Using IDL and Python with EPICS

Mark Rivers, University of Chicago

Argonne National Laboratory

Outline

- **Quick Overview of IDL**
- **ezca library**
- **Calling ezca from IDL**
- **IDL CA API**
- **IDL EPICS class libraries**
- **IDL applications**
- **Overview of Python**
- **Python class libraries**
- **Python applications**
- **Using EPICS from Visual Basic**

Overview of IDL

- **A high-level interpreted *programming language* with vector and array primitives - sort of a cross between BASIC and APL**
- **Modern programming language**
  - Flow control
  - Data structures
  - Objects
- **All operators and most functions work on scalar, vector or array data of any data type.**
- **Data *visualization tool,* advanced built-in graphics**
  - 2-D plots
  - Contour plots
  - Surface plots
  - Shaded surfaces
  - Gray scale/color images
  - Isosurfaces
  - Volume rendering
- **Multi-platform support**
  - Unix: Sun, Hewlett Packard, Silicon Graphics, IBM
  - Linux
  - Microsoft Windows
  - Mac Darwin
- **List price: ~$3,000 on workstations, ~$1,500 on PC, Mac**

Pioneering Science and Technology

**Office of Science
U.S. Department
of Energy**

Overview of IDL

- **Can call external C or other code**
- **Very fast for array operations, as fast as compiled languages**
- **GUI builder**
- **Multi-threaded**
- **Good vendor support**
- **IDL GUI applications can be run at no cost (IDL 6.0 and above)**
  - Must have license to use IDL command line

Overview of IDL

Data Structures

- **A variable in IDL has both a structure and a data type associated with it. Both of these are dynamic, i.e. they can be changed via an assignment statement at any time.**
- **Data types**
    - Byte (b bit, unsigned)
    - Integer (16 bit, signed)
    - Long (32 bit, signed)
    - Float (32 bit floating point)
    - Double (64 bit floating point)
    - Complex (pair of 32 bit floats)
    - Double complex (pair of 64 bit floats)
    - String (0 to 64k characters)
- **Data Structures**
    - Scalar
    - Vector
    - Array - up to 7 dimensions
    - Structure - composed of other elements, like C
- **The sizes of arrays are limited only by the available virtual memory.**

Pioneering Science and Technology

**Office of Science
U.S. Department
of Energy**

Overview of IDL

Assignment Statements

    A = B + 1

- **A has the same structure as B, with a data type equal to that of the most precise operand in the expression on the right hand side. In this case it could be any type except string.**
- **If B is a vector or array then 1 is added to each element.**

```
A = 0           ; A is a 16 bit integer
A = A * 0.5     ; A is now a 32 bit float
B = A(*,3)      ; B is equal to the 4th row of A
A(*,3) = 0      ; Set all elements in 4th row of A equal to 0
```

Syntax

- **Examples:**

```
image = fltarr(512, 512)        ; zero filled array
b = image(0:127, 0:127)         ; b is 128x128 array
image(*,100) = findgen(512)     ; replace row 100
plot, image(*,120)              ; plot row 121
; Display the power spectrum as an image
tvscl, alog(abs(fft(image, 1)))
```
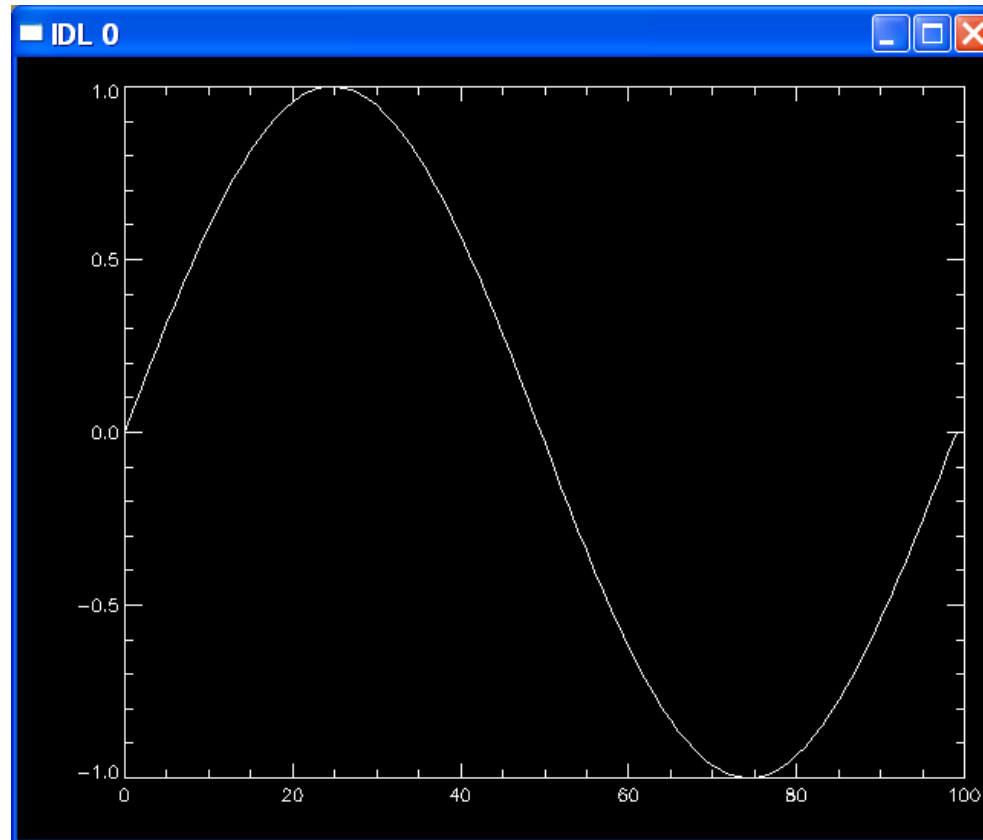
IDL Examples

```
IDL> a = sin(findgen(100)/99. * 2 * !pi)
IDL> help, a
A                 FLOAT        = Array[100]
IDL> plot, a
```
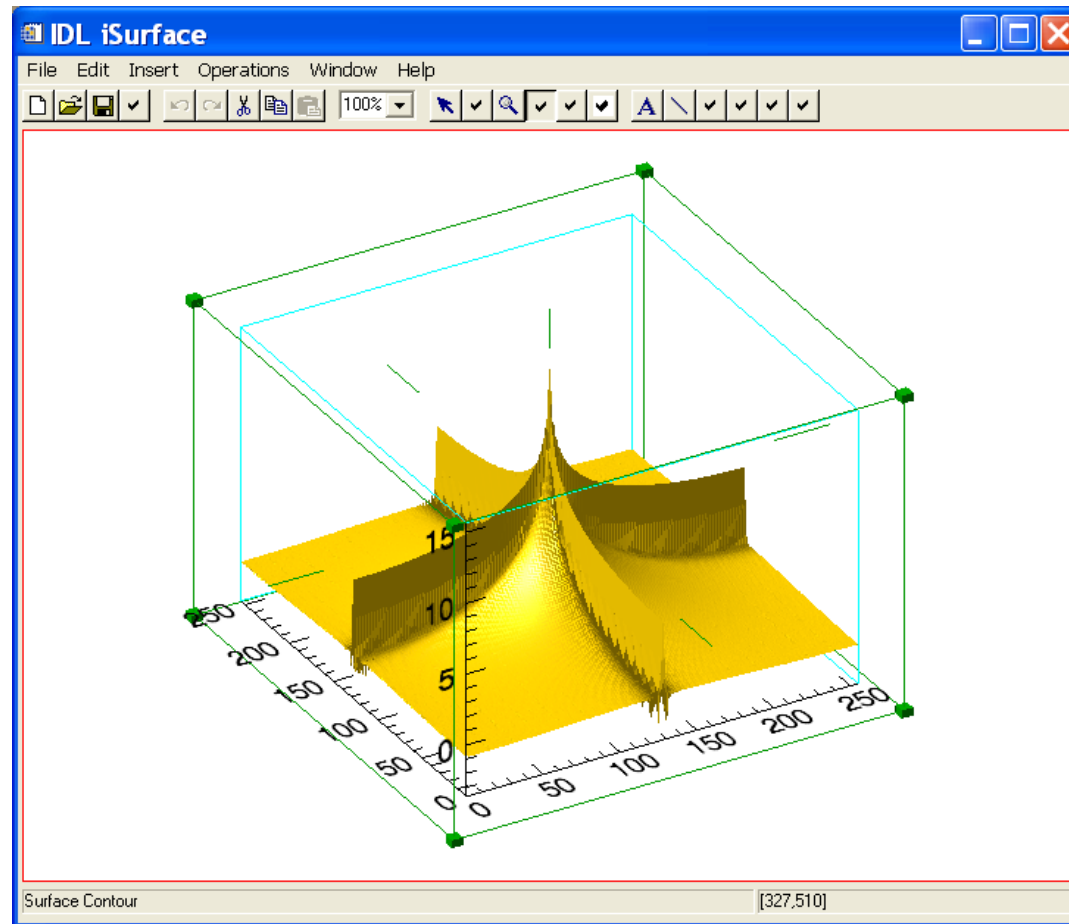
IDL Examples

```
IDL> a = shift(alog(abs(fft(dist(256),1))),128,128)
IDL> isurface, a
```

