ENNI/94/P7

Proposal for a European Neutron Scattering Data Exchange Format

Dr. Mark Könnecke
Rutherford Appleton Laboratory
ISIS-Facility
Building R3, Room 1.38
Chilton, Didcot
Oxon OX11 0QX
United Kingdom

Abstract

In this paper several scientific dataformats: HDF, netCDF, CDF, FITS and STEP/Express will be reviewed in order to determine their usability as a common european dataformat for neutron scattering data. netCDF is propsed as a physical file format for neutron scattering. A list of possible keywords in a common european dataformat file is given. This should be understood as a draft.

Version 4.2: 17.6.1994
ENNI/94/P7

Contents

Chapter 1

ENDCDF

At a European workshop on data visualisation at Cosener's House, Abingdon on the 26.11.1993 it was agreed that a decision on a common visualisation package for neutron scattering is not feasible for several reasons.

- Nobody can forecast the development of the visualisation software market. Companies may disappear in the current recession. Several public domain visualisation packages are currently under development.
- A visualisation package is a major investment in software and training. If a package is already present at a site nobody will be willing to change to another package just for evaluating neutron scattering experiments.

However, it would be a great help if all the visualisation packages would be able to access rawdata in some common format. So the idea for a European Neutron Scattering Dataformat (ENDCDF) was born. Additionally a common exchange format would facilitate software reuse through the neutron scattering community. The first step towards such a format is the evaluation of scientific dataformats already available. A report on such dataformats is presented here.

1.1 Requirements for a common neutron diffraction dataformat

In order to have a measure against which the dataformats in question could be compared, some requirements for a ENDCDF will be stated.

- ENDCDF must be network transparent and readable across a wide range of hardware platforms, from the IBM–PC to a Cray.
- it should be readable from a large variety of visualisation packages.
- a library of data access functions callable from FORTRAN and C, C++ should be available in order to speed up application development and to enforce standardization.
- with the advent of area detectors ENDCDF should allow for the efficient storage and retrieval of large amounts of data.
- a built–in means of effective data compression would be appreciated, as some of the neutron diffraction datasets are very large.
- the chosen dataformat should be flexible enough to store data of any kind, such as measurement parameters, data themselves, text data describing the sample or pointing to sources of related information.

1.2 Comparison of scientific dataformats

A survey of currently available scientific dataformats was performed in order to find a format which fulfills the requirements of the neutron scattering community. Scientific dataformats included in this comparison were ASCII, netCDF, CDF, FITS, CIF and STEP/Express. ASCII is the simple ASCII file as used by ILL. The result of the comparison of scientific dataformats is represented in table **??**. From the comparison of scientific dataformats, I propose that we should adopt netCDF. It fulfills most requirements except compression. But compression can be achieved by compressing data separately and storing the result as byte–array. The strongest argument in favour of netCDF is its support for extracting subsets of data and the mathematical tools already available. Moreover it will be included in HDF, which appears to develop in direction of a public domain multimedia standard. A more detailed description of all these formats is given in chapter **??**. NetCDF will be described in the next section.

1.3 Properties of netCDF

NetCDF stands for Network Common Data Format. NetCDF has been developed by Unidata, a software support group of UCAR (University Corporation for Atmospheric Research). NetCDF is a data format entirely devoted to storing scientific data. NetCDF can store multidimensional arrays of various integer

Table 1.1: Comparison of scientific dataformats

name	HW	LIB	VIS	Speed	COM	Flex
ASCII	X					X
netCDF	X	X	X	X		X
HDF	X	X	X	X		X
FITS			few	X		
CIF	X					
STEP	X					X

Legend:

HW	support for all important HW-platforms
LIB	availability of PD–access library
VIS	supported by visualisation packages
COM	built in compression
Flex	fexibility

and floating point types, together with some information to describe the data and their dimensions. New or additional data can be added to netCDF files at any time. NetCDF is self describing. For the actual storage of data netCDF uses the XDR–library for hardware independent binary representation of numerical values. This makes netCDF files portable to different hardware platforms via networks. All important hardware platforms used in the scientific community are supported. A public domain library of access functions both for FORTRAN and C is available. A highlight of this library is the support for reading and writing subsets of otherwise larger multidimensional arrays. A means of data compression was omitted from netCDF in order to allow for the previously mentioned subset access to data. For netCDF a whole variety of software tools is available or will become available soon:

- netCDF supports a tiny data definition language called CDL. Tools are available to translate CDL files into netCDF files or in C or FORTRAN source code templates to write the specified netCDF files.
- a tool for dumping a binary netCDF file into its CDL (ASCII) equivalent is supplied with the library.
- a utility for editing netCDF files and performing mathematical calculations on stored values is available. This tool supports a C alike programming syntax.
- a HDF interface is in early stages of usability.

Furthermore netCDF is supported by a large variety of public domain and commercial data visualisation and manipulation packages. Several organisations have adopted netCDF as their standard data format. You can obtain the netCDF software package and all detailed information you need via ftp from:

host: ftp.unidata.ucar.edu
directory: pub/netcdf

The netCDF software was tested at ISIS on a VMS, a Open–VMS and a Ultrix platform in a larger programming project. No bigger problems were encountered in building the netCDF– libraries and the accompanying tools on VMS and Ultrix. Building the Open–VMS version actualy involved porting the netCDF library. After the exchange of a few e–mail messages with UCAR all problems encountered were solved. No bugs were found in the library (However, I will not state that there are none!). The documentation provided by UCAR is thorough and good.

