DRAFT of Proposed Data Standard for the APS

Jon Tischler, ORNL & UNI-CAT

INTRODUCTION

For the past ~15 years I have been taking data at synchrotron radiation facilities. In that period I have dealt with approximately 10 different data file "standards". Each standard was, of course, unique to a particular program. At first, writing a program to read in the data file and output one that I could then plot was interesting; but about 12 years ago I stopped enjoying it. 10 years ago I presented a short paper at the 3rd US Synch. Rad. Instr. Conf. (at BNL) describing a possible data standard based on the ISO-8211 (Specification of a data descriptive file for information interchange). This was thoroughly ignored; even I did not implement it. Although the ISO-8211 standard is still there, the HDF data standard now exists, and it is supported by the National Center for Supercomputing Applications (NCSA), and many data analysis programs. The existence of HDF libraries in C and FORTRAN for many different computer systems (UNIX's, Mac, DOS,...) make it an obvious first choice. With existing libraries, we have less to do, and it makes implementing the standard both easier and more rigorous.

Other standards exist for data files. The netCDF standard provides elements that make it very attractive for storing experimental data. This includes the ability to assign units to data, and other informative attributes. Unfortunately netCDF provides no way to group data into scans. There is no mechanism for measuring 6 scans of {theta, 2theta, scintillator, and clock}. When you name a variable "theta" in a netCDF file, that name is global to the entire file.

The HDF standard stores values as either arrays (of any dimension called an SDS) or tables (called a vdata). Since most of the scanned data appears to be tables (a table of "theta", "scint"), this at first appears to be a good choice. However, vdatas have no means for identifying things such as the rank of the data, axes, or units, etc. HDF, also has something called a vgroup, which is way to group SDS's (and anything else) together. This allows HDF to form data into scans, so multiple scans can be put into one file.

This leaves us with a quandary, netCDF is good at describing data and HDF is good at organizing data, but not vice versa. That was the situation until early this year. The new version of HDF (HDF3.3r3) allows an SDS to look just like a netCDF variable with the ability to set dimension scales, attributes (like units) etc. Plus you can still group these SDS's into scans (which will be called "entrys") to put multiple scans into one file.

HDF DOCUMENTATION

Documentation for HDF is available via anonymous ftp from `ftp.ncsa.uiuc.edu`, in:

```
/Documentation/HDF.Vset2.1/
/Documentation/HDF_getting_started/
/Documentation/HDF3.3/
and the code is in /HDF/HDF3.3r3/
```

It is important to read the "HDF_getting_started" and the "HDF.Vset" manuals. Without Vset, HDF is rather useless, since it can only store images, large groups of numbers, and text. It needs Vset to specify relations between things (like grouping pieces of information together into a scan).

In writing this standard, the deciding factor at all times has been to organize the data so that it can be easily read and plotted by a generic program (not just by a specialized program customized to a particular experiment). This was the key factor used when closing among the various approaches possible with HDF.

CREATION of the DATA FILE

The method chosen for data storage is designed to make as much use as possible of the HDF standard which reduces the number of extra definitions. Thus just about everything is to be stored as a Scientific Data Set (SDS). And the individual scans are to be Vgroups with the sequential names entry1, entry2, entry3,...

The steps for making a data file are:

I) Open and initialize the file for writing for both Vgroups and SDS.
   A) Create Vgroup named "entry_default"
      1) Insert SDS's with global defaults
      2) detach "entry_default"
   B) Create a Vgroup named "entry1"
      1) Add SDS's for simple parameters (not scanned) and add to "entry1"
      2) Create Vgroup "data1" to hold the scanned information
         a) Create an SDS for each scanned variable (add appropriate attributes to it)
         b) add special attributes (axis, signal, ...) to scanned variables
         c) detach "data1"
      3) detach "entry1"
   C) Create a Vgroup named "entry2"
      1) repeat steps 1-3 from B
   D) Continue creating and filling "entry*n*" until done
   E) Close the file

OUTLINE of a TYPICAL DATA FILE

Useful variables to predefine, since they are used frequently.

```
#define VERSION  "0.1 alpha"
#define LOCATION "NSLS:X15A"
#define PROG_NAM "ORDIF"
#define MAX_SDS_LEN 23          /* maximum allowed length of an SDS name */
#define MAX_MOTOR_LEN 27        /* maximum allowed length of a motor name*/
int32 dim1[1]={1};
int32 edge1[1]={1};
int32 start0={0,0,0,0,0,0,0,0,0};/* max possible rank defined at start*/
int32 all_ones = -1;     /* used to flag non-existent data */
int16 one16 = 1;       /* a 16 bit one, used with DFNT_INT16 */
char  *file_name = "pdq_may20.hdf";
char  *user_name = "John Q. Public";
char  *user_mail = "Chemistry Dept.\nBerkeley U.\n CA  11111";
char  *user_phone = "900-555-1212";
char  *user_fax = "900-555-1213";
```

The first step in creating a data file is to open it, and set it up for HDF access by both the SD and Vgroup interfaces.

```
sfid = SDstart(file_name, DFACC_CREATE);  /*file name is "pdq_may20.hdf" */
vfid = Hopen(file_name,DFACC_RDWR,0);
Vstart(vfid);
```

Add global attributes to the file that apply to the *entire* file.

```
i = SDsetattr(sfid,"file_name",DFNT_CHAR8,strlen(file_name),file_name);
i = SDsetattr(sfid,"version",DFNT_CHAR8,strlen(VERSION),VERSION);
i = SDsetattr(sfid,"MAX_SDS_LEN",DFNT_INT16,1,MAX_SDS_LEN);
```

```
i = SDsetattr(sfid,"MAX_MOTOR_LEN",DFNT_INT16,1,MAX_MOTOR_LEN);
```

Create the Vgroup named "entry_default" to hold all of the information for the first entry.

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry_default");    /* this name is required */
Vsetclass(vgN,"APS_default");     /* this class is required */
id = -1;
```

Create the SDS's to get default values.

```
sid[++id] = sdstring(sfid,"location",LOCATION);  /* required */
sid[++id] = sdstring(sfid,"user_name",user_name);      /* required */
i = SDsetattr(sid[id],"user_mail",DFNT_CHAR8,strlen(user_mail),user_mail);
i = SDsetattr(sid[id],"user_email",DFNT_CHAR8,strlen(user_email),user_email);
i = SDsetattr(sid[id],"user_phone",DFNT_CHAR8,strlen(user_phone),user_phone);
i = SDsetattr(sid[id],"user_fax",DFNT_CHAR8,strlen(user_fax),user_fax);
sid[++id] = sdstring(sfid,"program_name",PROG_NAM);
sid[++id] = sdstring(sfid,"diffractometer","eulerian");
```

Group the SDS's into the entry_default group.