ezca and EzcaScan

- **EPICS extensions for "Easy Channel Access"**
  - Don't need to handle chids, just use PV name strings – hash table
  - Synchronous APIs - applications don't have to handle callbacks
- **Ezca (partial list)**

```
epicsShareFunc int epicsShareAPI ezcaGet(char *pvname, char ezcatype,
        int nelem, void *data_buff);
epicsShareFunc int epicsShareAPI ezcaPut(char *pvname, char ezcatype,
        int nelem, void *data_buff);
epicsShareFunc int epicsShareAPI ezcaPutOldCa(char *pvname, char ezcatype,
        int nelem, void *data_buff);
epicsShareFunc int epicsShareAPI ezcaNewMonitorValue(char *pvname,
        char ezcatype); /* returns TRUE/FALSE */
epicsShareFunc int epicsShareAPI ezcaSetTimeout(float sec);
epicsShareFunc float epicsShareAPI ezcaGetTimeout(void);
epicsShareFunc int epicsShareAPI ezcaSetRetryCount(int retry);
epicsShareFunc int epicsShareAPI ezcaGetRetryCount(void);
epicsShareFunc int epicsShareAPI ezcaPvToChid(char *pvname, chid **cid);
epicsShareFunc int epicsShareAPI ezcaSetMonitor(char *pvname, char ezcatype);
epicsShareFunc int epicsShareAPI ezcaClearMonitor(char *pvname, char ezcatype);
epicsShareFunc int epicsShareAPI ezcaStartGroup(void);
epicsShareFunc int epicsShareAPI ezcaEndGroup(void);
epicsShareFunc int epicsShareAPI ezcaGetControlLimits(char *pvname,
        double *low, double *high);
epicsShareFunc int epicsShareAPI ezcaGetGraphicLimits(char *pvname,
        double *low, double *high);
epicsShareFunc int epicsShareAPI ezcaGetNelem(char *pvname, int *nelem);
epicsShareFunc int epicsShareAPI ezcaGetPrecision(char *pvname,
        short *precision);
epicsShareFunc int epicsShareAPI ezcaGetStatus(char *pvname,
        TS_STAMP *timestamp, short *status, short *severity);
epicsShareFunc int epicsShareAPI ezcaGetUnits(char *pvname,
        char *units); /* units must be at least EZCA_UNITS_SIZE large */
```

**Pioneering
Science and
Technology**

**Office of Science
U.S. Department
of Energy**

ezca and EzcaScan

- ## EzcaScan (partial list)

  - ```
    epicsShareFunc int epicsShareAPI Ezca_getArray(int
    noName,char **pvName,int type,int nodata,void *value);
    ```

  - ```
    epicsShareFunc int epicsShareAPI Ezca_getArrayEvent(int
    noName,char **pvName,int type,int nodata,void *value);
    ```

  - ```
    epicsShareFunc int epicsShareAPI Ezca_putArray(int
    noName,char **pvName,int type ,int nodata,void *value);
    ```

  - ```
    epicsShareFunc int epicsShareAPI Ezca_putArrayEvent(int
    noName,char **pvName,int type,int nodata,void *value);
    ```

ezca and IDL

- **IDL can call "shareable libraries", e.g. .so files on Unix, .dll files on Windows**

- **The argument passing convention is fixed, it is not compatible with ezca.dll directly**

- **Need a thin glue layer between IDL and ezca/EzcaScan**

- **ezcaIDL is the glue layer. Mostly just changes calling conventions. Provides a few functions that ezca and EzcaScan do not. Use the ezcaPvToChid() function.**

    - ezcaIDLGetEnumStrings()

    - ezcaIDLGetCountAndType()

IDL Channel Access API

Routines which return information about process variables

```
Status = caGet(pvname, value, /string, maximum=max)
Status = caGetControlLimits(pvname, low, high)
Status = caGetGraphicLimits(pvname, low, high)
Status = caGetPrecision(pvname, precision)
Status = caGetStatus(pvname, timestamp, status, severity)
Status = caGetUnits(pvname, units)
Status = caGetEnumStrings(pvname, strings)
Status = caGetCountAndType(pvname, count, type)
```

Routines which write new values to process variables

```
Status = caPut(pvname, value, wait=wait)
```

Routines which control channel access timeouts

```
Timeout = caGetTimeout()
caSetTimeout, timeout
RetryCount = caGetRetryCount()
caSetRetryCount, retrycount
```

IDL Channel Access API

Routines which control synchronous groups

```
caStartGroup

stat = caEndGroup(status)
```

Routines which control channel access monitors

```
Status = caSetMonitor(pvname)

Status = caClearMonitor(pvname)

State = caCheckMonitor(pvname)
```

Routines which control debugging and error messages

```
caDebug, state

caTrace, state

caError, err_string, /ON, /OFF, /PRINT, prefix=prefix
```
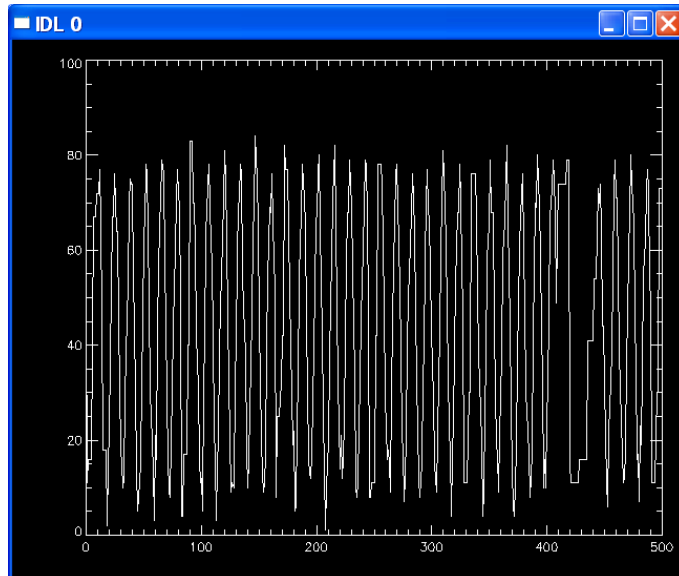
Documentation at

http://cars.uchicago.edu/software/idl/ezcaIDLGuide.html

http://cars.uchicago.edu/software/idl/ezcaIDLRef.html

IDL EPICS Examples

```
IDL> status = caget('13LAB:m1.VAL', position)
IDL> help, status, position
STATUS          LONG      =           0
POSITION        DOUBLE    =      517.19305
IDL> status = caget('13LAB:quadEM:mca1', spectrum)
IDL> plot, spectrum
IDL> help, status, spectrum
STATUS          LONG      =           0
SPECTRUM        LONG      = Array[2048]
IDL> plot, spectrum[0:500]
IDL> fft_data = alog(abs(fft(spectrum,1)))
IDL> plot, fft_data, xrange=[0,1023]
```

Pioneering Science and Technology

Office of Science U.S. Department of Energy
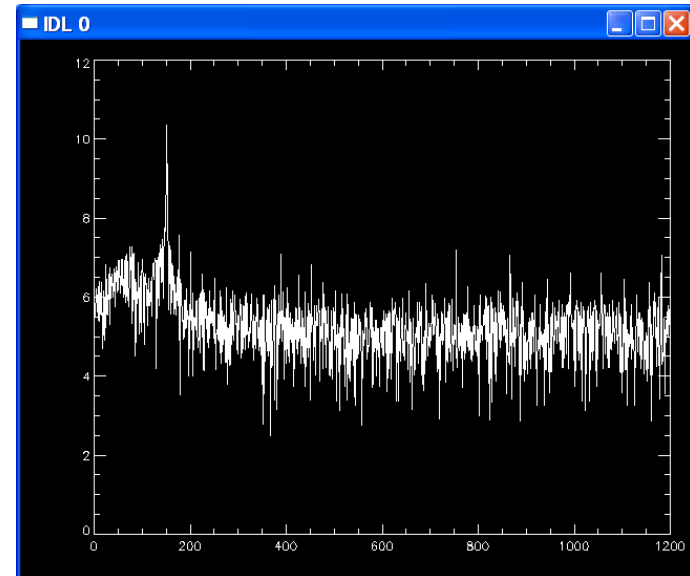
IDL EPICS Examples

Move a motor

```
DL> status = caput('13LAB:m8.VAL', 10000)
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
      215.52734
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
      835.64453
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
      1795.6055
```

Ezca timeout values are important!

```
IDL> print, cagettimeout()
    0.0200000
IDL> t0=systime(1)&for i=1,100 do t=caget('13LAB:m1', v)&print, systime(1)-t0
      2.9898720
IDL> casettimeout, .001
IDL> t0=systime(1)&for i=1,100 do t=caget('13LAB:m1', v)&print, systime(1)-t0
      0.21649790
```

IDL EPICS Examples

Using monitors

Monitored channels read the cached values on caget()

Can check whether a monitor has happened (a Channel Access value callback)

```
IDL> status = caSetMonitor('13LAB:m8.DMOV')
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =           1
IDL> status = caget('13LAB:m8.DMOV', done)
IDL> help, done
DONE            INT       =           1
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =           0
IDL> status = caput('13LAB:m8.VAL', 0)
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =           1
IDL> status = caget('13LAB:m8.DMOV', done)
IDL> help, state
STATE           LONG      =           1
IDL> help, done
DONE            INT       =           0
```

Monitors are useful for seeing that a PV changed state, even if its value is the same because one "missed" the transition. For example, PV goes 0->1->0. IDL polling might miss the one state, but checking a monitor would let one know that it happened.

