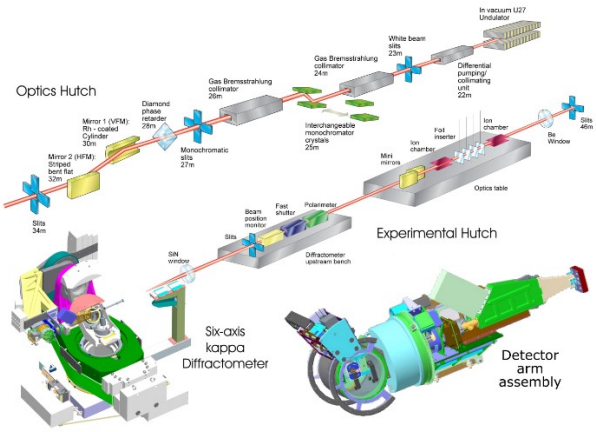
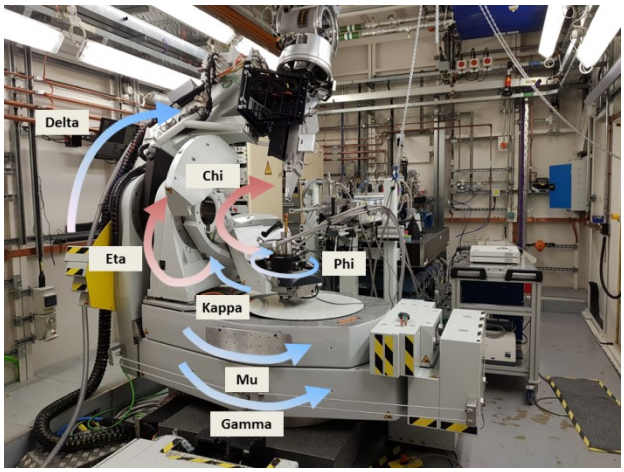
I16 User Guide

1. Introduction

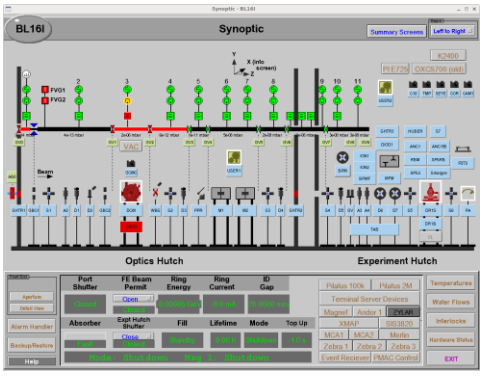




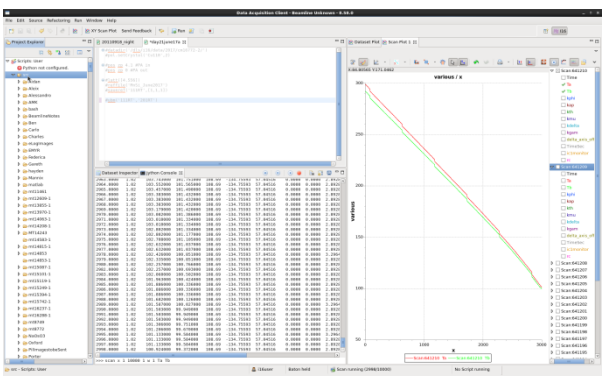
I16 is a general purpose hard x-ray beamline at Diamond Light Source, Rutherford Appleton Laboratory, UK. The main components of the beamline are a single crystal Si monochromator, beamline optics (including phase plates for polarisation control), a 6-circle kappa diffractometer, a detector arm with a photon counting area detector and rotation stage for polarisation analysis. The diffractometer can support a number of sample environments such as cryostats, magnets and furnaces.

