NeXus Code Camp 2012

Mark Könnecke

Paul Scherrer Institute
Switzerland

September 19, 2012

General Topics

- Finalize CIF coordinate Issue
- Change of documentation form docbook to Sphinx
- Cleanup trac tickets
- Develop a materials definition
- Do we need specials for timed data?
- DECTRIS (again)
- Review OO-NeXus status?

Technical Topics

- PyTree API Tests
- Automatisation and documentation of the NeXus release process
    - Continuous integration
    - Write more or proper unit tests
- CMake versus autoconf
- NXdict replacement design

CIF Coordinate Issue

- Decided: extend NeXus to allow full mapping from CBF to NeXus
- Information to encode:

type	rotation or translation: DONE! transformation_type attribute
direction	vector around which to rotate or along which to translate: DONE! attribute
value	The angle of rotation or the length of translation, DONE!
dependency	The order of operations to place a component, to be discussed!

Expressing Axis Dependency in NeXus

- Implied: use existing NeXus coordinate system
- dependson attribute pointing to depending axis
- transform field in base classes which becomes a comma separated list of the path to the transformations required to position this component
- Create a special container to hold axis dependencies, NXdependency, to collect the dependencies in one place for easy access. This is what CIF does

Dependons Option

```
sample,NXsample
        rotation_angle
        chi (dependson rotation_angle)
        phi (dependson phi)
```

Transform Option

```
sample,NXsample
        rotation_angle
        chi
        phi
        transform = rotation_angle,chi,phi
```

Separate Group Option

```
sample,NXsample
        rotation_angle
        chi
        phi
dependency,NXdependency
        sample/chi =
                sample/rotation_angle
        sample/phi =
                sample/chi
        instrument/detector/x_translation =
                instrument/detector/distance
        instrument/detector/distance =
                instrument/detector/polar_angle
```

Change Documentation from Docbook to Sphinx

- Current documentation in Docbook
- Only experts can write docbook
- Sphinx is restructed Text which is easy to write
- RST converts into many formats including html and pdf
- Issues:
    - Do we like the look of Sphinx?
    - How can we convert automatically?
    - Integration with CMake

NXdict replacement Design

- Current NXDict
  - File format to describe items in a NeXus file
  - API to generate structure and read data from NeXus file
  - Found little (or no) use outside of PSI

NXDict file format

```
##NXDICT-1.0
etitle=/entry,NXentry/SDS title -type NX_CHAR -rank 1
instrument=/entry,NXentry/SDS instrument -type NX_CHAR -ran
estart=/entry,NXentry/SDS start_time -type DFNT_CHAR -rank
eend=/entry,NXentry/SDS end_time -type DFNT_CHAR -rank 1
edef=/entry,NXentry/SDS definition -type DFNT_CHAR -rank 1

table=table2
var=sdw
units=mm
tablevar=/entry,NXentry/$(table),NXCollection/SDS $(var) -ra
tabledet=/entry,NXentry/$(table),NXCollection/SDS detector -
tablequip=/entry,NXentry/$(table),NXCollection/SDS equipmen
tabletext=/entry,NXentry/$(table),NXCollection/SDS $(var) -
```

NXDict dictionary Maintenance API

```
NXstatus NXDinitfromfile(char *filename, NXdict * pDict);
NXstatus NXDclose(NXdict handle, char *filename);

NXstatus NXDadd(NXdict handle, char *alias, char *DefString
NXstatus NXDget(NXdict handle, char *alias, char *pBuffer,
NXstatus NXDupdate(NXdict handle, char *alias, char *pNewVa
NXstatus NXDtextreplace(NXdict handle, char *pDefString, cha
                        int iBuflen);
```

NXDict dictionary Data Transfer API

```
NXstatus NXDputalias(NXhandle file, NXdict dict, char *alia
NXstatus NXDputdef(NXhandle file, NXdict dict, char *pDefSt
                   void *pData);

NXstatus NXDgetalias(NXhandle file, NXdict dict, char *alia
NXstatus NXDgetdef(NXhandle file, NXdict dict, char *pDefSt
                   void *pData);
NXstatus NXDdefget(NXdict handle, char *pKey, char *pBuffer

NXstatus NXDaliaslink(NXhandle file, NXdict dict,
                      char *pAlias1, char *pAlias2);
NXstatus NXDdeflink(NXhandle file, NXdict dict, char *pDef1

NXstatus NXDopenalias(NXhandle file, NXdict dict, char *ali
NXstatus NXDopendef(NXhandle file, NXdict dict, char *pDefSt
```

Future NXdict

- Base on NXDL?
- Competition to CDF?
- Is there still a need?