Pioneering
Science and
Technology

Office of Science
U.S. Department
of Energy

IDL EPICS Class Libraries

- **IDL object classes that hide the underlying EPICS process variables**

- **IDL objects treat all data as private, only accessible through methods.**

- **Provide an object-oriented interface to common beamline objects (motors, scalers, mcas, scans)**

  - epics_motor

  - epics_scaler

  - epics_mca (inherits device-independent mca class)

  - epics_med (multi-element detector)

  - epics_sscan

- **Example of epics_motor**

  ```
  IDL> motor = obj_new('EPICS_MOTOR', '13LAB:m8')
  IDL> motor->move, 1000.    ; Move to absolute position 10.
  IDL> motor->wait           ; Wait for it to get there
  ```

Pioneering Science and Technology

Office of Science U.S. Department of Energy

IDL EPICS Class Libraries

- ## Example of epics_mca

```
IDL> mca = obj_new('epics_mca', '13LAB:aim_adc1')
IDL> mca->erase
IDL> mca->acquire_on
IDL> data = mca->get_data()
IDL> plot, data
```
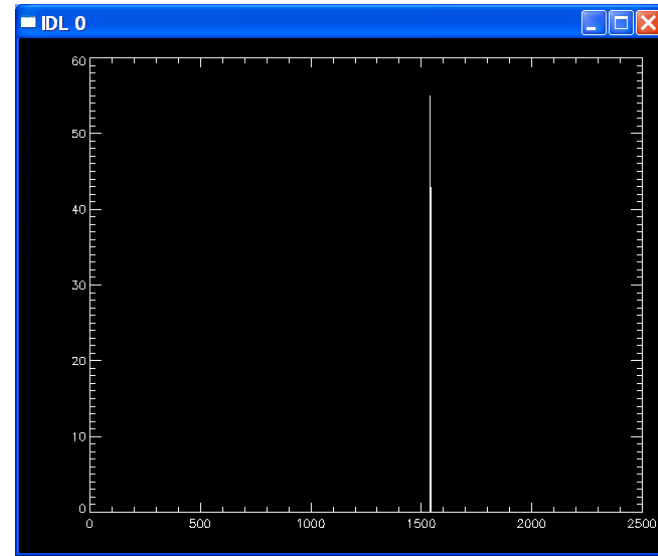

- ## Example of epics_scaler

```
IDL> scaler = obj_new('epics_scaler', '13LAB:scaler1')
IDL> scaler->start, 10.    ; Count for 10 seconds
IDL> scaler->wait          ; Wait for it to get done
IDL> counts = scaler->read(); Read the counts on each channel
IDL> print, counts
     100000000     0      0      0      0      0      0      0
```

IDL EPICS Class Libraries

- **epics_sscan**

- **Designed to do the following:**

  - Provide an object-oriented interface to standard EPICS scans, enabling user written software to easily access scan header information and data.

  - Provide an easy way to read MDA files written by the saveData function in synApps.

  - Provide an easy way to get scan data into the IDL iTools system. iTools provide powerful interfaces for visualizing data, zooming in, adding annotation, and producing publication quality plots.

  - Provide a way to convert binary scan files (e.g. MDA) into ASCII

  - Does not currently communicate with the IOC for real-time data, but this is planned for the future

IDL EPICS Class Libraries

Example: Simple 1D epics_sscan

```
IDL> s = read_mda('13IDC_0027.mda') ; Read the data
IDL> s->display   ; Display the first detector
```
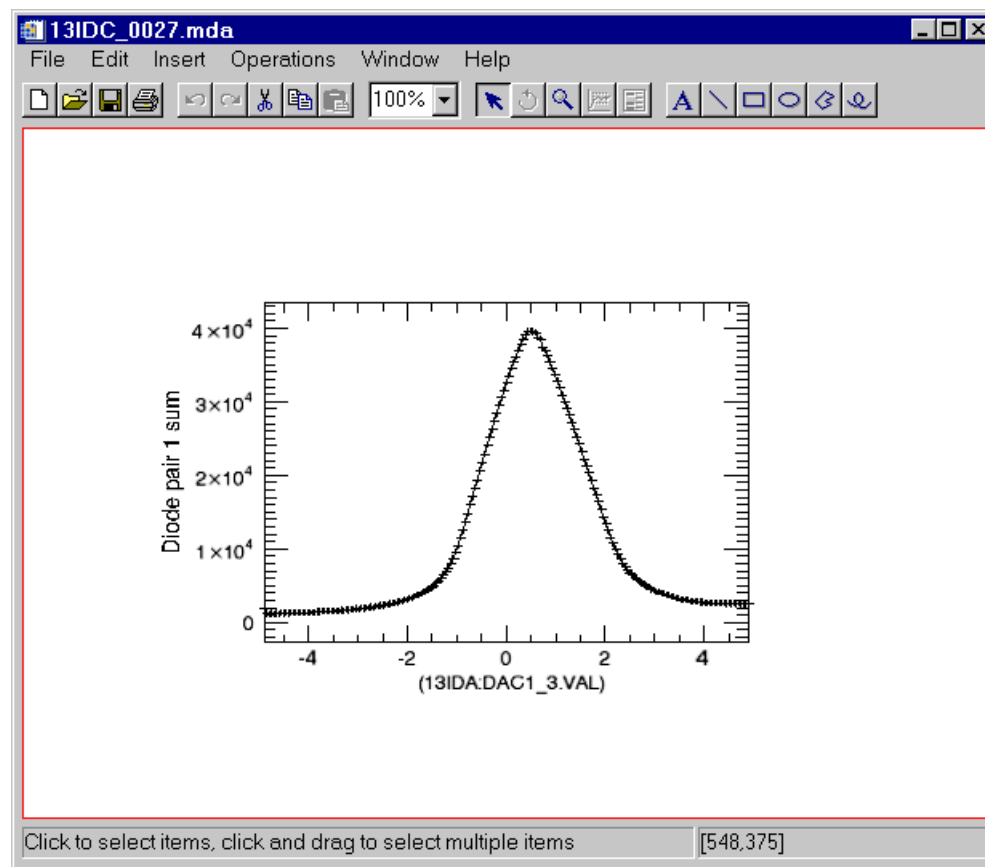
IDL EPICS Class Libraries