2. Controlling your experiment

Beamline motors, detectors and equipment are generally controlled through the program GDA on the beamline computers. Specific control of devices can be performed through EPICS, however this is not recommended for users. The computers use the linux operating system RedHat7, which is specially configured for use at Diamond Light Source. GDA can be run on any linux workstation in the I16 control cabin, however we normally use workstation i16-ws001 (the two monitors side-by-side, next to window looking into the experimental hutch). GDA uses a command line and scripting environment using a Python based syntax. Data is plotted directly in GDA and all data is stored remotely, allowing you to access it from any other workstation.



EPICS – Motor control



GDA – Data Acquisition

3. Where Data is stored

Experimental data, including scan files, detector images and the log file are all stored on a secure network server.

The file directory of the data will depend on your experiment number (mm#####-1):

```
/dls/data/i16/2022/mm12345-1
```

From GDA you can see your data directory using the command:

```
datadir()
```

You can access the data from any other workstation and you can back up your data using the I16 Data Dispenser. Data can be accessed remotely after your experiment for several months, after this the data is stored on the archive and can be accessed via the Diamond archive manager.

4. Basic Commands

In GDA all available devices are controlled by "pseudo devices" that control motors, detectors etc. These devices can read back their position, move to a position or be scanned over a range. Here is an example, moving the diffractometer:

```
pos shutter 1 # opens the x-ray shutter
pos eta 20 # Moves eta to 20
pos eta 10 delta 10 # Moves eta and delta simultaneously
inc eta 1 # Increments eta by +1 degree
pos pil3 1 # takes an exposure with the Pilatus detector
pos eta # display the current position
```

5. Scan Commands

There are two main types of scans of devices, centred and absolute. Both move to a position and perform an action at each point. Data from scans is stored in the experiment directory.

```
scan device start stop stepsize read-back-device(s)
scancn device stepsize steps read-back-device(s)
```

For example:

```
scan eta -1 1 0.02 w 0.1 diode
scancn eta 0.02 51 w 0.1 diode
```

HKL Scans

Once an orientation matrix is available, reciprocal space directions can be scanned:

```
scan hkl [0 3 0] [0 4 0] [0 0.1 0] pil3 1 roi2
scancn hkl [0 0.1 0] 21 pil3 1
scancn l 0.01 101 t 1
```

Energy Scans

Energy dependent scans need a little more care. For example, maintaining a fixed Q over an energy scan requires that the theta 2-theta positions be recalculated of each energy. Also if the phase plate or PA (polarization analyser) are being used their energy dependence must be compensated for in the scan. E.g.:

```
scan energy 7.0 7.3 0.1 hkl [0 3 0] pol 90 t 1
```

In the above energy scan, hkl is maintained at [0 3 0] and the PA is adjusted to maintain the 90 deg polarization channel.

2D Mapping Scans

Scans be be made of two scannables in a map:

```
scan sx -0.5 0.5 0.05 sy -0.5 0.5 0.05 pil 1 roi1 roi2
```

This scans repeats an sy scan at every sx position.

Adding metadata

Add temporary devices to the data file headers (metadata): 'addmeta'. Command takes either a single PD/scannable or a list of PD's.

```
scancn eta  0.1 31 pil 1 roi1 roi2 Ta Tb
```

The above is a centred scan of eta with the pilatus100k detector, "pil 1" means expose the detector for 1 second. "roi1" and "roi2" are regions of interest - the sum and maxval of each region will be stored at every point making plotting easier. "Ta" and "Tb" are temperature values, stored at every point of the scan.

Checkbeam

Adding the checkbeam command will check that there is beam before taking an exposure. Useful for scripting and overnight scans.

```
scancn eta 0.1 31 checkbeam t 10
```