The interchangebality of netCDF was tested between the platforms used. The test was positive. There is, however, a problem with ftp transfer to and from VMS. This problem is due to Digital's implementation of the ftp server and is no fault of netCDF. This problem can be solved by taking special precautions during transfer and/or with a special command to change filetype, which exists in VMS version 6 upwards.

1.4 General considerations for an exchange format

As netCDF is expandable, we only need to define *minimum* data structures, which *must* be included for a ENDCDF. Anybody may add what he likes without interfering with older data already stored in the file. This minimum should contain:

- the data themselves including dimensions and axis definitions.
- eventually a detector mask when position sensitive detectors are used.
- the experimental information necessary to do computations on the data (wavelength, flight path length etc.).
- sample environment: pressure, temperature, time
- a pointer to a description of the experiment.
- a pointer to a description of the sample.
- a pointer to the description of the instrument.

Recommended additions to such a file might be pointers to associated publications and results of non trivial calculations performed with the data.

Hard disks get cheaper any day. As consequence there is no need to compress data while calculations are done with them. But for archiving or network transport compression is an important issue. Two topics need to be discussed concerning compression:

- Compression at which stage
- type of compression

Concerning the first question we have the choice to compress the data in the file. This would make the communication of special reading-/ writing-routines together with the data necessary. Or we compress the exchange file. I vote for the second, more general, option.

There are two types of data compression: lossless and lossy compression. Lossy compression achieves the higher compression ratios, but you might loose a bit or two during compression and decompression. This is certainly not acceptable in the neutron scattering community. Lossless compression techniques do not have this drawback, but achieve generally lesser compression ratios than lossy compression.

A state of the art lossless compression algorithm is the Lempel–Ziv–Welch algorithm. Fortunately the public domain Gnuzip utility, available for all important HW–platforms, implements this algorithm. So, I recommend the use of this program as a means for compressing exchange files, if necessary.

One important issue in a discussion of an exchange format is the abstraction level on which the data should be transported. Should all raw data, background measurements and instrument settings be stored or should preprocessed data be stored? The first option would imply the distribution of all instrument specific software to users. This contradicts one main objective of a common exchange format: to facilitate software reuse. But clearly software written to evaluate powder diffraction data cannot be used to evaluate inelastic TOF–data. The conclusion is, that ENDCDF should provide an abstraction for each of the major types of instruments used.

So, the actual rawdata (motor positions, uncorrected counts etc.) would be stored on site. Some intrument specific preprocessing would be done on site and the result would be stored in the exchange format, for the user to take home. Anybody is free to store her or his motor positions etc. in such a file but this information is beyond the scope of an exchange format. The preprocessing step should contain no physical interpretation, just the conversion from hardware data to physical data and straightforward corrections such as normalisations against the incoming beam and background subtracion.

In order to enforce standardization it is necessary to define which values should be stored in a ENDEF–file. Furthermore the data stored may be more complete, if there exist guidelines.

Let us summarize the more political guidelines in the discussion of an european neutron scattering data exchange format:

- netCDF as underlying file format.
- compression of the netCDF–file rather than compression of the data stored in the file in order to avoid the need for specific software.
- abstraction of the raw data on instrument level. No motor counts!
- definition of a set of keywords for data to be stored.

1.5 Proposal for a set of neutron diffraction related data to be included in a ENDCDF-file

A problem is the multitude of different neutron scattering experiments which need to be handled. Information on an experiment come from four different sources:

- Administrative information (Who, Where, Why, When).
- Sample information.
- Instrument Hardware.
- Actual Data measured.

In order to get a clearer understanding of the instrument and sample information lets look at the pass of a poor neutron through the most general neutron scattering instrument (Figure **??**).

Coming from the source the neutron passes 0–1 monochromators. Monochromators come in three flavours: Crystals, Choppers and velocity selectors. Next to the monochromator and in principle anywhere in the instrument might be a filter. Common filters are Soller slits and pinholes. After flying the primary flightpath the neutron reaches the sample. Samples come in two flavours, too:

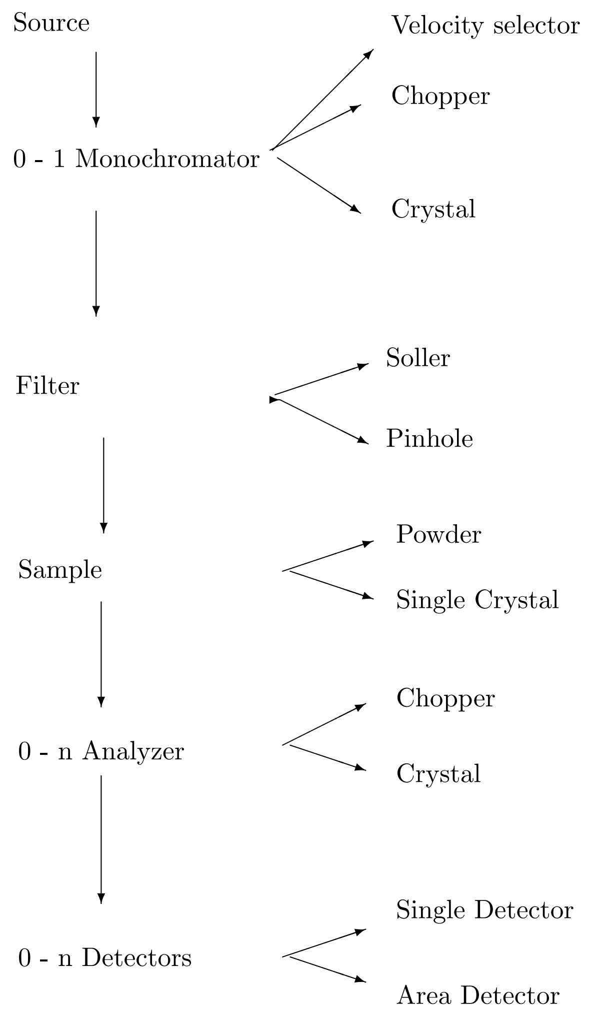

Figure 1.1: The path of a neutron through a scattering instrument

Table 1.2: Instrument parameter to be stored for different instruments

Instrument	Mono	Distances	Analyzer	Detector
4–circle	X			
mono Powder	X			
Triple–axis	X	X	X	
TOF	X	X		X
Laue–pulsed		X		X
inelasic–indirect		X	X	X
TOF–Powder		X		

Single Crystals and Powders. For the purpose of this paper lets consider all liquid, gaseous, glass or metal plate samples as some kind of powder. Furthermore, a single crystal can be oriented differently in respect to the incident neutron beam. According to common usage, this information will be stored together with the counts in the detector. At the sample the neutron is diffracted in one way or another. Next it might hit one of 0–n analyzers. Analyzers are in principle monochromators and come in the same two flavours as those. The neutron eventualy moves on at the secondary flightpath and reaches an detector in order to be counted. Detectors can be very different, too. There are single detectors, area detectors and both of these might have energy resolution, i.e. a time dimension. Furthermore the neutron needs to know where the detector is situated relative to the sample in order to get counted.

Clearly not all these details need to be stored for any type of experiment. Ther should be a section in the file, where the instrument is described in terms of the building units given above. For inelastic and TOF experiments all the distances between components are important in order to calculate the resolution function. Table **??** might help you to determine which data is necessary for your specific experiment. Sample data is not included in that table, because it is common for all experiments to act on a sample.

Another issue is the units, in which data should be stored. As people are very reluctant to adopt standard units, it is not reasonable to prescribe units for each variable. But everybody must state the units, she or he uses, as an attribute of the variable. And there should exist a recommendation for the units to use. Clearly this recommendation can be nothing else than the standard SI–units.

After this more general part, we will now take a closer look first on the administrative bit and than on the neutrons flightpath through the instrument and the parameters it encounters.

1.5.1 General Information

The following general information should be incorporated in any ENDCDF-file:

run_number a name for this file.

Owner Name of the owner of the file, complete with address.

Site the name of the site where the data were measured.

Date Date of the measurement.

ExperimentDescription a reference to some available paper which gives details of the experiment.

AditionalData1-n references to 1 − n aditional datasets such as background, calibration runs etc., which are necessary to evaluate the data.

History[NHistoryItems] What has happened with the data.

1.5.2 Instrument Data

To start with, we need some indication, what will await us:

Instrument name of the instrument on which the data were measured.

InstrumentDescription a reference to a publication, which describes the crucial parameters of the instrument which was used for the experiment.

LocalContact Name and adress of the local contact person responsible for the measurement at the site.

TOF_Flag yes if TOF–instrument, no else.

BuildingBlocks a text value describing the instrument in terms of the building blocks given above. The blocks come in a comma separated list. For example: "Monochromator, Filter1, Distance1, Sample, Distance2, Detector". If more than one item (Filter, Distance) occurs it should get a number appended. Number should start with 1 close to the source. The distances describe the distances between the items they separate. If a distance before the monochromator is given, it will describe the distance between the source and the following item.

Monochromators

Monochromators can either be crystals or choppers. So, the first thing to store is the type of monochromator present.

- Monochromator. This should be a text value with the permissible values of NONE, CRYSTAL, CHOPPER or VELSEL for a velocity selector.

Some people need the wavelength distribution in order to get their caculationd on the experiment done properly. This should be given as:

- NWaveSteps number of steps in the wavlength profile.
- WaveStart start wavelength of the wavelength profile.
- WaveStep stepsize of the wavelength profile.
- WaveProf(NWaveSteps) the intensity for each wavelength.

In some cases you might want to give more detail concerning monochromators or choppers. In the case of a crystal the apropriate values might be:

- MCrystal: text to describe crystal type and reflection used.
- MMosaicSpread: a value describing the spread of the crystal.

In the case of a chopper or velocity selector.

- MChopperType: text to describe chopper.
- MChopperSpeed in revolutions per second.
- MChopperPhase in degrees.

Filters

For all filter you need to give:

- FilterType, can be SOLLER or PINHOLE.

A Soller filter is described by:

- SollerLength in meter.
- SolleHeight in meter.
- SollerWidth in meter.

A pinhole filter is described by:

- PinLength: length of the pinhole.
- StartDiameter: diameter at the start of the pinhole in mm.
- EndDiameter: diameter at the end of the pinhole in mm.

All value names should get a number appended according to a number in the building block description in order to make names unique and distinguishable.

Distances

For all TOF–experiments and some inelastic experiments you need to store the distances. Distances before the sample are in general single. For detectors or analyzers, however, the distances may be arrays, as there might be more than one of these items. So you have two replace dimension in the following by the apropriate dimension for example: NumberOfAnalyzers or NumberOfDetectors.

- Distance1. Distance1 from the building block description.
- Distance2(dimension). Distance2 from the building block description.
- Distance3(dimension). Distance3 from the building block description.
- .....

All distances should be given in meter.

Sample data

For all samples the following information needs to be included:

PhaseName Chemical compound name and polymorph measured.