Timed Data

- Event mode data
- On the fly scans at synchrotrons
- Groups of parameters being collected on possibly different sampling intervalls
- Group of NXlogs? This then is the data!
- Or scan like: each parameter can become a NXlog in its place in the hierarchy, links in NXdata?
- Other ideas?
- Or no problem at all?
- I want a clear statement how this is done in NeXus!

Materials Definition

- How to describe complex materials: samples, sensors, multi layers etc?
- Chemical formula: steal CIF conventions?
- Some research required

NeXus Release Process

- Problem: only Freddie knows how to make a NeXus release
- Solution 1: document and have privileges
- Solution 2: automatise (but do not get lost in tooling....)

Prioritise!

Transformation Matrices

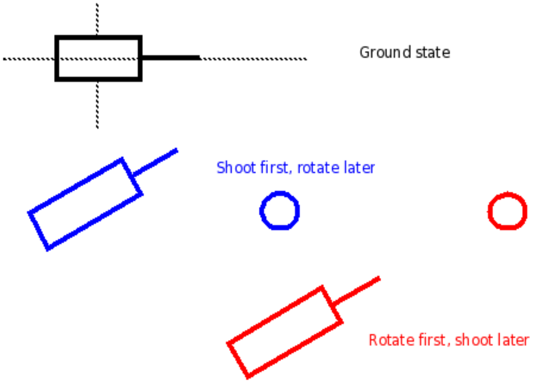
$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation Matrices

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} r11 & r12 & r13 & 0 \\ r21 & r22 & r23 & 0 \\ r31 & r32 & r33 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Combining Transformations



Some Properties

- Transformations can be combined by matrix multiplications
- Individual matrices can be derived by looking at the situation when everything else is 0
- Absolute positions can be obtained by multiplying the resulting matrix with its transpose
- Defines new coordinate systems at components
- CIF contains a duplication: vector, offset scheme

What Use Is This?

- Allows to calculate absolute positions of components in the laboratory coordinate systems
- Can directly convert from a detector coordinate system to vectors in Lab coordinate system
- Calculate things like impact of primary beam on detector, SAS
- Allows arbitray axis to be expressed
- Intuitively describe an instrument with angles and translations and still be able to recover absolute coordinates

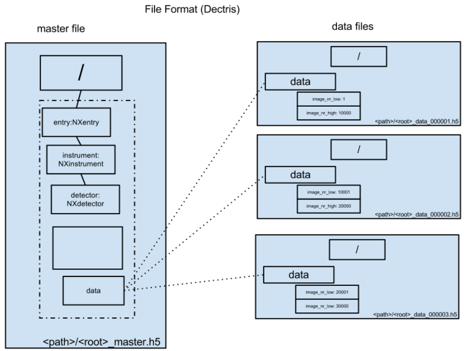
NeXus Axis Mapped

- rotation_angle, polar_angle, rotate 0 1 0
- azimuthal_angle, rotate 0 0 1
- distance, translate 0 0 1
- chi, rotate 0 0 1
- phi rotate, 0 1 0
- NeXus polar coordinate system: rotate azimuthal_angle, rotate polar_angle, translate by distance

CIF Dependency Table

axis-id	type	equipment	dependson	vector	offset
gonio_phi	rotation	goniometer	.	1,0,0,	...
det_z	translation	detector	.	0,0,-1	0 0 0
det_y	translation	detector	det_z	0,1,0	0,0,0
det_x	translation	detector	det_y	1,0,0	0,0,0

DECTRIS Again



- DECTRIS has a problem:
  - Detector outputs 5-10 GB/sec
  - The deliver the detector and the computer going with it
  - They cannot ask their customers to provide the appropriate hardware for such a detector: parallel file system etc.
  - Must compress and write the file on one computer
  - Compression has to be parallel as CPU intensive
- File structure a workaround for HDF-5 not allowing sections of datsets in different files
- Sidenote: LZ4 or snappy compression; up to 450MB/sec on write

Upcoming DECTRIS Meeting with Community

- DECTRIS aims at meeting customers in october
- How far can we compromise?
- Anyone from the NeXus community who wishes to join?