6. Detectors

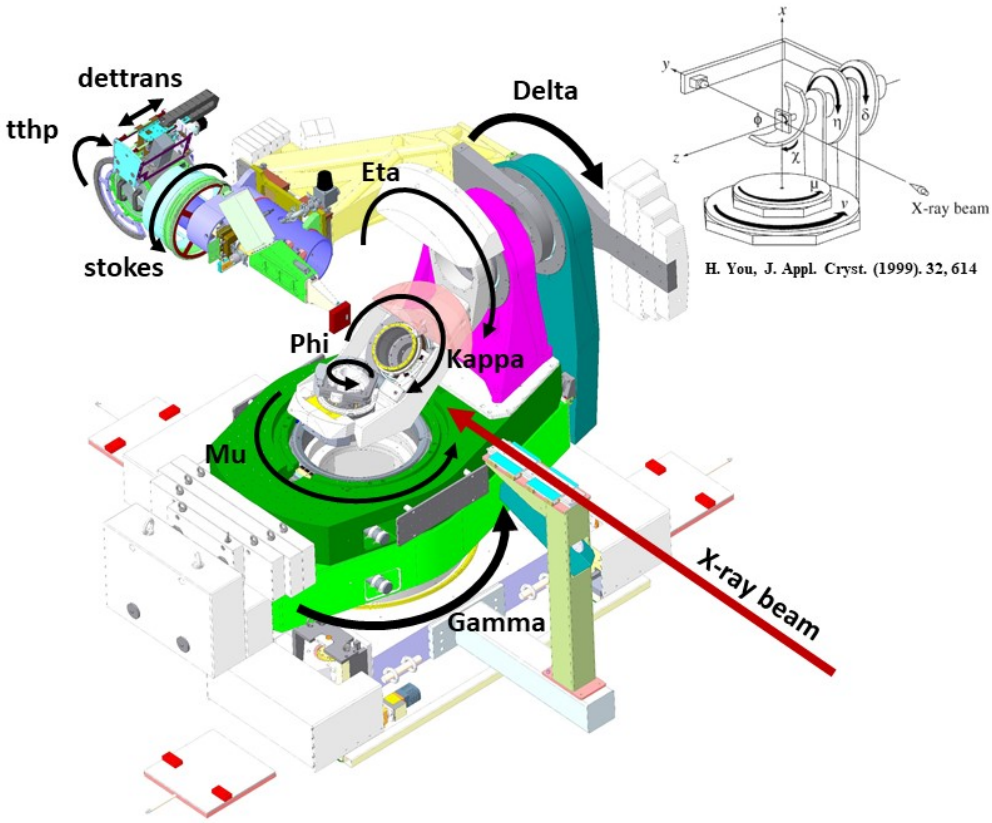
Different detectors are located around the detector arm, requiring an offset in delta. Moving the detector offset does not move the detector arm, so you much return to your required delta.

```
pos do do.pil # detector offset for Pilatus
pos do 0 # detector offset for analyser/ diode etc.
diodein() # selects diode on rotation stage
apdin() # selects APD on rotation stage
pilin()/pilout() # Changes delta offset and moves delta
```

Name	Commands	Notes
Diode	pos diode	Current amplifier, can be used in direct beam
APD	pos t 1	Avalanche photo diode, saturates at 1e8 ph/s
Pilatus	pos pil 1	Pilatus Area detector, saturates at 1e6 ph/s
Merlin	pos merlin 1	Merlin detector for analyser crystal, saturates 1e5 ph/s
BPM	pos bpm 0.001	Beam position monitor, puts scintillator in beam

7. Introduction to the Kappa Diffractometer

The I16 diffractometer is a 6-circle kappa geometry diffractometer, capable of aligning a sample in any orientation relative to the incident beam. The large sample space provides space to fit a wide range of sample environments including cryostats, fields and various detectors. We use a eulerian geometry to drive the sample and detector positions, however these are not always the true motors, for example Chi is a composite of real motors kappa and kphi.



The Eulerian PDs used to drive the diffractometer angles are:

Name	Description	Limits (conservative)
eta	sample rotation || to beam, vertical	0 - 110 Deg & < Delta
phi	sample rotation, kphi is real motor	0 - 360 Deg (watch cryostat pipes!)
chi	composite angle perpendicular to beam	98 - -5 Deg
mu	sample rotation, horizontal	0 - 100 Deg & < gam
delta	detector arm, vertical, depends on pos do	0 - 130 Deg (>0 if using Pilatus or Merlin)
gamma	detector arm, horizontal	0 - 100 Deg (>0 if using Pilatus or Merlin)

Vertical Geometry

In this geometry the scattering plane is vertical in the lab frame. We rotate the detector arm (Delta) vertically as the 2theta angle, eta rotates the sample parallel to the plane (sample theta), chi rotates the sample perpendicular to the scattering plane and phi rotates the sample about the scattering plane.