```
for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vdetach(vgN);
```

Create a Vgroup named "entry1" to hold all of the information for the first entry.

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry1");
Vsetclass(vgN,"APS_entry");        /* this class is required */
id = -1;
```

Create an SDS to describe certain general properties of this particular scan:

```
sid[++id] = sdstring(sfid,"date","22-feb-1996");      /* required */
sid[++id] = sdstring(sfid,"hour","23:59:59.12 UT");        /* required */
sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
sid[++id] = sdstring(sfid,"entry_intent","data");    /* required */
sid[++id] = sdstring(sfid,"sample","BiAsO on graphite #12345");
sid[++id] = sdstring(sfid,"title","First data on compound");

hutch_pressure = 1.001;
sid[++id] = SDcreate(sfid,"hutch_pressure",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"barometric pressure","atmosphere","%.3","");
SDwritedata(sid[id],start0,NULL,dim1,&hutch_pressure);

hutch_temperature = 27.5;
sid[++id] = SDcreate(sfid,"hutch_temperature ",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"room temperature","Celsius","%.2","");
SDwritedata(sid[id],start0,NULL,dim1,&hutch_pressure);

temperature = 100.5;
sid[++id] = SDcreate(sfid,"temperature",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"temperature","Kelvin","%.3","");
SDwritedata(sid[id],start0,NULL,dim1,&temperature);

mono_energy = 8.047;
sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"energy","keV","%.4","");
SDwritedata(sid[id],start0,NULL,dim1,&mono_energy);
```

Make a group to hold the scan, and write the scan data. This is a theta-2theta scan with 50 intervals(51 points). The scan range is from theta=[10,20], and 2theta=[20,40].

```
vgD = Vattach(vfid,-1,"w");
Vsetname(vgD,"data1");    /* ALWAYS "data1", even for "entry2" */
Vsetclass(vgD,"APS_scan");    /* this class name required */
Did = -1;
dims[0] edge[0] = = 51;  /* scan of 51 points, rank is 1 */

Dsid[++Did] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"theta","degrees","%.3","");
lo = 10.; hi = 20.;  SDsetrange(Dsid[Did],&lo.,&hi);
SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,1,"theta");
SDwritedata(Dsid[Did],start0,NULL,edge,column1);

Dsid[++Did] = SDcreate(sfid,"2theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"2theta","degrees","%.3","");
lo = 20.; hi = 40.;  SDsetrange(Dsid[Did],&lo,&hi);
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,1,"2theta");
SDwritedata(Dsid[Did],start0,NULL,edge,column2);

gain = 1.e8;
Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,&gain);
SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"photons at ic1","photons","%.3","");
SDsetcal(Dsid[Did],12345.,0.,22.,0.,DFNT_FLOAT32);
/* calibration contains ic gain and absorption to give photons */
SDsetattr(Dsid[Did],"serial_no",DFNT_CHAR8,strlen("1234-AX"),"1234-AX");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column3);

absorption_corr = 0.237;  deadtime=1.e-6;
Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"absorption_corr",DFNT_FLOAT32,1,&absorption_corr);
SDsetattr(Dsid[Did],"deadtime",DFNT_FLOAT32,1,&deadtime);
SDsetdatastrs(Dsid[Did],"lambda","photons","%.0","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column4);

Dsid[++Did] = SDcreate(sfid,"clock",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"count_time",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"clock","1/262144 seconds","%.3","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column5);
    /* attach SD's together to make data1*/
for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
Vdetach(vgD);
    /* attach SD's and vgD to make entry1*/
for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vaddtagref(vgN,DFTAG_VG,vgD);
Vdetach(vgN);
```

Repeat for each following entry

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry2");
Vsetclass(vgN,"APS_entry");
id=0;

etc...

for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vaddtagref(vgN,DFTAG_VG,vgD);
Vdetach(vgN);
```

Finally close the file

```
SDend(sfid);
Vend(vfid);
Hclose(vfid);
```

Column names are set by each station for what ever is convenient. However, for the automatic plotting and analysis of data, it is necessary to identify certain columns that have special meaning, such as the axes and the primary, signal, and the count time for a point. This is done with predefined attributes that are attached the appropriate SDS. In the example above the following five lines identify certain SDS's as the preferred x-axis, signal, etc.

```
SDsetattr(sid[id],"axis",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"primary",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"signal",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"count_time",DFNT_NONE,1,NULL);
SDsetattr(sid[id],"I_monitor",DFNT_NONE,1,NULL);
```

The full list of predefined attributes are listed in Appendix C. Note how the counting time was named "clock" but identified by the attribute "count_time" and the time in seconds was set by a "units" attribute to "sec". Also, the detector column was identified as both "primary" and "signal". The attribute "primary" flags the preferred variable(s) for plotting and "signal" is the measured variable with the closest relation to "primary". These two functions are separate because you might prefer to plot a quantity derived from the signal. If primary had been chosen to be the signal normalized by the monitor and corrected for detector deadtime, the source code would be changed to.

```
Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,&one16);
SDsetdatastrs(Dsid[Did],"lambda","photons","%.0","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column4);

for (i=0;i<dims[0];i++) {
    dt = 1. / (1. - exp(-column6[i]/262144./deadtime));
    column6[i] = column4[i]/column3[i] * gain * absorption_corr * dt;
}
Dsid[++Did] = SDcreate(sfid,"corrected_signal",DFNT_INT32,1,dims);
strcpy(equation,"lambda/ic1*gain*absorption_corr/(1.-exp(-column6[i]/262144./deadtime)));
SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,&one16);
SDsetdatastrs(Dsid[Did],"reflectivity","absolute","%.5","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column6);
```

This way, the reflectivity could be automatically plotted. This feature could also be quite useful in EXAFS data where the desired data is a normalized sum of numerous detectors. So you would have the individual counters labeled signal=1, signal=2, .. .and only one column labeled with primary=1. A value of 0 for primary, signal, and axis indicates no meaning. This allows the primary, signal, and axis attributes to be present without implying anything.

Some of the attribute names that were presented above are official names, and should only be used for the officially declared purpose and in the officially declared manner. The official names are all listed in APPENDIX C. A short list of required attributes is given here. In any scan that expects to be automatically plotted, some attributes must be preset, they are primary signal, and axis.

require	name	Description
*	`primary`	identifies primary variable(s) for plotting. 1=most important, 2=less important, ...
*	`signal`	most likely measured data channel to be plotted. (value is importance)
*	`axis`	preferred independent axes for plotting, 1=x-axis, 2=y-axis, ..., up to at least rank of data. Value of axis can range from 1 thru rank of data. e.g. axis=1 or 2 for surfaces. Value can exceed the rank for labeling purposes (i.e. FWHM_5).

These named attributes are used to identify the role of individual SDS's. This provides the plotting and analysis programs a means to process the data.

This indirect reference method (using signal, primary, axis) was chosen since there may many input channels that are collected in each scan, and the "primary" "signal" may change depending upon what the user is interested in for that scan. For example, the user has both a scintillator and a PIN diode operating. "signal" will be "scintillator" or "PIN" depending upon which detector is in use at the moment. Thus when "signal" is "PIN", the PIN column should be plotted. The same reasoning applies to the independent variables. For a scan in reciprocal space along the [010] direction where h,k,l,theta,2theta,chi,phi are all collected, it is necessary to identify that the k column is best choice for x-axis. If the scan were along the [011] direction, then a computed axis (not actually measured) would be the best choice.

Note on attribute "units". Acceptable units are described in the "udunits" file available with the netCDF documentation. Also included are utilities that can do unit conversions.

Utility function used above:

```
int32 sdstring(int32 sfid, char *sd_name,char *string)
/* this routine writes an SDS which only contains a single char string */
{
    int32       sid;
    int32       dim;
    int32       start=0;

    dim = strlen(string);
    sid = SDcreate(sfid,sd_name,DFNT_CHAR8,1,&dim);
    SDwritedata(sid,&start,NULL,&dim,string);
    return sid;
}
```

DEFAULTS

HDF does not include a provision for default attributes (or any other kind of defaults). However, their use is desirable, especially when the data file consists of many small scans where recording a large number of attributes for each one would noticeably increase the file length. Although HDF does not explicitly provide for default attributes, it is possible to include them while maintaining a legal HDF file. Since this default information is not part of the HDF standard, generic HDF programs (such as Mosaic, Spyglass, ...) will not will not recognize the meaning of this default information. However, these generic programs will still be able to process the files, since they were written using standard HDF routines. As a result of all this, defaults will be most useful for attributes that are defined in this standard (e.g. "diffract_axis"), and least useful for attributes that are generally recognized by all HDF programs (e.g.-"units").

Having said all that, the procedure for defining default attributes is as follows:

I) Create Vgroup named "entry_default", and class "APS_default"
   A) Create an SDS of minimal size (rank=1,dim=1) whose name is associated with the default information.
      1) Set all of the default attributes to that empty SDS.
   B) Repeat from A for the next SDS name to have default attributes.
   C) Attach these SDS's to the vgroup "entry_default"
   D) detach "entry_default"

The rule for defaults is that if an SDS has a default attribute in "entry_default", then that attribute applies unless explicitly overridden in the entry where it is used. In general the value of an SDS is not defaultable. However, for an SDS such as "program_name" which may not change in a file and is not analyzable, it may be used. Do not default an SDS such a the theta axis, since this would confuse generic plotting programs. There is one place where HDF does provide for a type of default. When axes are defined for an SDS by a call to SDsetdimscale, the name of the axis is assumed to be global for all SDS's that use an axis of that name. Thus when using a CCD with axes defined as length in mm, only a single axis definition is necessary even if many CCD images are saved.

The following code is an example for creating a default attributes.

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry_default");     /* this name is required */
Vsetclass(vgN,"APS_default");      /* this class is required */
id = -1;

sid[++id] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dim1);
SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,1,"theta");

sid[++id] = SDcreate(sfid,"2theta",DFNT_FLOAT32,1,dim1);
SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,1,"2theta");

sid[++id] = SDcreate(sfid,"h",DFNT_FLOAT32,1,dim1);
SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,1,"h1");

sid[++id] = SDcreate(sfid,"k",DFNT_FLOAT32,1,dim1);
SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,1,"k1");

sid[++id] = SDcreate(sfid,"l",DFNT_FLOAT32,1,dim1);
SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,1,"l1");

for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vdetach(vgN);
```

APPENDIX A
entry_analysis

`entry_analysis`, allowed values are:

entry_analysis	description
`diffraction/misc`	for data scans along theta,hkl,Q... General reciprocal space stuff.
`diffraction/peak`	like diffraction/misc, but requests peak analysis
`diffraction/powder`	only for powder diffraction scans
`crystallography`	mostly integrated intensities, scans may or may not be stored
`XAFS/EXAFS`	EXAFS
`XAFS/XANES`	XANES
`XAFS/misc`	everything else
`DAFS/EXAFS`	Just like EXAFS, but in diffraction
`DAFS/XANES`	
`DAFS/misc`	
`NONE`	status of something, no analyzable information (e.g.. slit status)

entry_intents

`entry_intent`, allowed values are:

entry_intent	description
`alignment`	alignment scans, not to be processed for data.
`calibration`	calibration information
`data`	actual data that is to be analyzed and used to draw conclusions. What you came to the synchrotron to measure.
`status`	status of something, no analyzable information (e.g.. slit status)
`misc`	miscellaneous
`testing`	generic beamline testing (not data)

Note on entry_intent and entry_analysis:

The choice of entry_analysis's is based on the kind of *automatic* analysis that might be applied. It is not a scientific category. For example, data identified as EXAFS might be automatically put through routines that find the edge, remove the edge jump, do the spline, and Fourier transform. Such automatic treatment would be inappropriate for powder diffraction data. Thus "entry_analysis" is intended to be used to simplify the reduction of data, *not* to describe the science. "entry_intent" is intended to separate the data from the many alignment scans that one does. It allows the experimenter (at some later time) to request plotting of only the usable data, or just the calibration peaks, ...

APPENDIX B
COMPLETE LIST of DEFINED SDS NAMES:

Names listed in **bold** are required, everything else should be implemented as needed. The first group of SDS's should not be part of a "data1" group, the second group can be placed anywhere in the entry.

SDS name	type	Description
hour	CHAR8	Time of day when scan was started any time zone may be used, but you must use include a time zone, which can be letters "GMT" or the offset "-5". Use 24 hour clock, not AM/PM. ex. "10:22:00.2 PDT" or "10:22:00 -7"
date	CHAR8	Date when scan was started. Do not use just two digits for year ex. "10-Mar-1999"
user_name	CHAR8	Name of the principal user. (preferred location is in entry_default) ex. "John Q. Public"
location	CHAR8	Where the data were measured. Include the facility as well as the beam-line used. The purpose is to identify to the experimenter where the data were taken, and to help figure out who owns the data file should it get misplaced. Preferred usage is in "entry_default" ex. "NSLS:X23"
title	CHAR8	A title, something you would like to see on a plot. A good thing to put here is the reason for this particular scan. ex. "measure mosaic width"
sample	CHAR8	Name or identifier for the current sample ex. "BiAsO on graphite #12345".
program_name	CHAR8	Name of program writing this data. Use attributes for version. ex. "ordif"
command_line	CHAR8	Command line used to initiate the current scan ex. "scan energy from 10 to 10.3 keV, 10 steps"
ABORT	CHAR8	Flags a scan that was aborted. This is mandatory for incomplete or aborted scans. It should not be present for properly completed scans. Use the text part to indicate what was wrong, e.g.. "forgot to open shutter", or "hard limit on theta". ex. "hard limit on monochromator"
constraint	CHAR8	Describes what is held constant by the scan (useful on plots). The default assumption here is based upon the particular experiment. ex. "chi=25" or "hkl=(0 0 1.2), Ef=30 meV"
plot_type	CHAR8	Flags preference for plottable data. Currently defined types are line, 3d, contour, and image. This tag is not required since the structure of the data itself suggests a default. i.e. a raster image should default to a picture, a 1-d scan defaults to line plot, etc. ex. "image"
hkl_near	3*FLOAT32	The hkl that best describes the scan. May even be used when doing an hkl scan, since this would be used to identify the reflection scanning near. Useful for sorting and labeling. ex. {1,1,1.5} or {3,1,1}