SampleDescription a more detailed description of the samples synthesis

temperature measurement temperature in Kelvin.

Pressure experiment pressure in bar.

SpecialEnvironment information on special conditions of the experiment, if any.

Cell(6): Cell constants a, b, c, α, β, γ in nanometer and degrees.

Absorption: absorption coefficient.

Samples come in two common flavours: single crystals and powders. For single crystals we need to include:

UB(3,3): Crystal orientation matrix.

nShape: number of parameters describing the crystals shape.

Shape(4,nShape): H, K, L, distance(mm) values to describe the crystals shape. 0,0,0, distance should be used to describe a spherical crystal.

For powders:

SampleGeometry: may be plate or cylinder.

Analyzers and Detectors

In a lot of cases the position of detectors or analyzers is valuable to know. In order to describe these positions we need to define a coordinate system. It will be a spherical one with the sample at the center. Gamma will measure values in the horizontal plane which includes the sample and the primary beam. In many cases this corresponds to two Theta. Zeta will describe angles in the vertical plane, perpendicular to the Gamma–plane. Zetas zeropoint is the horizontal plane. For area detectors Gamma and Zeta will point on the center of the detector. So for all all cases where the position of analyzers or detectors is vital to understand the data you need to give the following values:

- DGamma,AGamma(NumberOfDetectors or NumberOfAnalyzers) in degrees.
- DZeta,AZheta(NumberOfDetectors or NumberOfAnalyzers) in degrees.

D is the suffix for detetcor, A for analyzer.

Analyzers
Analyzers are quite similar to Monochromators. So the data to store are too quite similar.

- Analyzer. This should be a text value with the permissible values of NONE, CRYSTAL or CHOPPER.
- NumberOfAnalyzers. You need to know if there are more than one.

In some cases you might want to give more detail concerning analyzers. In the case of a crystal the apropriate values might be:

- ACrystal(NumberOfAnalyzers): text to describe crystal type and reflection used.
- AMosaicSpread(NumberOfAnalyzers): a value describing the spread of the crystal.

In the case of a chopper:

- AChopperType(NumberOfAnalyzers): text to describe chopper.
- AChopperSpeed(NumberOfAnalyzers) in revolutions per second.
- AChopperPhase(NumberOfAnalyzers) in degrees.

Detectors
Neutrons are expensive. In order to get the most from them people tend to install area sensitive detectors on their instruments. In order to understand the data gathered in them, you need to describe the geometry of these detectors. So you need to store:

- NumberOfDetectors
- DetectorType some text to describe the detector
- detector_width(NumberOfDetectors) physical size of detector in meters.
- detector_height(NumberOfDetectors) physical size of detector in meters.
- number_x(NumberOfDetectors) number of detector pixels in x direction.
- number_y(NumberOfDetectors) number of detector pixels in y direction.

- number_t(NumberOfDetectors) number of detector pixels in time direction for TOF–detectors.
- Name(NumberOfDetectors) variable name under which the detector array is stored in the data section.
- detector_efficiency(NumberOfDetectors) gives the relative efficiency of the detectors if you use different detctors on the same instrument.

1.5.3 Data

Let us start with a Definition:

Definition: Data are the normalised counts in the detectors and the instrument settings varied through the experiment.

So, everything that is hard coded into the instrument or will not be varied througout the experiment should be put in the instrument section, if you need to know it. Such things as H, K, L or Chi, Phi, Omega which will be varied in the experiment will be stored with the counts. This is dictated by common usage. Of course, different instruments need to store different data. Look for your type of instrument in the instrument hierarchy given in figure **??** and include all data items you find from the root to you specific case.

Let us now discuss the data in the hierarchy in more detail:

The first distinction to be made is between data gathered with a monochromatized beam or of those gathered with a white beam. In the case of a monochromatized incident beam we have to store the

- Wavelength in nanometer.

Now, with a monochromatized beam we can choose to make inelastic or elastic experiments. Elastic does not imply more data, but with inelastic data you will want to state the Q–vector scanned:

- Chi, Phi, Lambda
- H, K, L

For a triple–axis machine you need:

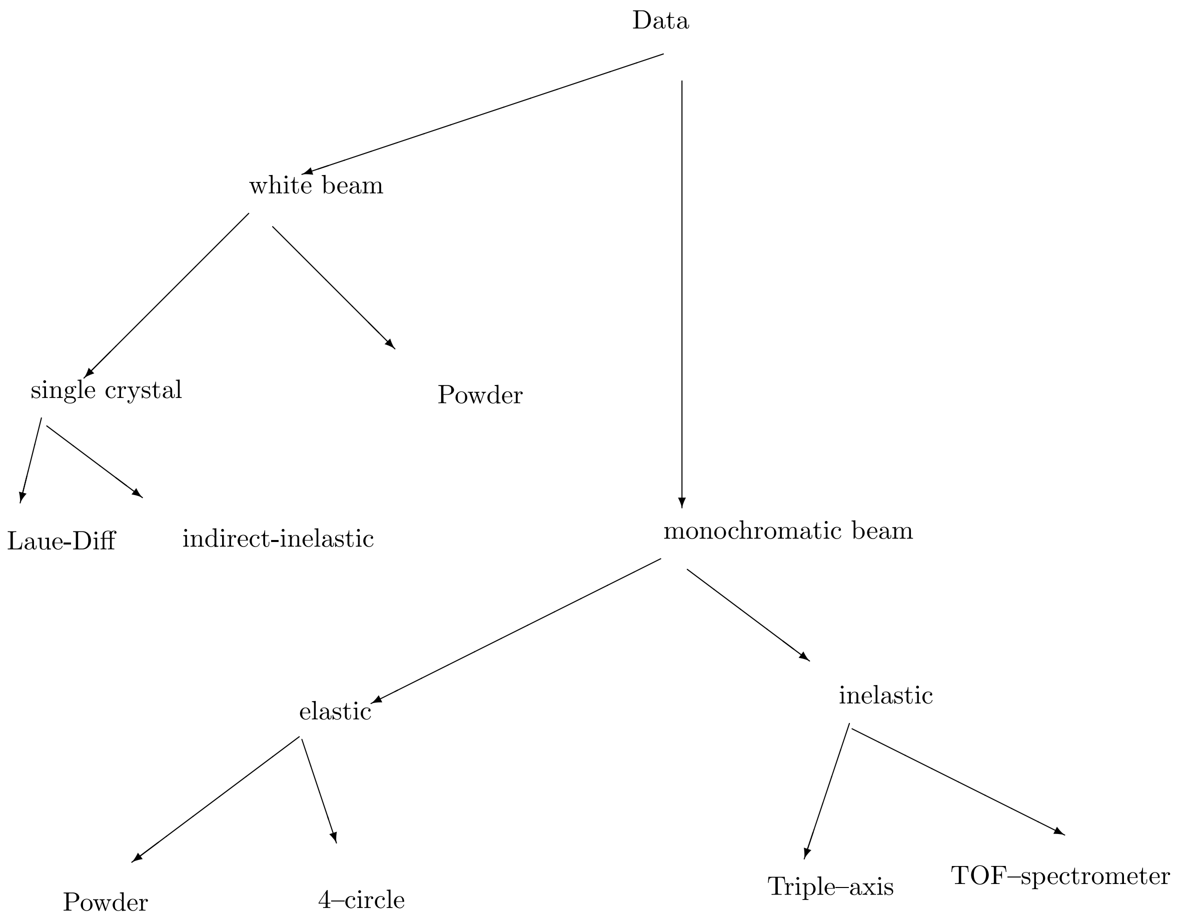

Figure 1.2: Data hierarchy for neutron scattering experiments

- NumberOfQValues
- Qx(NumberOfQValues)
- Qy(NumberOfQValues)
- Qz(NumberOfQValues)
- Energy(NumberOfQValues)
- QIntensity(NumberOfQValues)

For a TOF–machine, you need:

- The normalised counts in a array, named according to the name defined in the instrument section and with the dimensions in x, y and time, defined there.
- TimeMapping(NumberOfTimeChannels, NumberOfDetectors) maps time indices in the above array to time values in microseconds.

Lets get back to the elastic branch. Consider a four–circle diffractometer first. Most people will be content with integrated intensities, some will want to have the reflection profiles measured. So there are two formats:

- NumberOfReflections
- Data(9,NumberOfRelections) Number of reflection, Chi, Phi, Omega, H, K, L, F_{HKL}, Sigma.

The second format:

- NumberOfProfiles
- ProfileLength maximum length of profiles measured.
- Reflections(NumberOfProfiles, ProfileLength) All the counts.

For all four circle data you need:

- ScanType either Omega or Omega–Two Theta.

Powder people always want a profile:

- StartTwoTheta
- TwoThetaIncrement
- ProfileLength
- Counts(ProfileLength)

Now let us switch to the white beam part of the hierarchy. A distinction between single crystals and powder expreiments can be made: For single crystals you need to store:

- Chi, Phi, Lambda

in order to get the crystals orientation.

For an indirect geometry instrument you need to include:

- Analyzer Anergy in meV.

For both single crystal instruments types in this branch you need additionally:

- The normalised counts in a array, named according to the name defined in the instrument section and with the dimensions in x, y and time, defined there.
- TimeMapping(NumberOfTimeChannels, NumberOfDetectors) maps time indices in the above array to time values in microseconds.

TOF–flight powder machines are simpler. You include:

- ProfileLength
- Counts(ProfileLength)
- TimeMapping(ProfileLength) a array which maps indices in the Counts array to time values in microseconds.

1.6 Examples

Two example CDL files for ENDCDF will be given. CDl is the data descrption language of netCDF and its ASCII representation. From these files C- or Fortran77 code can be generated to write a netCDF file.