An example specular reflection is:

```
pos delta c2th([0,0,12]) eta c2th([0,0,12])/2 chi 90 phi 0
```

Horizontal Geometry

In this geometry the scattering plane is horizontal in the lab frame. We rotate the detector axis (gamma) horizontally as the 2theta angle, mu rotates the sample parallel to the plane (sample theta), chi rotates the sample perpendicular to the scattering plane and phi (chi=0) or eta (chi=90) rotates the sample about the scattering plane.

An example specular reflection is:

```
pos gamma c2th([0,0,12]) mu c2th([0,0,12])/2
```

8. Aligning Samples

Aligning the crystal to the diffractometer requires the knowledge of two crystallographic orientations – either explicitly defined relative to the coordinate system or based on two reflections. Here is a standard set of commands for aligning a sample with roughly known faces:

```
newub # interactive... sample name and lattice parameters

# Align sample optically in COR camera with eta 20, chi 90. Choose phi such
that a known face is pointing towards the camera and the sample surface in
pointing vertically on the camera screen.
# This function creates two sample orientations and creates a dummy UB.
'optical normal' and 'optical parallel'
opticalalign([0,0,1], [1,1,0]) # surface normal, direction facing camera

# Add contraints
con gam 0 mu 0 phi phi() # phi fixed
con gam 0 mu 0 bisect # bisecting

# Move to position (with pilatus)
pos do do.pil
simhkl [h,k,l] # Simulate position
pos hkl [h,k,l]

# Centre detector on reflection
scancn eta 0.05 61 pil 1
go maxval
pil2max

# Add reflection to reflist
addref [h,k,l] '300K specular'

# Regenerate orientation matrix with 1st ref
calcub 1 'optical parallel'

# Find second reflection
simhkl [h2,k2,l2]
pos hkl [h2,k2,l2]
scancn eta 0.05 61 pil 1
go maxval
pil2max
#---OR---
simallsr2 [h2,k2,l2] # pick a set of azimuths
scan sr2 [h2 k2 l2 -90] [h2 k2 l2 90] [0 0 0 1] pil3 1 roi2

# Add reflection to reflist
addref [h2,k2,l2] '300K off-specular'

# Regenerate orientation matrix with 1st ref
calcub 1 2

# Refine lattice parameters
hklcur=hkl(); lp=latt()
latt(lp[0]*h2/hklcur[0], lp[1]*k2/hklcur[1], lp[2], lp[3], lp[4], lp[5])
```

9. Introduction to Scripting

Scripts can be created in GDA to run measurements overnight. They are written using the same commands as in the GDA Jython terminal, but also allow definition of variables, loops and functions.

Creating a Script

Scripts are stored in the directory /dls_sw/i16/scripts/YYYY/mt####-# where YYYY is the year and #### is your experiment number. You may need to create a folder in the scripts directory, to do this in GDA:

```
newexperiment('mm12345-1') # create scripts folder and templates
```

In GDA, create a new python file:

```
newscript('2022_05_01_night') # create new script
newscript('2022_05_01_temp', 'temperature') # use template
```

Available templates for scripts are: 'temperature', 'surface', 'psi', 'chi'.

Running a Script

In GDA, put the script in the editor and click the button "Run"

Or, from the terminal:

```
script_runner('2022_05_01_night', 'email@address.com')
```

Starts the script in a try/except loop and sends an email if the script stops or finishes.

The script_runner function also creates a new script, e.g. '2022_05_01_night_next.py' that will run after the first script completes. By default this script will delete itself when run.

Stopping a Script

In GDA, click the "Stop" button. This will end the current scan and script, but may not end the current movement - it is a "gentle stop".