Example: 2-D epics_sscan

```
IDL> s=read_mda('2idd_0087.mda') ; Read the 2-D dataset
IDL> s->display, /all, /grid ; Display all of the
     images in a grid
```
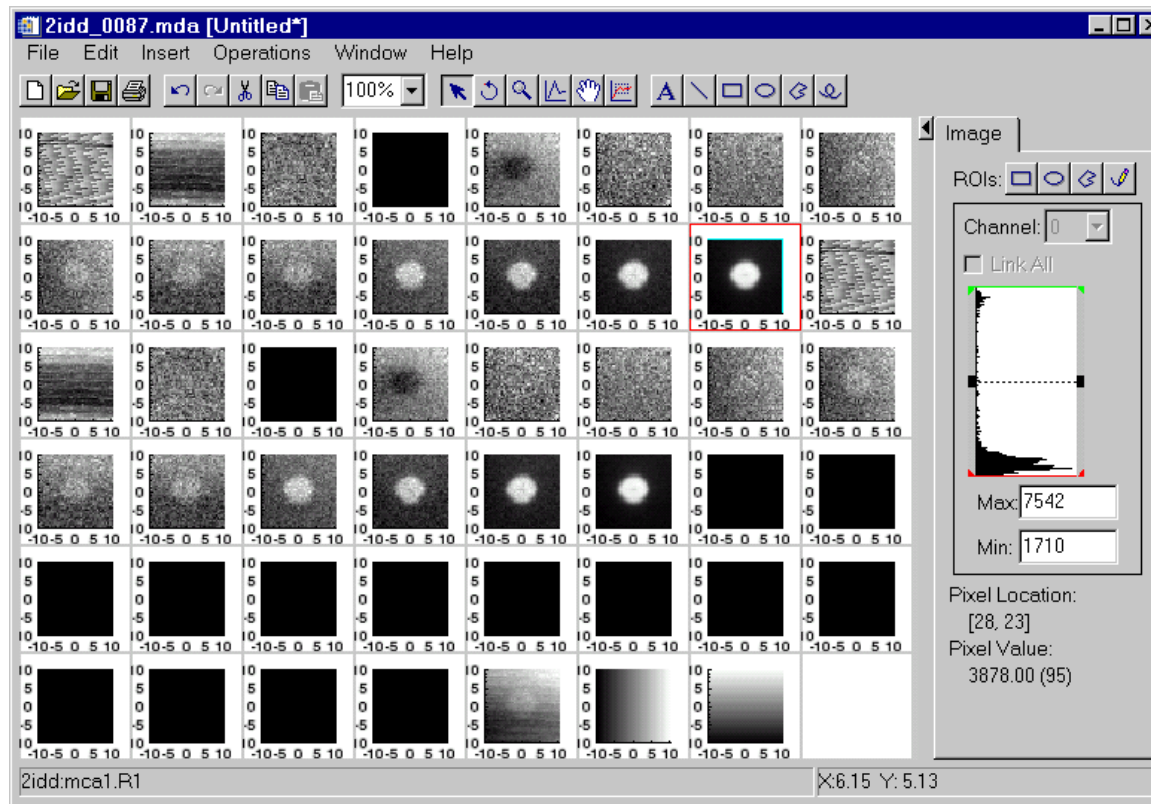
IDL EPICS Class Libraries

Example: 2-D epics_sscan

```
; Plot a profile of column 20 (X=20) in detector 15.
IDL> s->display, detector=15, xrange=20
```
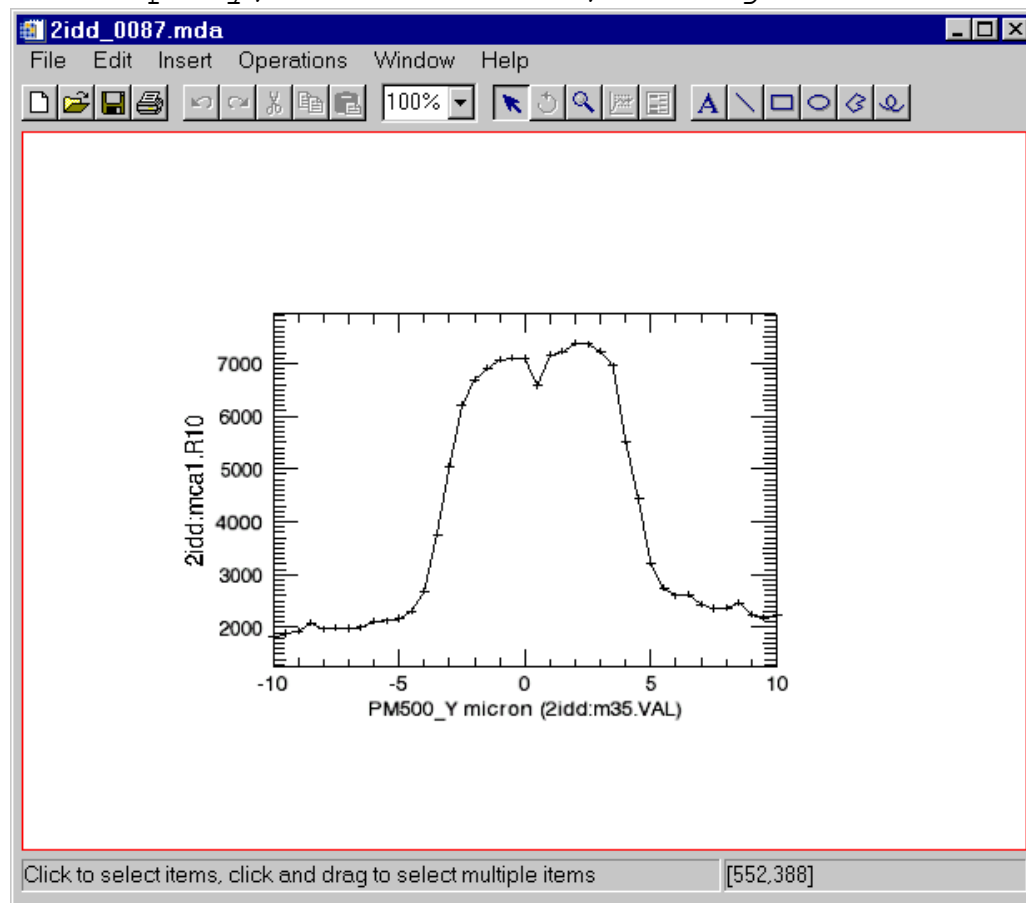
IDL EPICS Class Libraries

Documentation: Reference manual for each class library

EPICS_MCA Class

This page was created by the IDL library routine mk_html_help**. For more information on this routine, refer to the IDL Online Help Navigator or type:**

? mk_html_help **at the IDL command line prompt.**

Last modified: Sat Jul 14 10:16:05 2001.

List of Routines

- EPICS_MCA::ACQUIRE_OFF
- EPICS_MCA::ACQUIRE_ON
- EPICS_MCA::ACQUIRE_WAIT
- EPICS_MCA::ADD_ROI
- EPICS_MCA::DEL_ROI
- EPICS_MCA::ERASE
- EPICS_MCA::GET_ACQUIRE_STATUS
- EPICS_MCA::GET_CALIBRATION
- EPICS_MCA::GET_DATA
- EPICS_MCA::GET_ELAPSED
- EPICS_MCA::GET_PRESETS
- EPICS_MCA::GET_ROIS
- EPICS_MCA::GET_ROI_COUNTS
- EPICS_MCA::GET_SEQUENCE
- EPICS_MCA::INIT
- EPICS_MCA::SET_CALIBRATION
- EPICS_MCA::SET_DATA
- EPICS_MCA::SET_PRESETS
- EPICS_MCA::SET_ROIS
- EPICS_MCA::SET_SEQUENCE
- EPICS_MCA::SPECTRA_SCAN
- EPICS_MCA::WRITE_FILE
- EPICS_MCA__DEFINE
- RELEASE_NOTES

Documentation at:

http://cars9.uchicago.edu/software/idl/

EPICS_MCA::GET_DATA

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      EPICS_MCA::GET_DATA

PURPOSE:
      This function returns the data from the MCA.

CATEGORY:
      EPICS device class library.

CALLING SEQUENCE:
      Result = epics_mca->GET_DATA()

KEYWORD_PARAMETERS:
      CHECK_NEW:
          A flag which indicates that this routine should only return
          the data if it has changed.

OPTIONAL OUTPUTS:
      NEW_FLAG:
          If CHECK_FLAG is set, then NEW_FLAG will be 1 if the function
          is returning new data, 0 if the function is not returning new
          data.  If CHECK_FLAG is set and NEW_FLAG is 0 then the function
          returns -1.
PROCEDURE:
      This function reads the data from the hardware using the EPICS MCA
      record, and then invokes MCA::GET_DATA

ADDITIONAL INFORMATION:
      See MCA::GET_DATA().