hkl_surf	3*FLOAT32	The hkl of the surface normal. ex. {1,1,1.1}
orient_matrix1	9*FLOAT32	First orientation matrix (orient_matrix2, ... are also allowed) {u11,u21,u31, u12,u22,u32, u13,u23,u33}
unit_cell1	6*FLOAT32	First unit cell a,b,c (set units), alpha beta gamma in degrees. ex. {5.431,5.431,5.431,90,90,90}
unit_cell_err1	6*FLOAT32	The errors associated with first unit cell parameters a,b,c, alpha beta gamma in degrees. ex. {.0011,.0013,.001,.1,.1,.11}
F1	2*FLOAT32	The complex structure factor for unit_cell1. For units, use either "electrons" or the scattering length. ex. {51.6,2.2}, with "units" = "electrons" or {145.4,6.2}, with "units" = "fm"
hklN1	3*FLOAT32	reference vector for computing ψ_{N1}, representing orientation of the sample relative to a specified direction.
diffractometer	CHAR8	Describing the diffractometer geometry (entry_default preferred) ex. "eulerian" or "kappa"
mono_type	CHAR8	Description of the monochromator ex. "Si (111) double crystal, focused"
mono_dspace	FLOAT32	dspacing of monochromator, include units. ex. 0.31356 with "units" = "nm"
analyzer_dspace	FLOAT32	dspacing of analyzer, include units ex. 3.1356 with "units" = "Angstroms"
		The above SDS's should not be in a "data1" group, the following may be anywhere, as appropriate.
hutch_pressure	FLOAT32	Local barometric pressure, useful if you want absolute information from ion chambers. (be sure to set the units!). ex. 760.1, with "units" = "torr"
hutch_temperature	FLOAT32	Air temperature inside of hutch. Include the units! ex. 28.3, with "units" = "Celsius"
temperature	FLOAT32	The sample temperature at start of scan. If you don't have a furnace or refrigerator, this is probably of no use at all. Include the units. ex. 123.7 with "units" = "Celsius"
pressure	FLOAT32	Pressure of sample. This is not for storing the atmospheric pressure (see hutch_pressure). Include the units. ex. 2.3 with "units" = "bar"
mono_energy	FLOAT32	The monochromator energy. Include units. ex. 8.3124 with "units" = "keV"
mono_wavelength	FLOAT32	The monochromator wavelength. Include units. ex. 1.5224 with "units" = "Angstroms"
analyzer_energy	FLOAT32	The analyzer energy. Include units! ex. 8.3122 with "units" = "keV"

analyzer_denergy	FLOAT32	(analyzer energy) - (monochromator energy). Include units. ex. -200 with "units" = "meV"
analyzer_wavelength	FLOAT32	The analyzer wavelength. Include units. ex. 0.15224 with "units" = "nm"
B_field	3*FLOAT32	Magnetic field applied to sample, include units. ex. {0.5, 0., 0.} with "units" = "tesla"
ring_current	FLOAT32	storage ring current, include the units. At start of scan. ex. 770., with "units" = "mAmp"
ring_energy	FLOAT32	storage ring energy, include the units. At start of scan. ex. 7.1, with "units" = "GeV"
undulator_gap	FLOAT32	Undulator gap, include the units. ex. 7.1, with "units" = "mm"
undulator_energy	FLOAT32	Undulator peak energy, include the units. ex. 7.1, with "units" = "keV"
xray_tube_voltage	FLOAT32	Voltage on an xray tube (not for synchrotrons). ex. 25, with "units" = "kV"
xray_tube_current	FLOAT32	Current on an xray tube (not for synchrotrons). ex. 30, with "units" = "mA"

APPENDIX C
COMPLETE LIST of DEFINED SDS ATTRIBUTES:

First come the global attributes, then ones that are local to an SDS. Attributes in **boldface** are required.

attribute	type	Description
file_name	CHAR8	original file name when data file was opened.
version	CHAR8	as a global attribute it is the version of the data file. This attribute can also be applied to other SDS's (such as "program_name") if desired.
MAX_SDS_LEN	INT16	maximum length of an SDS name used (used for allocating space in analysis)
MAX_MOTOR_LEN	INT16	maximum length of a motor name used (used for allocating space in analysis)
		The above SDS attributes are global, the following are local to an entry.
signal	INT16	Indicates importance of column for plotting. 1 is mot important. Not all columns should have a value. Only ONE SDS can be signal=1.
primary	INT16	Indicates importance of column for plotting. 1 is most important. Not all columns should have a value. Only ONE SDS can be primary=1.
axis	INT16	Indicates the default independent axes for plotting. Value between 1 and rank of data are required. Only one SDS can have each value.
count_time	NONE	Attached to the column (or single value) which contains the time counted (in some units). Only its presence counts. Only ONE SDS can be count_time. Be sure to set units to be able to get count time into seconds.
equation	CHAR8	Its presence indicates a derived quantity. the text gives the equation used.
gain	FLOAT32	A switchable gain, such as might occur with an ion chamber. It is usually attached to a column.
deadtime	FLOAT32	The dead time in a detector, such as a scintillator. It's usually attached to a column with the detector. Include the units.
serial_no	CHAR8	serial no. for a detector, or any other equipment that you want to identify.
part_no	CHAR8	Manufacturers part number, to identify a type of detector for example.
Q	Q	Identifies the Q value defined as 2π/d. Units are set with a "units" attribute.
analyzer	CHAR8	Identifies presence of an analyzer, and its type if present. Attach to detector. ex. "Ge (220)"
I_monitor	NONE	Attached to the column with best intensity measurement before sample. Only its presence counts. Only ONE SDS in an entry may be I_monitor.
polarization_vec	4*FLOAT	Stokes vector (polarization state) at detector . Usually attached to I_0. {1, .8, 0., 0.} = $\{I_\sigma+I_\pi,\ I_\sigma-I_\pi,\ I_{(+45°)}-I_{(-45°)},\ I_{RH}-I_{LH}\}$
polarization_mat	16*FLOAT	Polarization transfer matrix for detector, Stokes matrix (no units).
sample_field	3*FLOAT	Orientation of a field on sample (no units, just direction)
flipper	CHAR8	Status of flipper. (only used for neutron data) ex. "on" or "off"
		The following four attributes are for use with a "user_name" SDS.
user_mail	CHAR8	Where to send the data, when you have it in your hand, and you do not own it or want it. ex. "Chem. Dept., Berkeley U., CA 11001"