Example Script

```
# This is a comment

# When defining new variables, check they are not already in the terminal
namespace - you'll have serious problems otherwise!
beat_eta_pos = 55.4321

# Loops are defined by indentation:
for loopval in frange(-1,1,0.1):
    print loopval
    pos eta loopval
    for innerloop in frange(-0.5,0.5,0.1):
        print loopval, innerloop
        pos delta innerloop
        scancn chi 0.05 31 checkbeam pil 1

# Functions can be useful to keep code tidy
def quick_align(hval,kval,lval):
    pos hkl [hval,kval,lval]
    scancn eta 0.01 61 checkbeam pil 1 roi2
    go maxval

# run another script
run('/dls_sw/i16/scripts/2018/mt1234-1/followup_script.py')
```

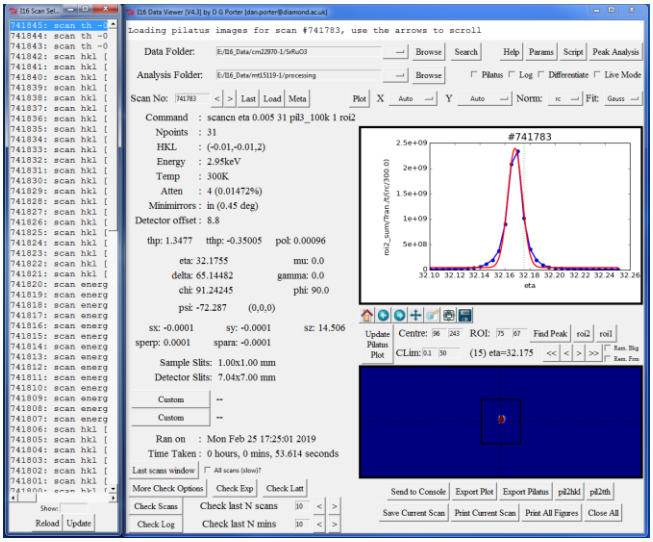
Example Temperature Script

```
# Temperature Dependence Script
# First scan #54321
# 25-12-2052

# Define the measurements
con gam 0 mu 0 phi 0 # phi-fixed mode
hkl_vals = [[0,0,1], [0,0,2], [1,0,3], [0,1,3]]

# setup the beamline
pos shutter 1
pos x1 1
pos energy 8
pos ss [0.5,0.5]
pos ds [0.5,0.5]

# set temp_stop=True at any time to end the script after the current
temperature
temp_stop = False
for tempval in frange(5,300,5):
    if temp_stop is True: break

    # Change + wait for the temperature
    pos tset tempval
    numgoes=0
    while abs(Ta()-tempval) > 0.5 and numgoes<10:
        numgoes+=1
        print 'Waiting...Ta=',Ta()
        w(30)
    w(60) # stabilising time
    pos szc 0 # change the sample height

    for n in range(len(hkl_vals)):
        pos hkl hkl_vals[n]
        scancn eta 0.05 61 checkbeam pil3 1 roi1 roi2
        go peak
        pil2max(min_intensity=1000) # centre peak on detector
        pil2max(min_intensity=1000)
        hkl_vals[n] = hkl() # update peak position to follow reflection

print 'Finished Script!'
```