MODIFICATION HISTORY:
      Written by:    Mark Rivers, October 1, 1997
      Nov. 14, 1997  Mark Rivers.  Changed routine to eliminate setting
                     rec.READ back to 0, since record support does this
                     automatically and it was causing record to process
                     again.
      19-Sep-1998 MLR  Added /WAIT to caput, since default is not to wait
                        for callback now.
      17-Mar-1999 MLR  Removed /WAIT from caput, to be compatible with
                        version 4.3 and later of the MCA record, which does
                        not fire forward links until acquisition is complete.
      28-Mar-1999 MLR  Changed routine so it no longer pokes READ field.
                        This assumes that someone else (typically a database)
                        is periodically poking the READ field.  The object
                        initialization code now sets a monitor on the VAL
                        field. Added New_flag output and CHECK_NEW keyword.
```
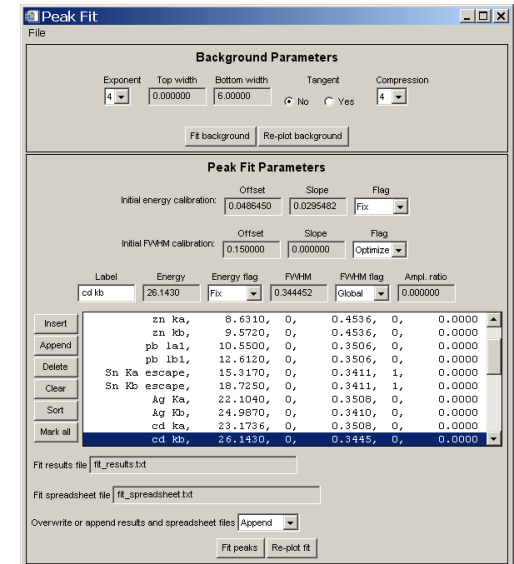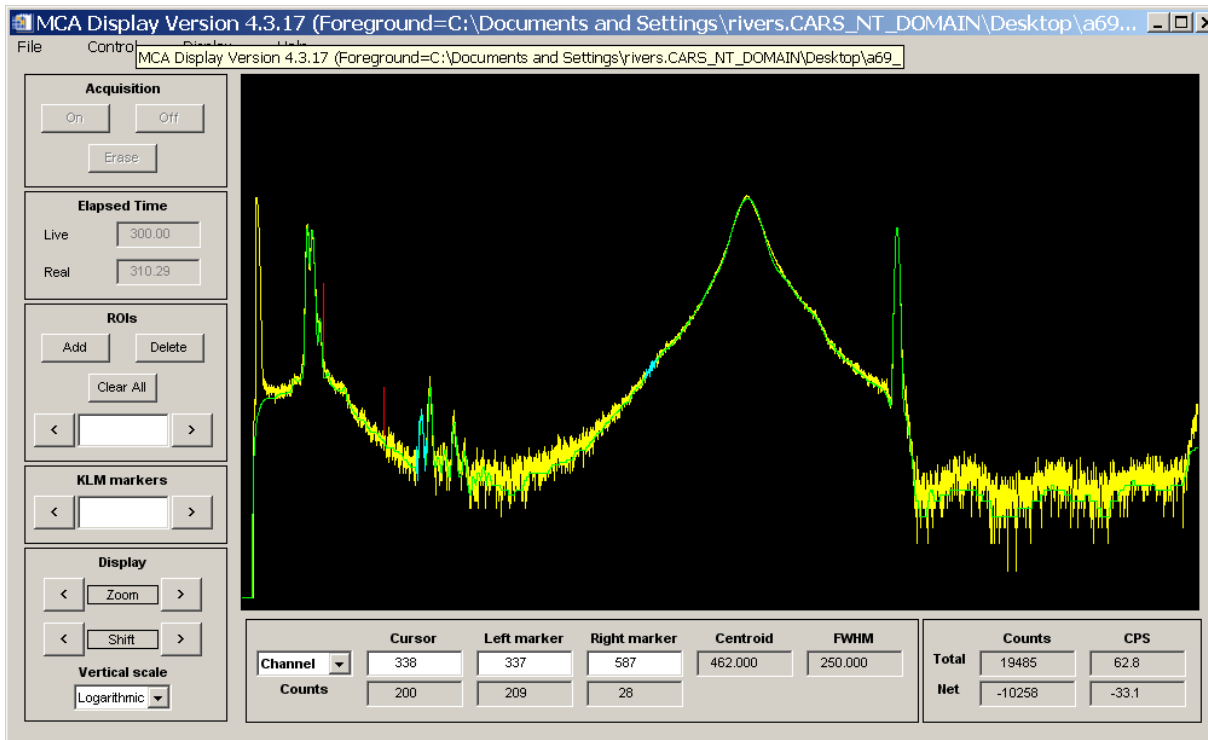
(See epics_mca__define.pro)

Pioneering Science and Technology

Office of Science U.S. Department of Energy

IDL EPICS Applications

- **mcaDisplay**
    - Full-featured program for displaying, controlling EPICS multi-channel analysers, including peak fitting
    - Uses epics_mca class library, and exports mca_display class, so it can be controlled by other IDL applications

IDL EPICS Applications

Data catcher and data viewer (Ben-Chin Cha)

Pioneering Science and Technology

Office of Science U.S. Department of Energy
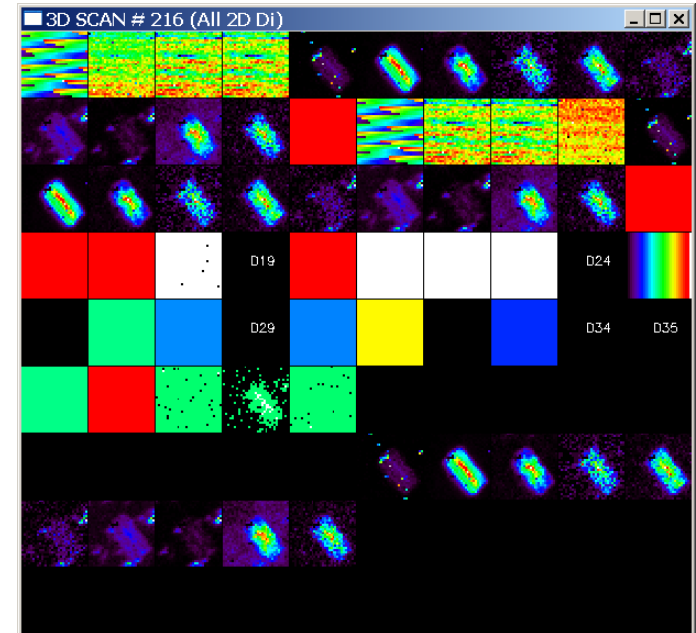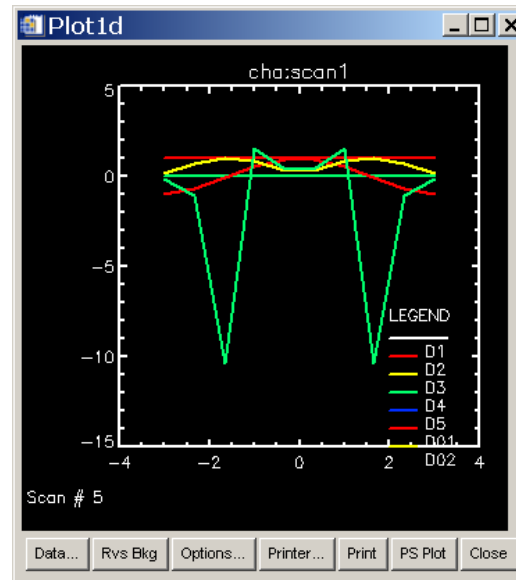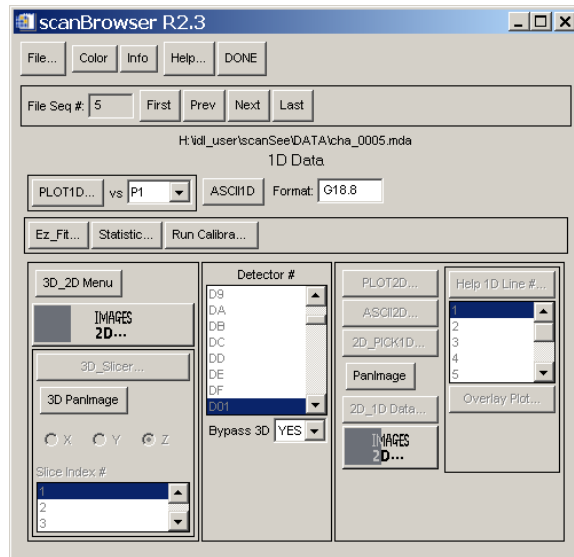
Using EPICS from Visual Basic

- **ezca.dll can be called directly from Visual Basic on Windows**
- **ezca.bas provides the interface**

```
Public Const ezcaByte   As Byte = 0
Public Const ezcaString As Byte = 1
Public Const ezcaShort  As Byte = 2
Public Const ezcaLong   As Byte = 3
Public Const ezcaFloat  As Byte = 4
Public Const ezcaDouble As Byte = 5

Public Declare Function ezcaGet Lib "ezca.dll" _
    (ByVal pvname As String, _
     ByVal ezcatype As Byte, _
     ByVal nelem As Long, _
     ByRef data As Any) As Long


Public Declare Function ezcaPut Lib "ezca.dll" Alias "ezcaPutOldCa" _
    (ByVal pvname As String, _
     ByVal ezcatype As Byte, _
     ByVal nelem As Long, _
     ByRef data As Any) As Long

Public Declare Function ezcaPutString Lib "ezca.dll" Alias "ezcaPutOldCa" _
    (ByVal pvname As String, _
     ByVal ezcatype As Byte, _
     ByVal nelem As Long, _
     ByVal data As Any) As Long