user_email	CHAR8	email address of user. ex. "joe@pdp8.phys.mit.edu"
user_phone	CHAR8	Telephone number of user. ex. "900-555-1212"
user_fax	CHAR8	FAX telephone number of user. ex. "900-555-1213"
		When a I_monitor is present, the following 6 may be attached to a detector.
air_path	FLOAT32	path length thru air from front of I_monitor to detector (in mm).
He_path	FLOAT32	path length thru He path from front of I_monitor to detector (in mm).
Be_path	FLOAT32	path length thru Be (windows) from front of I_monitor to detector (in mm)..
plastic_path	FLOAT32	path length thru plastic (windows) from front of I_monitor to detector (mm).
efficiency	FLOAT32	detector efficiency (at the current energy).
absorption_corr	FLOAT32	total absorption , computed from all the paths and the efficiency.
		The following group of attributes refers to crystallography
diffract_axis	CHAR8	Identifies a crystallographic diffractometer axis, regardless of the name, allowed values are described below:
	2theta	Identifies the 2θ axis. Also use a units attribute.
	theta	Identifies the θ axis. Also use a units attribute.
	chi	Identifies the χ axis. Also use a units attribute.
	phi	Identifies the ϕ axis. Also use a units attribute.
	alpha	Angle between incident beam and surface (use with units)
	beta	Angle between exit beam and surface (use with units)
	h1	h value in reciprocal space for orientation matrix 1, dimensionless
	k1	k value in reciprocal space for orientation matrix 1, dimensionless
	l1	l (lower case L) value in reciprocal space for orientation matrix 1, dimensionless
	psi01	ψ_0, represents orientation of the sample relative to the symmetric geometry.
	psiN1	ψ_N, represents orientation of the sample relative to a specified direction.
		h1,k1,l1,psi01,psiN1,hklN1 can be repeated with a 2 (etc.) for orient_matrix2
		the following refers to peak fitting
FWHM_1 FWHM_2	2*FLOAT	FWHM and error of SDS with axis=1 as x-axis Use FWHM_2 for width using axis=2, etc.
Integral_1 Integral_2	2*FLOAT	Integral and error of SDS with axis=1 as x-axis (=2 for y-axis) Use Integral_2 for integral using axis=2, etc.
voigt_1 (or) gaussian_1 (or) lorentzian_1	14*FLOAT	fit to a voigt function with axis=1 as x-axis (use voigt_2 for axis=2) The parameters are: background_offset, slope, amplitude, x-position, shape, FWHM, Integral followed by errors in each of these values.

APPENDIX D
EXAMPLE OF A MOTOR BLOCK (VDATA):

To store a list of motor (or other actuator) positions you can use a vdata. It is probably better than using a multitude of SDS's. More than one motor block vdata may be present, as long as the names as set by VSsetname(...) differ. A motor block would usually be expected to be placed an entryN vgroup along with the title, sample, etc.

```
vs = VSattach(vfid,-1,"w");
VSdefine(vs,"MOTOR_NAME",DFNT_CHAR8,MAX_MOTOR_LEN);
VSdefine(vs,"POSITION",DFNT_FLOAT32,1);
VSdefine(vs,"units",DFNT_CHAR8,30);
VSsetfields(vs,"MOTOR_NAME,POSITION,units");
VSsetname(vs,"diffractometer");
VSsetclass(vs,"motor_block");               /* required */
VSwrite(vs,array_of_data,Nmotors,FULL_INTERLACE);
VSdetach(vs);
```

First fill in array_of_data with the motor names, positions, and units. Then write the Vdata, and set name and class.

This method of storing data is good for analysis programs cognizant of this standard. However, generic plotting programs will largely ignore this type of data (it won't cause problems, but it won't do much good either).

APPENDIX E
EXAMPLES

```
/*
CFLAGS = -g -qextchk -DANSI -I/usr/local/include
LIBFLGS = -lm -lzzt -L/home/zzt/c/util -lnetcdf -ldf -L/usr/local/lib

examples: examples.o examples.c
        cc -o examples $(CFLAGS) $(LIBFLGS) examples.o
        @chmod 755 examples
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#ifdef __THINK__
#define MAC
#define macintosh
#endif
#include <mfhdf.h>

#define BELL            '\007'          /* bell character */
#define LEN_str         40              /* generic string length */
#define MAX_ID          50
#define PI              3.14159265
#define LOCATION        "APS:6-BM-R"
#define PROG_NAM        "ORDIF"
#define VERSION         "0.1 alpha"
#define MAX_SDS_LEN     23              /* maximum allowed length of an SDS name */
#define MAX_MOTOR_LEN   27              /* maximum allowed length of a motor name*/

#define hiw(j)          ( (j)/65536 )
#define low(j)          ( (j)%65536 )

int32 sdstring(int32 sfid, char *sd_name,char *string);
void entry_log(int entry, int32 vgN, int32 vgD, int id, int Did,
   int32 sid[], int32 Dsid[]);

void main(void)
{
   int32        dim1[1]={1};
   int32        edge1[1]={1};
   int32        dims[10],edge[10];
   int32        start0[]={0,0,0,0,0,0,0,0,0};   /* max possible rank defined at start*/
   int32        all_ones = -1;                   /* used to flag non-existant data */
   int          j, i, ii, iii;
   int16        i16;
   float32      x;
   int16        one16 = 1;                       /* a 16 bit one, used with DFNT_INT16 */
   char         equation[255];
   char         *file_name = "examples.hdf";
   char         *user_name = "John Q. Public";
   char         *user_mail = "Chemistry Dept.\nBerkeley U.\n CA  11111";
   char         *user_email = "joe@pdp8.chem.mit.edu";
   char         *user_phone = "900-555-1212";
   char         *user_fax = "900-555-1213";
```

```
    int32       sfid,vfid;                        /* HDF column identifiers */
    int32       vgN;                              /* HDF group identifier for entry */
    int32       vgD;                              /* HDF group identifier for data1 */
    int32       sid[MAX_ID],Dsid[MAX_ID];
    int         id,Did;

    int32       dim_id;
    float32     CCD_x_axis[64],CCD_y_axis[64];
    int16       ccd_data[11][64][64];

    float32     gain, gain1, gain2;
    float32     lo,hi,absorption_corr,deadtime;
    float32     hutch_pressure,hutch_temperature,temperature;
    float32     hkl_near[3];

    float32     mono_energy[501];
    float32     theta[51];
    float32     ttheta[51];
    float32     chi[51];
    float32     phi[51];
    float32     h1[360], k1[360], l1[360];
    int32       ic1[501];
    int32       ic2[501];
    int32       scint[360];
    float32     computed[501];

    float32     h1a[21][21], k1a[21][21], l1a[21][21];
    int32       ic1a[21][21];
    float32     computeda[21][21];
  /* open the file for Vset and SD */
    printf("opening file\n");
    sfid = SDstart(file_name, DFACC_CREATE);
    vfid = Hopen(file_name,DFACC_RDWR,0);
    Vstart(vfid);
  /* write global attributes to the file */
    printf("writing global attributes\n");
    i = SDsetattr(sfid,"file_name",DFNT_CHAR8,strlen(file_name),file_name);
    i = SDsetattr(sfid,"version",DFNT_CHAR8,strlen(VERSION),VERSION);
    i16 = MAX_SDS_LEN;
    i = SDsetattr(sfid,"MAX_SDS_LEN",DFNT_INT16,1,(VOIDP) &i16);
    i16 = MAX_MOTOR_LEN;
    i = SDsetattr(sfid,"MAX_MOTOR_LEN",DFNT_INT16,1,(VOIDP) &i16);
```