Beamline Specific Commands

```
i16_gda_functions functions available in GDA
These functions are only available on I16 and not part of the standard
syntax.

i16_gda_functions - module of various useful functions
 help_i16() - prints this list

Diffcalc:
 latt(a,b,c) - highly versatile lattice setting function
 last_ref_index() - returns the number of stored reflections
 currentub() - displays currently used reflections in ub calculation
 opticalalign([0,0,1],[1,0,0]) - create orientation matrix based on camera
 simhkl([h,k,l]) - returns simulated positions including kphi
 simallsr2([h,k,l]) - prints all positions of sr2 scan
 simallpsi([h,k,l]) - prints all positions of psi scan
 simallhkl(hmax, kmax, lmax) - prints all available reflection positions
 maxhkl(max_twotheta, energy_kev) - returns [hmax, kmax, lmax] at energy
 movetoref(refno) - move to exact location of reflection in reflist
 get_cif_hkl("file.cif") - Returns list of allowed hkl values available

Experiment Management:
 newexperiment('mm12345-1') - change datadir and create scripts/folders
 newscript('temp_dep') - create new script in scripts/exp directory
 script_runner('script_name.py', 'your.email@diamond.ac.uk') - run script
 script_timer('script_name.py') - returns predicted time for script
 currentscan() - returns current scan number
 nextscan() - returns the next scan number
 scanfiles(0) - returns filename of scan number
 nexusfiles(0) - return nexus filename of scan number
 benchmark(atten_val, notes) - run standard diode scan and bpm scan in
direct beam

Other:
 pil2max - moves delta, chi to move pilatus max to centre
 base2bpm - moves base_y, base_z to move bpm max to centre
 merlin2max - moves delta, chi to move merlin max to centre
 mirrormove - function to move mirrors in a relative way
 edgefit() - edge function with fitting
 edgesimple() - simple edge function, returns edge of last scan
 simbl()  - Send current diffractometer positions to the Blender simulator
 help_processors() - gives help on available auto-processors
```

10. Data Analysis



Data analysis can be performed quickly using the "I16 Data Viewer". Start the program from the desktop icon. You can browse scan files and metadata, create high quality plots, fit and integrate peaks, redefine regions of interest on area detectors and perform automated analysis of scripts.

Once the application has started, begin by selecting your experiment folder ("Data Folder") and the location to place analysis files ("Analysis Folder").

The program runs on Python 2+/3+ and works on all operating systems. It is available for download here: https://github.com/DanPorter/Py16

11. Useful Contacts

Internal Number	Name
8787	EHC / out-of-hours
8899	Control Room
8571	User Office
8087	Steve Collins
8226	Alessandro Bombardi
8786	Gareth Nisbet
8345	Dan Porter
8062	Colin Brodie

12. Useful Information

https://www.diamond.ac.uk/Users/Experiment-at-Diamond/Your-time-at-Diamond.html

Restaurant Opening Times

Monday – Friday	Weekend
Breakfast 07:30 - 09:30	Breakfast 07:30 – 09:00
Lunch 11:45 - 13:45	Lunch 12:00 - 13:30
Dinner 18:15 - 20:15	Dinner 18:15 - 20:15

13. Remote Operations

NX Server (NoMachine)

Any user can connect to the Diamond network by creating a client on the NX server using NoMachine using your FedID and password. Installing NoMachine is required, then a connection to nx-user.diamond.ac.uk, Port: 4000. For more details, see here.

Once a client has been created, you have a personal desktop where you can personalise your environment and have access to Diamond's network and software. Useful I16 directories are listed below:

/dls/i16/data/2022/mm####-1	Data directory (mm####-1 = your experiment ID)	Read only
/dls_sw/i16/scripts/2022/mm####-1	Scripts directory	Read/ write
/dls_sw/i16/software/python/userscripts/i16user	Software directory	Read/ write
/dlw_sw/i16/software/python/Py16	I16 Dataviewer directory	Read only

SynchWeb (ISPyB)

ISPyB is an online database that captures information on every scan performed so it is easy to monitor the progress of scripts etc. The service is available at the following address:

https://ispyb.diamond.ac.uk

You are required to sign in with your FedID and password.

Currently the information visible is limited but this will improve over time.

JupyterHub

JupyterHub is a cloud based computing system hosted on site allowing you to create and run Jupyter Notebooks for data analysis:

https://jupyterhub.diamond.ac.uk/

You are required to sign in with your FedID and password.

A Jupyter Notebook will be created for you in the scripts directory:

/dls_sw/i16/scripts/YYYY/mm#####-#/mm#####-#.ipynb

By default, jupyterhub only has access to your personal user space and not to the scripts directory. You should therefore create a link to this directory in your Home directory using a terminal in NX or from a workstation at Diamond:

$ ln -s /dls_sw/i16/scripts/YYYY/mm#####-# ~/i16_scripts_mm#####-#