Public Declare Function ezcaPutCallback Lib "ezca.dll" Alias "ezcaPut" _
    (ByVal pvname As String, _
     ByVal ezcatype As Byte, _
     ByVal nelem As Long, _
     ByRef data As Any) As Long
```

Using EPICS from Visual Basic

- **Example: tomography data collection. VB used because it can easily control Roper's WinView program for the CCD detector**

Pioneering
Science and
Technology

Office of Science
U.S. Department
of Energy
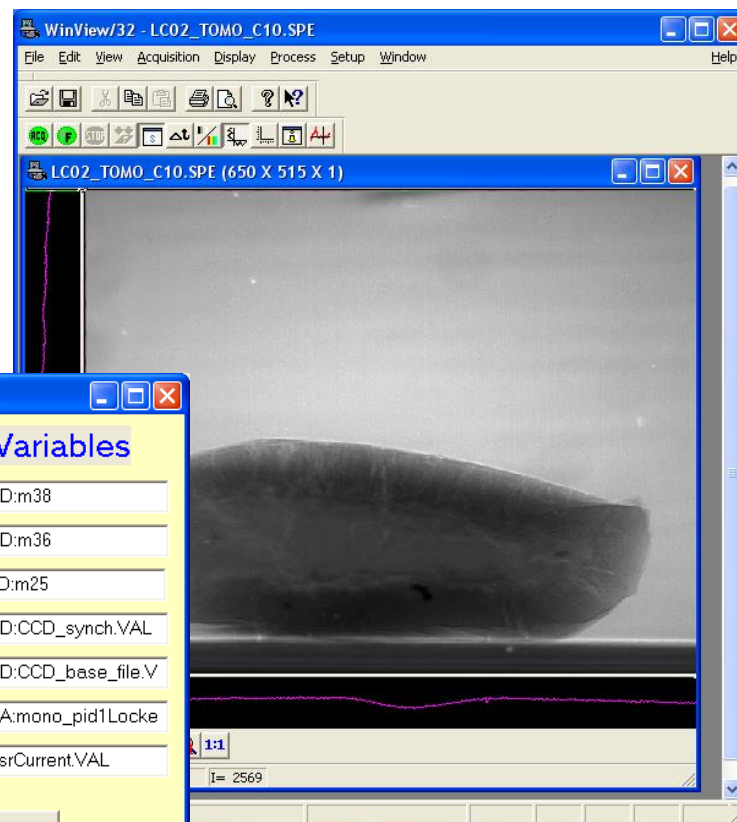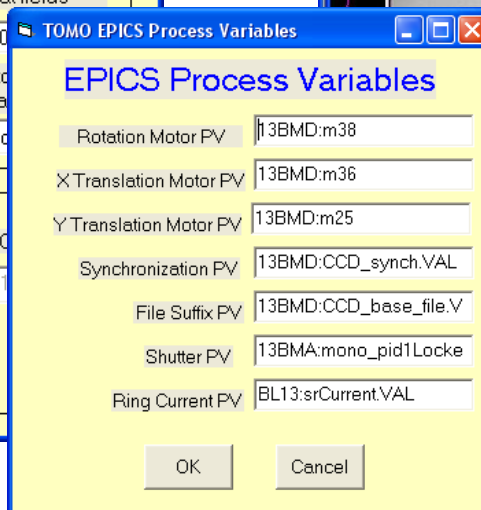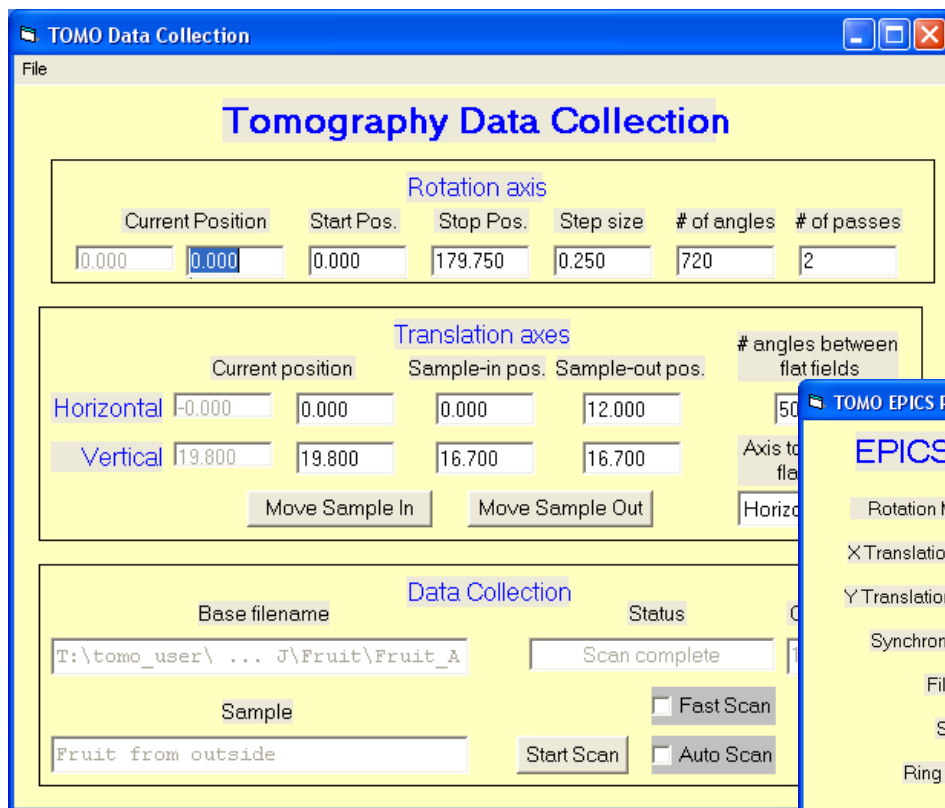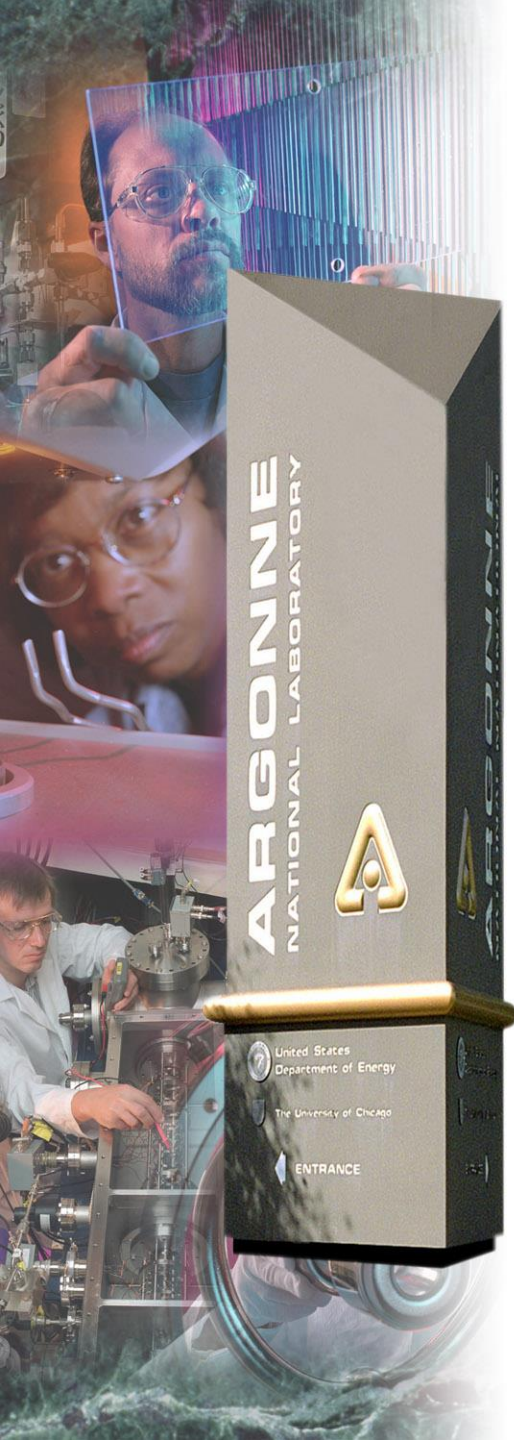
Python Applications for Beamline Control

Mark Rivers

Argonne National Laboratory

A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago

Office of Science
U.S. Department of Energy

Motivation

- **Replace IDL applications (e.g. MCA GUI) with Python so that other beamlines don't need to buy IDL**

- **Send users home with data and display/analysis programs that are free.**

  - They don't want to buy IDL.

Building Blocks

- **CaChannel from Geoff Savage for EPICS interface**

- **Tkinter and Pmw for GUIs**

- **Pmw.Blt for plots**

- **Numeric for arrays**

BltPlot: Enhancements to the Pmw.Blt.Graph widget

- **A standalone plotting widget, BltPlot.BltPlot. This widget has menus to:**
  - Configure all of the plot characteristics
  - Save and restore the plot settings and data
  - Print the plot to a file or printer
- **Methods (BltPlot.plot and BltPlot.oplot) to create a new plot, to overplot more data, etc.**
- **Designed to provide a rough emulation of the command line plotting capabilities of IDL.**

BltPlot
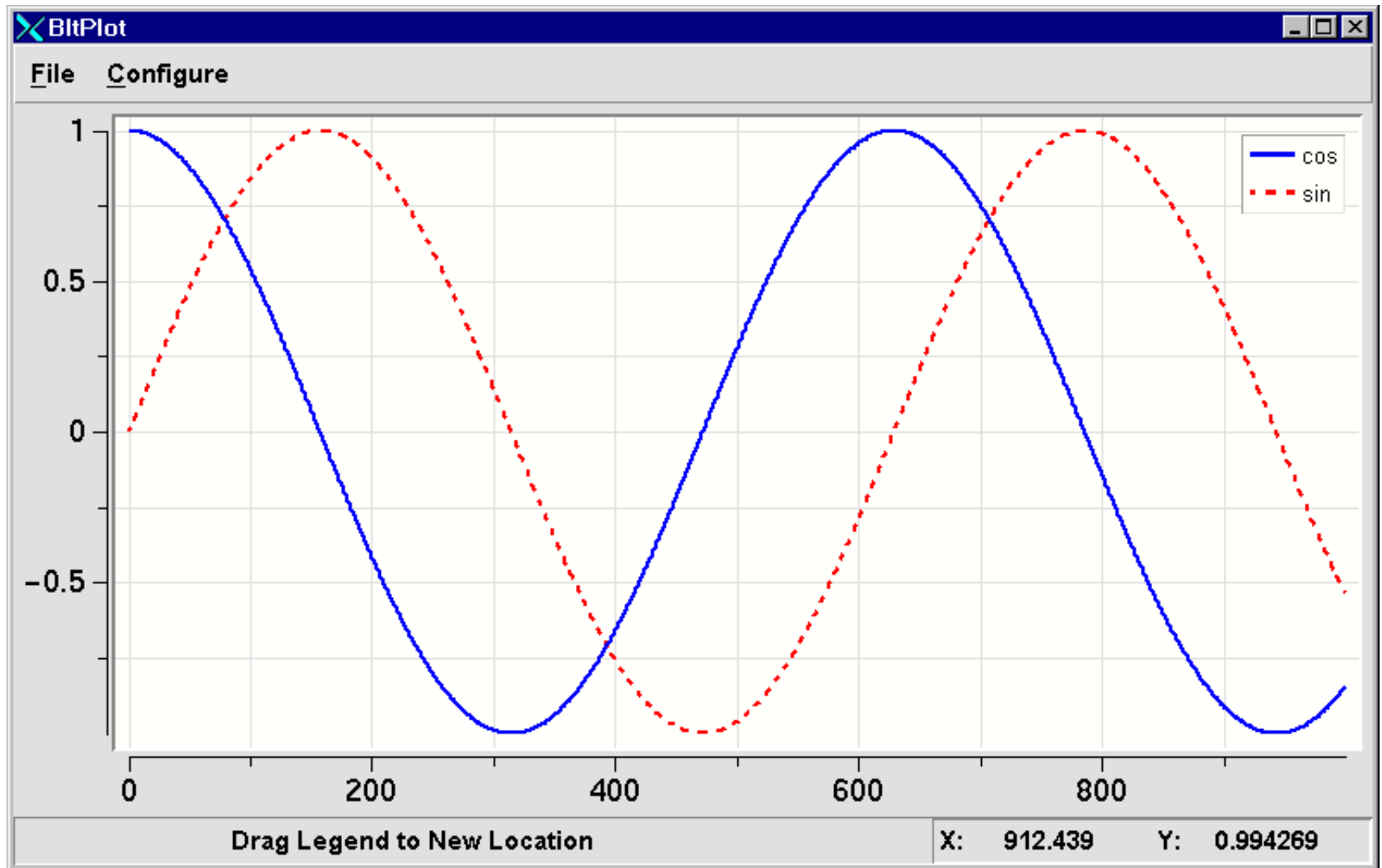
BltPlot: Enhancements to the Pmw.Blt.Graph widget

- **GUI routines to configure all of the plot characteristics, such as axes, markers, legends, etc.**
  - These routines work with any Pmw.Blt.Graph instance so they can be used from the standalone plotting widget in this package (BltPlot.BltPlot) or from any application that uses the Pmw.Blt.Graph widget

- **Routines to save and restore plot settings and data.**

- **Used in the mcaDisplay described later.**

BltPlot Dialogs

Pioneering Science and Technology

Office of Science U.S. Department of Energy

epicsPV: Subclasses Geoff Savage's CaChannel class

- If a PV name is given then the class constructor will do a searchw() by default.

- setMonitor() sets a generic callback routine for value change events. Subsequent getw(), getValue() or array_get() calls will return the value from the most recent callback, and hence do not result in any network activity or latency. This can greatly improve performance.

- checkMonitor() returns a flag to indicate if a callback has occurred since the last call to checkMonitor(), getw(), getValue() or array_get(). It can be used to increase efficiency in polling applications.

Pioneering Science and Technology

**Office of Science
U.S. Department
of Energy**

epicsPV

- getControl() reads the "control" and other information from an EPICS PV without having to use callbacks. In addition to the PV value, this will return the graphic, control and alarm limits, etc.

- putWait() calls array_put_callback() and waits for the callback to occur before it returns. This allows programs to use array_put_callback() synchronously and without user-written callbacks.

epicsMotor

- **Class library for EPICS motor record**
- **Methods:**
  - move(), stop(), wait(), get_position(), set_position()
- **Virtual attributes:**
  - slew_speed, base_speed, high_limit, low_limit, done_moving, backlash, resolution, etc.
- **Example use:**