/* **entry_default**, The default entry. */

```
    printf("\n\nstarting 'entry_default'\n");
    vgN = Vattach(vfid,-1,"w");
    Vsetname(vgN,"entry_default");              /* this name is required */
    Vsetclass(vgN,"APS_default");               /* this class is required */
    id = -1;

    sid[++id] = sdstring(sfid,"location",LOCATION);
    sid[++id] = sdstring(sfid,"program_name",PROG_NAM);
    i = SDsetattr(sid[id],"version",DFNT_CHAR8,3,"2.3");
    sid[++id] = sdstring(sfid,"user_name",user_name);
    i = SDsetattr(sid[id],"user_mail",DFNT_CHAR8,strlen(user_mail),user_mail);
    i=SDsetattr(sid[id],"user_email",DFNT_CHAR8,strlen(user_email),user_email);
```

```
        i=SDsetattr(sid[id],"user_phone",DFNT_CHAR8,strlen(user_phone),user_phone);
        i = SDsetattr(sid[id],"user_fax",DFNT_CHAR8,strlen(user_fax),user_fax);
        sid[++id] = SDcreate(sfid,"h",DFNT_FLOAT32,1,dim1);          /* default, sets h1 */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,2,"h1");
        sid[++id] = SDcreate(sfid,"k",DFNT_FLOAT32,1,dim1);          /*default, sets k1 */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,2,"k1");
        sid[++id] = SDcreate(sfid,"l",DFNT_FLOAT32,1,dim1);          /*default, sets l1 */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,2,"l1");
        sid[++id] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dim1);      /*default, sets theta*/
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,6,"theta");
        sid[++id] = SDcreate(sfid,"2theta",DFNT_FLOAT32,1,dim1);/*default, set 2theta */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,7,"2theta");
        sid[++id] = SDcreate(sfid,"chi",DFNT_FLOAT32,1,dim1);        /*default, sets chi */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,3,"chi");
        sid[++id] = SDcreate(sfid,"phi",DFNT_FLOAT32,1,dim1);        /*default, sets phi */
        SDsetattr(sid[id],"diffract_axis",DFNT_CHAR8,3,"phi");

        entry_log(-1, vgN, vgD, id, (int32) -1, sid, Dsid);
        for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
        Vdetach(vgN);

/* entry1, The maximally minimal entry. */
        printf("\n\nstarting 'entry1'\n");
        vgN = Vattach(vfid,-1,"w");
        Vsetname(vgN,"entry1");
        Vsetclass(vgN,"APS_entry");
        id = -1;

        sid[++id] = sdstring(sfid,"date","22-feb-1996");
        sid[++id] = sdstring(sfid,"hour","23:59:59.00 UT");
        sid[++id] = sdstring(sfid,"entry_analysis","NONE");
        sid[++id] = sdstring(sfid,"entry_intent","misc");

        entry_log(1, vgN, vgD, id, (int32) -1, sid, Dsid);
        for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
        Vdetach(vgN);
  /* entry2, A simple EXAFS scan with log ratio stored in data file. */
        printf("\n\nstarting 'entry2'\n");
        for (i=0;i<501;i++) {                          /* make a dummy scan of 501 points */
            mono_energy[i] = 20. + .001 * i;
            ic1[i] = 200000;
            ic2[i] = 100000 + 2*i;
        }
        edge[0]=dims[0]=501;                           /* scan of 501 points, rank=1 */
        gain1 = gain2 = 1.e8;
        vgN = Vattach(vfid,-1,"w");
        Vsetname(vgN,"entry2");
        Vsetclass(vgN,"APS_entry");
        id = -1;

        sid[++id] = sdstring(sfid,"date","23-feb-1996");
        sid[++id] = sdstring(sfid,"hour","00:10:59.10 -7");
        sid[++id] = sdstring(sfid,"entry_analysis","EXAFS");
        sid[++id] = sdstring(sfid,"entry_intent","data");
        sid[++id] = sdstring(sfid,"title","Mo K-edge");
        vgD = Vattach(vfid,-1,"w");
        Vsetname(vgD,"data1");
        Vsetclass(vgD,"APS_scan");
        Did = -1;

        Dsid[++Did] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"energy","keV","%.5","");
        SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &one16);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) mono_energy);

        Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
        SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain1);
        SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
        SDsetdatastrs(Dsid[Did],"monitor","photons","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic1);

        Dsid[++Did] = SDcreate(sfid,"ic2",DFNT_INT32,1,dims);
        SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain2);
        SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,(VOIDP) &one16);
        SDsetdatastrs(Dsid[Did],"detector","photons","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic2);

        for (i=0;i<dims[0];i++) { computed[i] = log(ic2[i]/ic1[i]); }
        Dsid[++Did] = SDcreate(sfid,"corrected_signal",DFNT_FLOAT32,1,dims);
        strcpy(equation,"log(ic2/ic1)");
        SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
        SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,(VOIDP) &one16);
        SDsetdatastrs(Dsid[Did],"log(ic2/ic1)","","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) computed);

        entry_log(2, vgN, vgD, id, Did, sid, Dsid);
        for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
        Vdetach(vgD);
        for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
        Vaddtagref(vgN,DFTAG_VG,vgD);
        Vdetach(vgN);
```

/* **entry3**, A theta scan of 10 intervals with one counter and one CCD image saved at each point (64 x 64). */

```
        printf("\n\nstarting 'entry3'\n");
        for (i=0;i<11;i++) {              /* make a dummy scan of 11 points */
              theta[i] = 10.+ i;
              ic1[i] = 200000;
              scint[i] = 250 + i;
              for (ii=0;ii<64;ii++) {
                   for (iii=0;iii<64;iii++) ccd_data[i][ii][iii] = 1000*i+100*ii+iii;
              }
        }
        edge[0]=dims[0]=11;
        edge[1]=edge[2]=dims[1]=dims[2]=64;
        vgN = Vattach(vfid,-1,"w");
        Vsetname(vgN,"entry3");
        Vsetclass(vgN,"APS_entry");
        id = -1;
```

```
sid[++id] = sdstring(sfid,"date","23-feb-1996");
sid[++id] = sdstring(sfid,"hour","02:10:59.10 -7");
sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
sid[++id] = sdstring(sfid,"entry_intent","data");
sid[++id] = sdstring(sfid,"title","fancy CCD scan");

mono_energy[0] = 8.047;
sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"energy","keV","%.4","");
SDwritedata(sid[id],start0,NULL,dim1,(VOIDP) mono_energy);

vgD = Vattach(vfid,-1,"w");
Vsetname(vgD,"data1");
Vsetclass(vgD,"APS_scan");
Did = -1;
```