Please note, that netCDF requires data to be organised in three sections: in a section called dimension everything is declared, which is used as a dimension. In the section variables all variables are declared, not unlikely a variable declaration in Pascal or C. Items starting with a : are global attributes, something like variable:Units is a attribute belonging to that variable. Attributes are explanatory data to data. In the data section, finally, all the declared variables can be assigned values. Everything after `//` to the end of a line is comment. There are two examples: One for a simple powder diffractometer and the other one for the SXd diffractometer at ISIS.

```
netcdf puder {
// example of CDL-notation for a Powder diffractometer

dimensions:
ProfileLength = 3; // length of measured profile
Help2 =6;

variables: // and attributes

// general

:run_number = "007";
:Owner      = "Harry Hirsch, Waldweg 15, 64532 Dambach";
:Site       = "Sacclay";
:Date       = "26/4/1994";
:ExperimentDescription =
" J. Stink (4), 1993, p32-78";
:AdditionalData = "sister compound: 008";

// Instrument

:Instrument = "Powder 5";
:InstrumentDescription = " Z. Krist (4), 1992, 45-67";
:LocalContact = "J. Balu ";
:TOF_Flag = "No";
:BuildingBlocks = "Monochromator, Sample, Detector";

// Monochromators
:Monochromator = "Crystal";
:MCrystal = "Ge (110) ";

// Sample
:PhaseName = "Aquavit-B";
:SampleDescription =
"Bulletin f. Chemistry (7), 1888, 34-67";
float Temperature;
Temperature:Units = "Kelvin";
float Pressure;
Pressure:Units = "bar";
float Cell(Help2);
Cell:Units = "nm and degrees";
float Absorption;
Absorption:Units = "Rehe";
:SampleGeometry = "Cylinder";

// Detector

:DetectorType = "Powder-PSD";


// Data!

float  Wavelength;
Wavelength:Units = "nm";

float StartTwoTheta;
StartTwoTheta:Units = "degree";

float TwoThetaIncrement;
TwoThetaIncrement:Units = "degree";

int Counts(ProfileLength);

data:
// purely invented data

Temperature = 290;
Pressure = 1;
Cell = 2.3, 2.3 , 4.5, 90., 90., 90.;
Absorption = 0.8;

Wavelength = 0.12;
StartTwoTheta = 22.;
TwoThetaIncrement = 0.2;
Counts = 2,2,2;
}
```

```
netcdf SXD {
// example of CDL-notation for the SXD diffractometer at ISIS

dimensions:
Help1 = 3;
Help2 = 6;
number_x = 64;
number_y = 64;
number_t = 663;

variables: // and attributes

// general

:run_number = "6077";
:Owner      = "A. Nobel";
:Site       = "ISIS";
:Date       = "26/4/1994";
:ExperimentDescription =
" J. Explosives (4), 1993, p32-78";
:AdditionalData = "Vanadium run 6078";

// Instrument

:Instrument = "SXD";
:InstrumentDescription = " Z. Krist (4), 1992, 45-67";
:LocalContact = "C. Wilson ";
:TOF_Flag = "Yes";
:BuildingBlocks = "Distance1, Sample, Distance2, Detector";


// Distance

float Distance1;
Distance1:Units = "meters";
float Distance2;
Distance2:Units = "meters";

// Sample

:PhaseName = "Dynamit-5";
```

```
:SampleDescription =
"Bulletin f. Chemistry (7), 1866, 54-67";
float Temperature;
Temperature:Units = "Kelvin";
float Pressure;
Pressure:Units = "bar";
float Cell(Help2);
Cell:Units = "nm and degrees";
float UB(Help1, Help1);
float Absorption;
Absorption:Units = "Rehe";


// Detector

:DetectorType = "SXD";
float DGamma;
Gamma:Units = "degree";
float detector_width;
detector_width:Units = "mm";
float detector_height;
detector_height:Units = "mm";
:Name = "RawData";

// Data!

float RawData(number_x, number_y, number_t);
float TimeMap(number_t);
float Chi;
Chi:Units = "degrees";
float Phi;
Phi:Units = "degrees";
float Omega;
Omega:Units = "degrees";

data:
// purely invented data

Distance1 = 10;
Distance2 = 5;
Temperature = 290;
Pressure = 1;
Cell = 2.3, 2.3 , 4.5, 90., 90., 90.;
UB = 0.2, 0.3, 0.4 ,
     0.8, 0.9, 0.6,
     0.2, 0.3, 0.7;
Absorption = 0.8;

DGamma = 90.;
detector_width = 192.;
detector_height = 192.;
Chi = 56;
Phi = 123.444;
Omega = 45.;

// RawData and TimeMap omitted due to typing laziness


}
```

1.7 How to convert?

The conversion from existing raw dataformats to an exchange format will cost some programming effort. But the netCDF tools facilitate this process. The following steps are necessary to do the conversion:

- Define which data from your instrument needs to be incorporated in an exchange format file. Describe these data in the form of a CDL–file. Typical CDL–files for the most common neutron scattering instruments will be provided by ISIS.
- Review the file description with your instruments users. Make your users happy.
- generate a Fortran77– or ANSI–C–language template program from your CDL–file. This takes a few seconds.
- You must assign your values to the variables in the generated template code. So you need to edit this file and add calls to your existing preprocessing or raw data reading routines in order to fill in the variables. Might be, that you need to write a few routines to convert data.
- Compile your code and link it with the netCDF–library.

Assuming the existance of raw data reading routines this conversion program will be finished within a day by your star programmer. A student might take longer. As an example, a Fortran77 source code was generated from the CDL–file for a powder diffractometer given above. This Fortran code (with a few comments added) can be found in picture below.

Making an application program read the netCDF data is easy, too.