```
from epicsMotor import *
m = epicsMotor('13LAB:m5')
m.move(10.)
m.wait()
m.get_position(dial=1, readback=1)
9.9609375
```
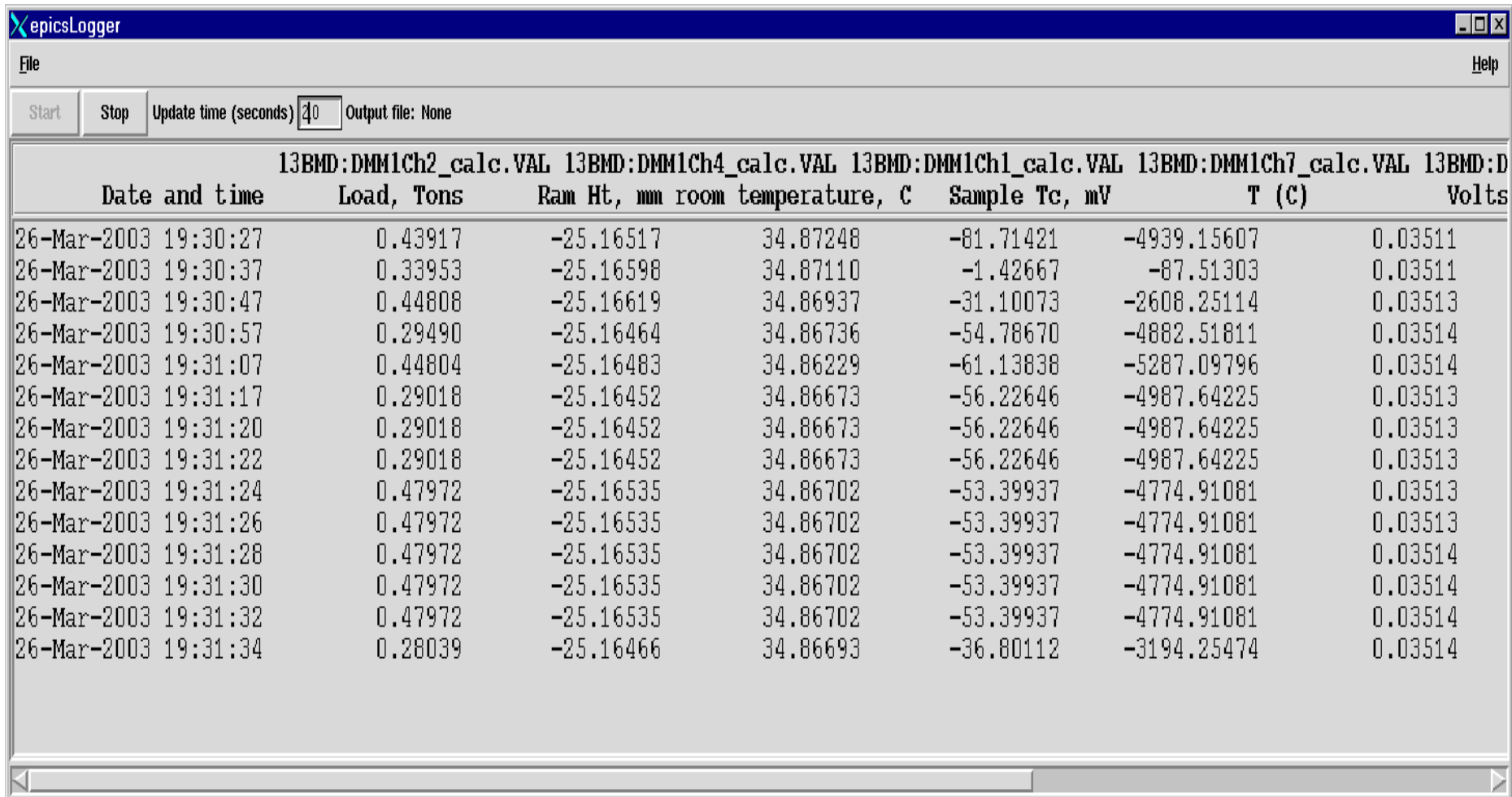
epicsScaler

- **Class library for EPICS scaler record**
- **Methods:**
  - start(), stop(), read(), wait(), get_label(), set_label()
- **Example use:**