/* the SDsetattr for "diffract_axis" is redundant, see entry_default */

```
Dsid[++Did] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"theta","degrees","%.3","");
SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &one16);
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,6,"theta");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) theta);

Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain1);
SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"monitor","","%.5","");
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic1);

Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"flourescence detector","counts","%.0","");
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) scint);

Dsid[++Did] = SDcreate(sfid,"CCD",DFNT_INT16,3,dims);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,(VOIDP) &one16);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,(VOIDP) &one16);
SDsetdatastrs(Dsid[Did],"CCD detector","photons","%.5","");
SDgetdimid(Dsid[Did],2);
i = SDsetdimscale(dim_id,dims[1],DFNT_FLOAT32,(VOIDP) CCD_x_axis);
SDgetdimid(Dsid[Did],3);
i = SDsetdimscale(dim_id,dims[2],DFNT_FLOAT32,(VOIDP) CCD_y_axis);
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ccd_data);
```

/* Note, for a CCD there is more information to store. So, it is best to create another Vgroup called "CCD_spec", and attach it to the Vgroup vgN (not to vgD). In this new Vgroup make numerous simple SDS's, one for each piece of information. The reason for using SDS's is so that you can attach dimensions to the information. For example to store an sid which refers to a dark current exposure, or distance from the CCD to the sample, etc. */

```
entry_log(3, vgN, vgD, id, Did, sid, Dsid);
for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
Vdetach(vgD);
for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vaddtagref(vgN,DFTAG_VG,vgD);
Vdetach(vgN);
```

/* **entry4**, A line in hkl space. The scan direction is along the [111] */

```
printf("\n\nstarting 'entry4'\n");
for (i=0;i<51;i++) {                    /* make a dummy scan of 51 points */
     theta[i] = 10. + .02*i;
     ttheta[i] = 2*theta[i];
     chi[i] = 45.;
     phi[i] = 20.;
     h1[i] = k1[i] = l1[i] = 1. + i*0.002;
     ic1[i]  = 2000000;
     x = theta[i]-10.5;
     scint[i] = 3000. * exp(-(x*x/.1));
}
edge[0]=dims[0]=51;
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry4");
Vsetclass(vgN,"APS_entry");
id = -1;

sid[++id] = sdstring(sfid,"date","25-Feb-1996");
sid[++id] = sdstring(sfid,"hour","02:10:59.10 -7");
sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
sid[++id] = sdstring(sfid,"entry_intent","data");
sid[++id] = sdstring(sfid,"title","along the [111] direction");
sid[++id] = sdstring(sfid,"constraint","delta_h = delta_k = delta_l");

mono_energy[0] = 8.047;
sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"energy","keV","%.4","");
SDwritedata(sid[id],start0,NULL,dim1,(VOIDP) mono_energy);

vgD = Vattach(vfid,-1,"w");
Vsetname(vgD,"data1");
Vsetclass(vgD,"APS_scan");
Did = -1;
```

/* the next seven callso to SDsetattr for "diffract_axis" are redundant, see entry_default above */

```
Dsid[++Did] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"theta","degrees","%.3","");
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,6,"theta");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) theta);

Dsid[++Did] = SDcreate(sfid,"2theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"2theta","degrees","%.3","");
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,7,"2theta");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ttheta);

Dsid[++Did] = SDcreate(sfid,"chi",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"chi","degrees","%.3","");
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,3,"chi");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) chi);

Dsid[++Did] = SDcreate(sfid,"phi",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"phi","degrees","%.3","");
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,3,"phi");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) phi);

Dsid[++Did] = SDcreate(sfid,"h1",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"h","","%.3","");
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,2,"h1");
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) h1);
```

```
        Dsid[++Did] = SDcreate(sfid,"k1",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"k","","%.3","");
        SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,2,"k1");
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) k1);

        Dsid[++Did] = SDcreate(sfid,"l1",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"l","","%.3","");
        SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,2,"l1");
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) l1);

        for (i=0;i<dims[0];i++) { computed[i] =
             sqrt(h1[i]*h1[i]+k1[i]*k1[i]+l1[i]*l1[i]); }
        Dsid[++Did] = SDcreate(sfid,"x-axis",DFNT_FLOAT32,1,dims);
        strcpy(equation,"sqrt(h*h+k*k+l*l)");
        SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
        SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &one16);
        SDsetdatastrs(Dsid[Did],"distance along [111]","","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) computed);

        Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
        SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain1);
        SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
        SDsetdatastrs(Dsid[Did],"monitor","photons","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic1);

        Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
        SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,(VOIDP) &one16);
        SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,(VOIDP) &one16);
        SDsetdatastrs(Dsid[Did],"detector","photons","%.5","");
        SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
        SDsetattr(Dsid[Did],"analyzer",DFNT_CHAR8,5,"Ge111");
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) scint);

        entry_log(4, vgN, vgD, id, Did, sid, Dsid);
        for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
        Vdetach(vgD);
        for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
        Vaddtagref(vgN,DFTAG_VG,vgD);
        Vdetach(vgN);

/* entry5. A surface in hkl space. The scan directions are along the [100] and [011] */
        printf("\n\nstarting 'entry5'\n");
        for (j=0;j<21;j++) {                    /* make a dummy scan of 21 x 21 points */
             for (i=0;i<21;i++) {
                  h1a[j][i] = 1. + .1*j;
                  k1a[j][i] = l1a[j][i] = 1. + .05*i;
                  ic1a[j][i] = 20000;
                  x = (i-10)*(i-10) /.3 + (j-10)*(j-10)/.5;
             scint[i] = 1000. * exp(-x);
             }
        }
        edge[0]=edge[1]=dims[0]=dims[1]=21;
        vgN = Vattach(vfid,-1,"w");
        Vsetname(vgN,"entry5");
        Vsetclass(vgN,"APS_entry");
        id = -1;
```

```
sid[++id] = sdstring(sfid,"date","25-Feb-1996");
sid[++id] = sdstring(sfid,"hour","03:10:59.10 -7");
sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
sid[++id] = sdstring(sfid,"entry_intent","data");
sid[++id] = sdstring(sfid,"title","the perpendicular surface");
sid[++id] = sdstring(sfid,"constraint","perpendicular to (0 -1 1)");

mono_energy[0] = 8.047;
sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"energy","keV","%.4","");
SDwritedata(sid[id],start0,NULL,dim1,(VOIDP) mono_energy);

vgD = Vattach(vfid,-1,"w");
Vsetname(vgD,"data1");
Vsetclass(vgD,"APS_scan");
Did = -1;

Dsid[++Did] = SDcreate(sfid,"h1",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"h","","%.3",""); /* diffract_axis attribute set in entry_default */
SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &one16);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) h1a);

Dsid[++Did] = SDcreate(sfid,"k1",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"k","","%.3",""); /* diffract_axis attribute set in entry_default */
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) k1a);

Dsid[++Did] = SDcreate(sfid,"l1",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"l","","%.3",""); /* diffract_axis attribute set in entry_default */
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) l1a);

for (j=0;j<dims[0];j++) {
      for (i=0;i<dims[1];i++) {
          computeda[j][i] = sqrt(k1a[j][i]*k1a[j][i]+l1a[j][i]*l1a[j][i]);
      }
}
Dsid[++Did] = SDcreate(sfid,"y-axis",DFNT_FLOAT32,1,dims);
strcpy(equation,"sqrt(k*k+l*l)");
SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
i16 = 2;
SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &i16);
SDsetdatastrs(Dsid[Did],"distance along [011]","","%.3","");
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) computeda);

Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain1);
SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"monitor","photons","%.5","");
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic1a);

Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,(VOIDP) &one16);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,(VOIDP) &one16);
SDsetdatastrs(Dsid[Did],"detector","photons","%.5","");
SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) scint);
```

```
        entry_log(5, vgN, vgD, id, Did, sid, Dsid);
        for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
        Vdetach(vgD);
        for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
        Vaddtagref(vgN,DFTAG_VG,vgD);
        Vdetach(vgN);
```