Example Fortran77 code for writing powder diffraction data. Generated automatically from the CDL–file above.

```
      program fgennc
      include 'netcdf.inc'
      integer  iret
* netCDF id
      integer  ncid
* dimension ids
      integer  ProfileLengthdim, Help2dim
* variable ids
      integer  Temperatureid, Pressureid, Cellid, Absorptionid, Waveleng
     1thid, StartTwoThetaid, TwoThetaIncrementid, Countsid
* variable shapes
      integer dims(1)
* corners and edge lengths
      integer corner(1), edges(1)
* data variables

C !!!!!!!! These variables you will have to fill in! !!!!!!!!!!!!!!
C
      real Temperature
      real Pressure
      real Cell(6)
      real Absorption
      real Wavelength
      real StartTwoTheta
      real TwoThetaIncrement
      integer Counts(3)
C------------------------------------------------------------
* attribute vectors
* enter define mode
      ncid = nccre ('puder', NCCLOB, iret)
* define dimensions
      ProfileLengthdim = ncddef(ncid, 'ProfileLength', 3, iret)
      Help2dim = ncddef(ncid, 'Help2', 6, iret)
* define variables
      Temperatureid = ncvdef (ncid, 'Temperature', NCFLOAT, 0, 0, iret)
      Pressureid = ncvdef (ncid, 'Pressure', NCFLOAT, 0, 0, iret)
      dims(1) = Help2dim
      Cellid = ncvdef (ncid, 'Cell', NCFLOAT, 1, dims, iret)
      Absorptionid = ncvdef (ncid, 'Absorption', NCFLOAT, 0, 0, iret)
      Wavelengthid = ncvdef (ncid, 'Wavelength', NCFLOAT, 0, 0, iret)
      StartTwoThetaid = ncvdef (ncid, 'StartTwoTheta', NCFLOAT, 0, 0, ir
     1et)
      TwoThetaIncrementid = ncvdef (ncid, 'TwoThetaIncrement', NCFLOAT,
     10, 0, iret)
      dims(1) = ProfileLengthdim
      Countsid = ncvdef (ncid, 'Counts', NCLONG, 1, dims, iret)
* assign attributes
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C  Here you will have to call you subroutines which read or define such
C  parameters as run_number, Owner, Site, Date ........
C
C The following code will write these attributes to the file
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      call ncaptc(ncid, NCGLOBAL, 'run_number', NCCHAR, 3, '007', iret)
      call ncaptc(ncid, NCGLOBAL, 'Owner', NCCHAR, 39, 'Harry Hirsch, Wa
     1ldweg 15, 64532 Dambach', iret)
      call ncaptc(ncid, NCGLOBAL, 'Site', NCCHAR, 7, 'Sacclay', iret)
      call ncaptc(ncid, NCGLOBAL, 'Date', NCCHAR, 9, '26/4/1994', iret)
      call ncaptc(ncid, NCGLOBAL, 'ExperimentDescription', NCCHAR, 27, '
     1 J. Stink (4), 1993, p32-78', iret)
      call ncaptc(ncid, NCGLOBAL, 'AdditionalData', NCCHAR, 20, 'sister
     1compound: 008', iret)
      call ncaptc(ncid, NCGLOBAL, 'Instrument', NCCHAR, 8, 'Powder 5', i
     1ret)
      call ncaptc(ncid, NCGLOBAL, 'InstrumentDescription', NCCHAR, 26, '
     1 Z. Krist (4), 1992, 45-67', iret)
      call ncaptc(ncid, NCGLOBAL, 'LocalContact', NCCHAR, 8, 'J. Balu ',
     1 iret)
      call ncaptc(ncid, NCGLOBAL, 'TOF_Flag', NCCHAR, 2, 'No', iret)
      call ncaptc(ncid, NCGLOBAL, 'BuildingBlocks', NCCHAR, 31, 'Monochr
     1omator, Sample, Detector', iret)
      call ncaptc(ncid, NCGLOBAL, 'Monochromator', NCCHAR, 7, 'Crystal',
     1 iret)
      call ncaptc(ncid, NCGLOBAL, 'MCrystal', NCCHAR, 9, 'Ge (110) ', ir
     1et)
      call ncaptc(ncid, NCGLOBAL, 'PhaseName', NCCHAR, 9, 'Aquavit-B', i
     1ret)
      call ncaptc(ncid, NCGLOBAL, 'SampleDescription', NCCHAR, 38, 'Bull
     1etin f. Chemistry (7), 1888, 34-67', iret)
      call ncaptc(ncid, Temperatureid, 'Units', NCCHAR, 6, 'Kelvin', ire
     1t)
      call ncaptc(ncid, Pressureid, 'Units', NCCHAR, 3, 'bar', iret)
      call ncaptc(ncid, Cellid, 'Units', NCCHAR, 14, 'nm and degrees', i
     1ret)
```

```
      call ncaptc(ncid, Absorptionid, 'Units', NCCHAR, 4, 'Rehe', iret)
      call ncaptc(ncid, NCGLOBAL, 'SampleGeometry', NCCHAR, 8, 'Cylinder
     1', iret)
      call ncaptc(ncid, NCGLOBAL, 'DetectorType', NCCHAR, 10, 'Powder-PS
     1D', iret)
      call ncaptc(ncid, Wavelengthid, 'Units', NCCHAR, 2, 'nm', iret)
      call ncaptc(ncid, StartTwoThetaid, 'Units', NCCHAR, 6, 'degree', i
     1ret)
      call ncaptc(ncid, TwoThetaIncrementid, 'Units', NCCHAR, 6, 'degree
     1', iret)
C-----------------------------------------------------------------
* leave define mode
      call ncendf(ncid, iret)
* store Temperature
      corner(1) = 1
      edges(1) = 1
C !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C Delete all the data statements in the following section and
C       replace them by assignements to your values.
C       Call your subroutines which return the values needed to fill
C       in the variables. Such as Temperature, Cell, ......
C
      data Temperature /290/
      call ncvpt(ncid, Temperatureid, corner, edges, Temperature, iret)
* store Pressure
      corner(1) = 1
      edges(1) = 1
      data Pressure /1/
      call ncvpt(ncid, Pressureid, corner, edges, Pressure, iret)
* store Cell
      corner(1) = 1
      edges(1) = 6
      data Cell /2.3, 2.3, 4.5, 90, 90, 90/
      call ncvpt(ncid, Cellid, corner, edges, Cell, iret)
* store Absorption
      corner(1) = 1
      edges(1) = 1
      data Absorption /0.80000001/
      call ncvpt(ncid, Absorptionid, corner, edges, Absorption, iret)
* store Wavelength
      corner(1) = 1
      edges(1) = 1
      data Wavelength /0.12/
      call ncvpt(ncid, Wavelengthid, corner, edges, Wavelength, iret)
* store StartTwoTheta
      corner(1) = 1
      edges(1) = 1
      data StartTwoTheta /22/
      call ncvpt(ncid, StartTwoThetaid, corner, edges, StartTwoTheta, ir
     1et)
* store TwoThetaIncrement
      corner(1) = 1
      edges(1) = 1
      data TwoThetaIncrement /0.2/
      call ncvpt(ncid, TwoThetaIncrementid, corner, edges, TwoThetaIncre
     1ment, iret)
* store Counts
      corner(1) = 1
      edges(1) = 3
      data Counts /2, 2, 2/
      call ncvpt(ncid, Countsid, corner, edges, Counts, iret)
C-----------------------------------------------------------------------
      call ncclos (ncid, iret)
      end
```