```
>>> from epicsScaler import *
>>> s = epicsScaler('13LAB:scaler1')
>>> s.get_counts()
>>> s.read()
[0, 0, 0, 0, 0, 0, 0, 0]
>>> s.start(1.)
>>> s.wait()
>>> s.read()
[10000000, 0, 0, 0, 0, 0, 0, 0]
```

epicsLogger

GUI appplication for logging EPICS PVs to the screen and to a disk file

mcaDisplay

Office of Science
U.S. Department
of Energy

Pioneering
Science and
Technology

mcaDisplay()

- **Replacement for my IDL MCA display program**

- **Much nicer in many respects, since the Blt plot widget has many more east-to-use features than IDL's direct graphics**

- **Python object with callable methods, so it can be remotely controlled**

- **Device independent. It reads files and controls the "hardware_mca" class. "hardware_mca" can be subclassed for any hardware. Presently the EPICS MCA record is supported**

Mca: Device-independent MCA class

- **Support classes: mcaROI, mcaCalibration, mcaElapsed, mcaPresets, mcaPeak, etc.**

- **Many methods: add_roi(), fit_background(), fit_peaks(), get_calibration(), set_calibration(), write_file(), read_file(), etc.**

- **Used as base class of epicsMca.**

epicsMca:
Subclass of hardwareMca, which is subclass of Mca

- **All methods of Mca, plus start(), stop(), erase(), wait(), etc.**
- **Re-implements base class routines for set_calibration(), set_rois(), etc. to communicate with fields in the EPICS MCA record**
- **Example use:**

```
from epicsMca import *
mca = epicsMca('13BMD:aim_adc1')
mca.erase()
mca.start()
mca.wait()
mca.write_file('test.001')
```

mcaPeakFit

mcaPeakFit Parameters
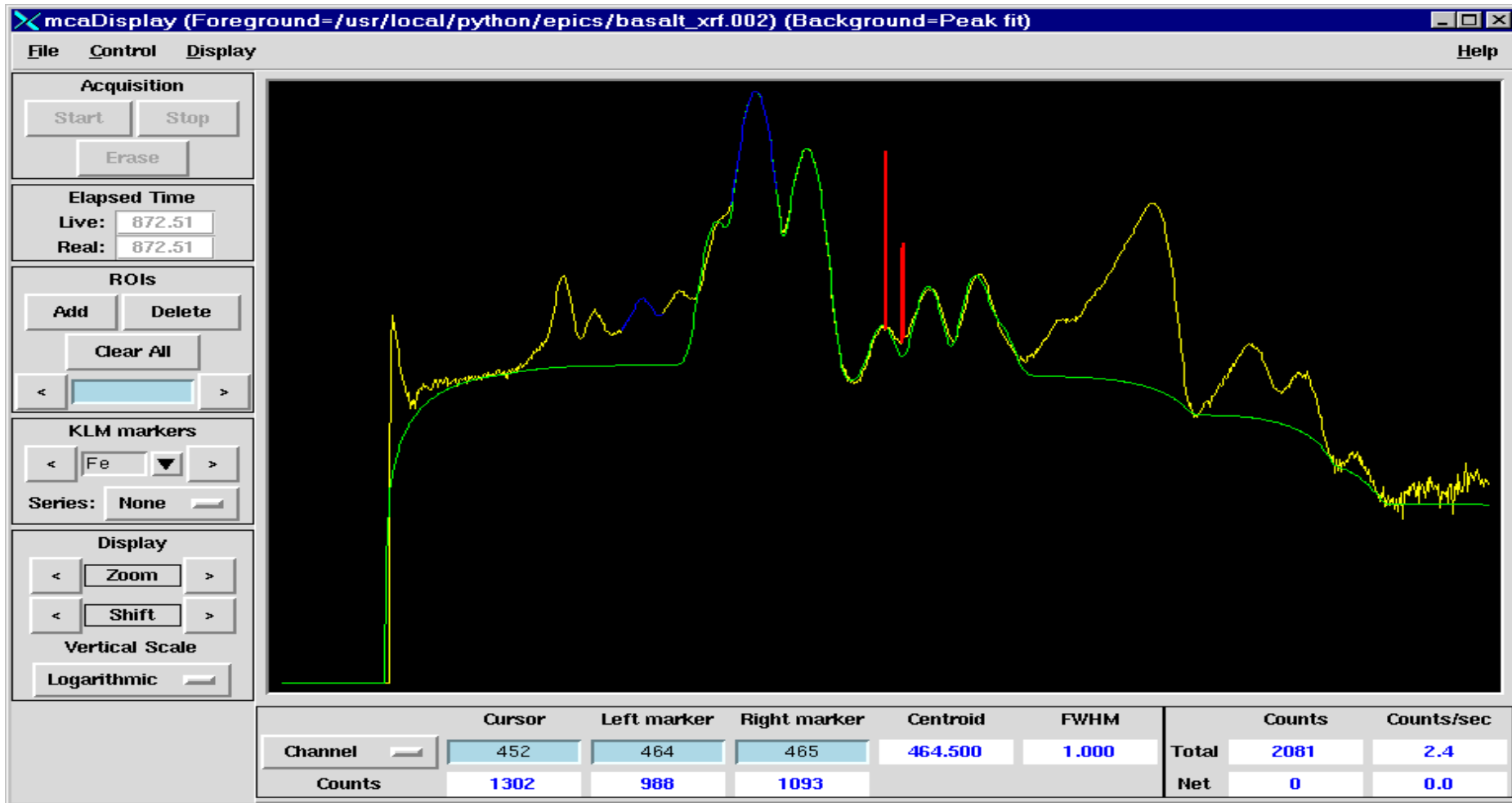
Pioneering Science and Technology

**Office of Science
U.S. Department
of Energy**

mcaPeakFit Output

```
*******************************************************
     Fit of /usr/local/python/epics/basalt_xrf.002:3

Real time (seconds):                  872.51
Live time (seconds):                  872.51
Initial FWHM offset, slope:        0.150000   0.000000
Optimized FWHM offset, slope:      0.305848   0.002562
Initial energy offset, slope:      0.047064   0.017652
Optimized energy offset, slope:   -0.046042   0.017947
# Iterations, function evals:             8          93
Chi squared:                      5.92047e+09
Status code:                           2
Time to fit:                       3.246

        Peak       Energy    FWHM      Area      Background   Area/MDL   Area/Bkg

        Mn Ka      5.895    0.3121    204198.9    19513.0      487.3      10.5
        Fe Ka      6.400    0.3123   3079580.9    19196.0     7409.1     160.4
        Fe Kb      7.059    0.3127    949900.2    17498.0     2393.7      54.3
        Ni Ka      7.472    0.3129         0.0    15107.0        0.0       0.0
        Cu Ka      8.041    0.3131     17554.6    15052.0       47.7       1.2
        Zn Ka      8.631    0.3134     49345.0    15625.0      131.6       3.2
        Zn Kb      9.572    0.3138     14772.8    16192.0       38.7       0.9
        Ga Ka      9.243    0.3136     63076.2    15686.0      167.9       4.0
```

OK

Mpfit

- **Generalized non-linear least squares data fitting**
- **Based on LMFIT from Minpack**
- **Originally translated to IDL by Craig Markwardt, I translated to Python**
- **Much faster and more accurate than the version provided in the Scientific Python package in Scientific.Functions.LeastSquares.**
- **Contraints, fixed parameters, analytic or numerical derivatives, etc.**
- **Used in mcaPeakFit**

Med: Device-independent multi-element detector class

- **Collection of Mca objects. Methods operate on all contained Mca objects. Example:**
  - add_roi(), set_presets(), get_calibration(), etc.
- **Used as base class of epicsMed.**

epicsMed

- **Subclass of Mca and Med**
- **All methods of Mca and Med, plus start(), stop(), erase(), wait(), etc.**
- **Re-implements base class routines for set_calibration(), set_rois(), etc. to communicate with fields in the EPICS MCA record**
- **Example use:**

```
from epicsMed import *
med = epicsMed('13GE1:med:', 16)
med.erase()
med.start()
med.wait()
med.write_file('test.001')
```

medDisplay

Pioneering Science and Technology

Office of Science U.S. Department of Energy