/* **entry6**, A circle in hkl space, with points every degree. */

```
        printf("\n\nstarting 'entry6'\n");
        for (i=0;i<360;i++) {                    /* make a dummy scan of 360 points */
              x = i * PI/180.;
              h1[i] = .1 * cos(x);
              k1[i] = .1 * sin(x);
              l1[i] = 2.;
              ic1[i] = 200000;
              scint[i] = pow(sin(x),10.);
        }
        edge[0]=dims[0]=360;
        vgN = Vattach(vfid,-1,"w");
        Vsetname(vgN,"entry6");
        Vsetclass(vgN,"APS_entry");
        id = -1;
        sid[++id] = sdstring(sfid,"date","25-Feb-1996");
        sid[++id] = sdstring(sfid,"hour","02:10:59.10 -7");
        sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
        sid[++id] = sdstring(sfid,"entry_intent","data");
        sid[++id] = sdstring(sfid,"title","0.1 rad circle about (002)");
        sid[++id] = sdstring(sfid,"constraint",
                        "|(hkl)-(002)|=.1 and (hkl)-(002) perpendicular to (002)");

        mono_energy[0] = 8.047;
        sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
        SDsetdatastrs(sid[id],"energy","keV","%.4","");
        SDwritedata(sid[id],start0,NULL,dim1,(VOIDP) mono_energy);

        hkl_near[0]=0;  hkl_near[1]=0;  hkl_near[2]=2;
        edge[0]=dims[0]=3;
        sid[++id] = SDcreate(sfid,"hkl_near",DFNT_FLOAT32,1,dims);
        SDwritedata(sid[id],start0,NULL,edge,(VOIDP) hkl_near);

        vgD = Vattach(vfid,-1,"w");
        Vsetname(vgD,"data1");
        Vsetclass(vgD,"APS_scan");
        Did = -1;
        edge[0]=dims[0]=360;

        Dsid[++Did] = SDcreate(sfid,"h1",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"h","","%.3","");     /* diffract_axsi set to h1 in entry_default */
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) h1);

        Dsid[++Did] = SDcreate(sfid,"k1",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"k","","%.3","");     /* diffract_axsi set to k1 in entry_default */
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) k1);

        Dsid[++Did] = SDcreate(sfid,"l1",DFNT_FLOAT32,1,dims);
        SDsetdatastrs(Dsid[Did],"l","","%.3","");     /* diffract_axsi set to l1 in entry_default */
        SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) l1);
```

```
    for (i=0;i<dims[0];i++) { computed[i] = i; }
    Dsid[++Did] = SDcreate(sfid,"circle",DFNT_FLOAT32,1,dims);
    strcpy(equation,"angle");
    SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
    SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,(VOIDP) &one16);
    SDsetdatastrs(Dsid[Did],"angle on circle","degrees","%.1","");
    SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
    SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) computed);

    Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
    SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,(VOIDP) &gain1);
    SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
    SDsetdatastrs(Dsid[Did],"monitor","photons","%.5","");
    SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
    SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) ic1);

    Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
    SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,(VOIDP) &one16);
    SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,(VOIDP) &one16);
    SDsetdatastrs(Dsid[Did],"detector","photons","%.5","");
    SDsetfillvalue(Dsid[Did],(VOIDP) &all_ones);
    SDwritedata(Dsid[Did],start0,NULL,edge,(VOIDP) scint);

    entry_log(6, vgN, vgD, id, Did, sid, Dsid);
    for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
    Vdetach(vgD);
    for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
    Vaddtagref(vgN,DFTAG_VG,vgD);
    Vdetach(vgN);

    printf("\n\ncalling end routines and closing the file\n");
    SDend(sfid);
    Vend(vfid);
    Hclose(vfid);
    putchar(BELL);
}
```

```
int32 sdstring(int32 sfid, char *sd_name,char *string)
```
/* this routine writes an SDS which only contins a single char string */
```
{
    int32        sid;
    int32        dim;
    int32        start=0;

    dim = strlen(string);
    sid = SDcreate(sfid,sd_name,DFNT_CHAR8,1,&dim);
    SDwritedata(sid,&start,NULL,&dim,string);
    return sid;
}
```

/* this routine is only for debugging purposes, in actual use, it would not be here. */
```
void entry_log(
int         entry,
int32   vgN,
int32   vgD,
int         id,
int         Did,
int32   sid[],
int32   Dsid[])
{
    int      i;
    char str[60];
    int32    ref;

    if (entry>=1) sprintf(str,"entry%d",entry);
    else          strcpy(str,"entry_default");

    printf("   The Vgroup '%s' is id = %ld:%ld, and Vgroup 'data1' is id =
%ld:%ld\n",str,hiw(vgN),low(vgN),hiw(vgD),low(vgD));
    printf("   The SDS's in '%s' are:\n",str);
    for (i=0;i<=id;i++) {
        ref = SDidtoref(sid[i]);
        printf("    %ld  =  %ld:%ld\n",sid[i],hiw(ref),low(ref));
        }

    if (Did<0) printf("\n   There is no 'data1' for %s\n",str);
    else       printf("\n   The SDS's in 'data1' of %s are:\n",str);
    for (i=0;i<=Did;i++) {
        ref =  SDidtoref(Dsid[i]);
        printf("    %ld  =  %ld:%ld\n",Dsid[i],hiw(ref),low(ref)); }

    return;
}
```