Chapter 2

Details of the scientific dataformats compared

2.1 HDF

HDF has been developed by the National Center for Supercomputing Applications at the University of Illinois, USA. HDF is a abreviation for Hierarchical Data Format. This name can be understood by a short description of the actual file format. HDF is organized like a book. At the beginning there is a table of contents which points to chapters, where the actual data are stored. So the internal structure of the file is like a hierarchy. A benefit of this technique is that HDF files are selfdescribing. In addition it is expandable. Currently HDF supports the following datatypes:

- 8– and 24–bit raster images and colour palettes for these pictures.
- textual annotations to data.
- free form tables of data.
- scientific data as multidimensional arrays of 8–, 32–, 64–bit Integers and 32– and 64–bit floating point values.

All data is stored in binary format. The people at NCSA promise to include further data formats to their system if necessary. Proposed are formats for TV and voice. Work is in progress to provide an interface for netCDF type data (Described later). HDF comes with a public domain library of access functions callable from C and FORTRAN. HDF is portable on all hardware platforms popular among scientists (PC, Mac, UNIX, VMS ....). Compression is supported only for bitmapped images, not for scientific data. But we would be free to add a compressed format to HDF. HDF is supported by AVS, IDL and a public domain visualisation package called SciAn. Furthermore software tools are available to manipulate HDF–data:

- an editor, which allows for easy viewing of file contents and some manipulations.
- a tool, which displays the list of contents of a HDF file.
- various conversion utilities for bitmap data.
- compression utilities for 8–bit raster images.

2.2 FITS

FITS is probably the oldest data format to be discussed. It has evolved in the astronomycal community since the 70's. Its primary purpose was to transport digital images. That's why FITS is a abreviation for Flexible Image Transport System. Over the years support for some more data structures was added. Now FITS is capable of storing:

- multidimensional floating point and Integer arrays.
- ASCII– and binary tables.

Further data formats such as binary dumps or compressed data are proposed but not yet standardized. A FITS file consists of a sequence of HDU's. A HDU is a header data unit. These HDU's contain a description of the data stored and the actual data themselves. Header Infomation is ASCII, while data may be stored as binary. FITS is maintained by a committee of the IAU (International Astronomers Union). FITS is also approved by NOST (Nasa organisation for Standardization) and is the official means of data transfer with NASA. Visualisation support for FITS is included in a special astronomers package for IDL. A AVS-module for reading FITS data is available. Futhermore a public domain FORTRAN library for reading and writing FITS data has been published. Besides that, support for FITS in the application programmers community seems to be sparse.

2.3 CIF

CIF is Crystallographic Interchange Format. CIF has been devised by the IUCR (International Union of Crystallographers) as a standardized method for storing and exchanging results of crystal structure refinements. The format is pure ASCII. Any possible parameter has got a unique name, specified in a dictionary. There are fields defined for observed structure factors. No means for storing profiles or large amounts of binary data are provided. No known data visualisation package supports CIF. Clearly, CIF should be supported by molecular graphics packages and structure drawing programs.

2.4 CDF

CDF is a predessor of netCDF, developed by the National Space Science Center, USA. NetCDF branched from CDF in some stage, both data formats developed independently but ended up in quite similar specifications. So, CDF gives the same functionality (besides minor differences) as netCDF. Yet, netCDF seems to have the better support in the application developers community. Nevertheless, IDL and AVS can read CDF data.

2.5 STEP/Express

STEP/Express is an effort of the engineering community to simplify the data exchange between all stages of designing, producing and maintaining a product within an enterprise. This means integrating CAD, CAM etc applications with database management applications, version control programs and management information systems. This whole effort is an ISO standard. STEP/Express consists of several parts:

- a data description language called Express, which is used to model data within STEP
- a Data interchange format based on Express descriptions of the data. This file format is defined as ASCII files in order to obtain maximum portability across platforms.
- a standard interface to DBMS-systems
- for the future the integration of STEP with knowledge based systems is planned.

Some software for interpreting Express files and a programming interface for a database based on Express is available at RAL. There are thoughts on using this system at ISIS to store run and instrument information in a database. However, the STEP interchange format is not supported by visualisation packages, only translators to some CAD systems exist. ASCII files as means of data storage are very portable but impose some performance overhead on any application due to the necessity of interpretation. Furthermore there is the danger of a loss in precesion in floating point values stored in ASCII-files.