPut. When should I set the parameter "synchronous" to TRUE?
A: If it is necessary to wait until all record processing of written PV has been finished before next value can be set then you should set the connector of caLabPut "synchronous"=TRUE. The "Motor Record" is a good example.
Please check, that your timeout is as long as needed for response.

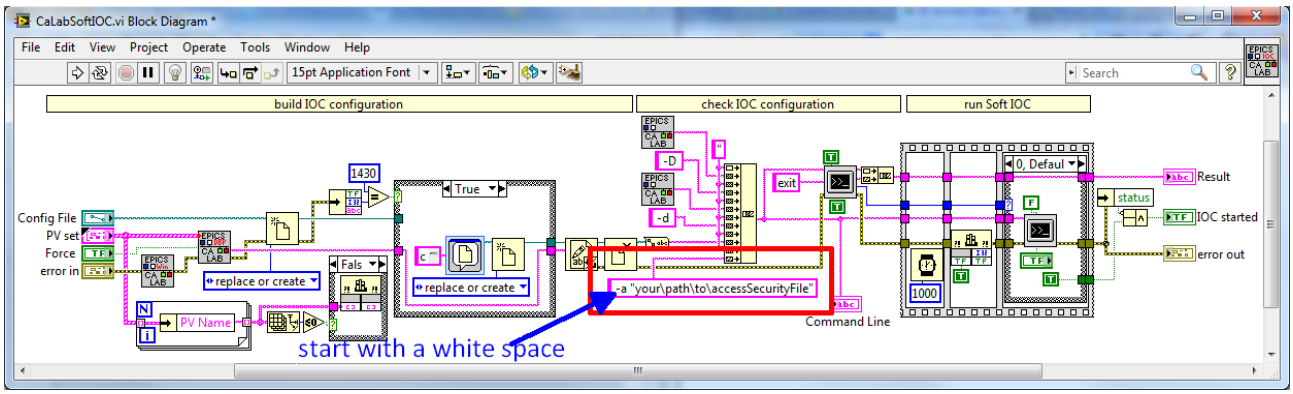
Q: There is a 'Debugging window' with the message "Error: xxx has been configured with different optional fields". What can I do?
A: This message will appear if there are several views of a PV with different field names. Please use the same count and order of field names for same PV name.

Q: I get errors when I compile VIs with CA Lab.
A: Please set your destination folder of executable outside of "user.lib" and check, you have all needed binaries included.



Q: The network and memory load does not decrease as expected when I stop a VI with CA Lab. Is there a memory leak?
A: There is no memory leak. The CA Lab library keeps all PV connections until all instances of CA Lab are closed. This prevents problems of rebuilding Channel Access connections we had in past. If you want to unload all EPICS PVs you have to close all VIs with CA Lab.

Q: How can I add an access security file to CaLabSoftIOC?
A: This quickest solution is to directly add the parameter in CaLabSoftIOC.vi



Q: How can I hide the debug window?
A: Please set environment variable CALAB_NODBG = PATH_NAME_OF_LOG_FILE
All warnings and debug messages will will be redirected to the defined file.

Q: When I use CA Lab there is a huge memory consumption. Why?
A: Please check whether EPICS variables are used by different data types at same connector. Separate the data types per "PV name(s)" connector**.'**

You can also reduce memory consumption by using the optional filter for CA Lab Init / CA Lab Get VI.

Q: Is caLab available for real-time units?
A: The interface works for x86 real-time devices running NI Linux RT.

Q: I always get the following error with a compiled LabVIEW application (exe): "Missing external function calab.dll"
A: Most likely you have installed both the 32-bit and 64-bit versions of CA Lab. Please make sure that the appropriate DLLs (caLab.dll, ca.dll, Com.dll) are used in the LabVIEW application. (see also question "I get errors when I compile VIs with CA Lab.")

Any other questions are welcome! Please send me a mail (carsten.winkler@helmholtz-berlin.de):