Details of the GSAS2CIF program

Please note, the software documented on these web pages is slightly out of date with respect to the current GSAS distribution. The web pages will be updated soon.

Template files

As will be discussed further, below, the GSAS2CIF program utilizes three CIF template files:

template_publ.cif
: A template file with publication information and descriptive information about how a refinement was performed.

template_phase.cif
: A template file with descriptive information about a chemical phase.

template_instrument.cif
: A template file with descriptive information about a diffraction instrument used for powder diffraction data collection.

Initialization

The standard GSAS header file, DISAGLCM.FOR, is used to define the PARNAMS array, needed later for the call to RDCOVAR. The standard GSAS header file, ARRAYSZE.FOR, defines the array sizes used in DISAGLCM.FOR. The first two subroutine calls (STRTRN & PROGNAM) in GSAS2CIF are the same as are found in every GSAS program

pdCIFs have a unique feature, a block id, which is used to make references between blocks. The block id is intended to be a unique string that will never appear in any other CIF, so for this reason, it is typically composed of several items, including:

- date & time,
- author name,
- instrument name &
- project name.

The next step in the code sets variable EXPRNAME to the name based on the GSAS experiment file. This is used as the project name for both the block_id and the data block name. Note that EXPRNAME variable is restricted to 20 characters. Subroutine VSTRNG is used to make sure that the project name is valid (printable ASCII characters, no spaces & no vertical bars ("|")]. Subroutine LENCH is used to determine the length of an ASCII string.

The file for CIF output is then opened.

The date & time is obtained in the CIF format using the GSAS routine GSDATE. Note that this code is somewhat compiler-specific.

The author name is read from the GSAS experiment file. If it is not present, it is requested from the user and is then saved in the experiment file. The name is saved in two forms, AUTHOR, as entered and SAUTHOR, without spaces & special characters, for use in block_id's.

At this stage the number of histograms and phases are counted and several flags are set:

IFPWDR
: True if one or more powder histograms are present

IFSNGL
: True if one or more single crystal histograms are present

NUMPHAS
: Number of phases

NPWDHIST
: Number of powder histograms

ONEBLOCK
: True if the CIF can be a single block -- one phase and one histogram

For each histogram a check is made to see if a name exists for the instrument and if not the user is requested to input the name. This name is used in the block_id for data blocks. Ideally this instrument name is read from the instrument parameter file associated with the original raw data for the histogram.

To generate uncertainties on coordinates, the variance-covariance matrix is read, written by program GENLES, using the standard GSAS routine RDCOVAR. If the .CMT file, which contains this information, is out of date, noted because it does not match the cycle number in the current experiment file, RDCOVAR generates a warning message and sets variable NUMPAR to zero. If this happens, or the file cannot be read, then the user is consulted to see if the program should continue, as uncertainies may not be needed when a CIF will be created for export of coordinates to a plotting programs, but will be needed to fully document a result for publication.

The cycle number and the most recent sum of squares of differences

(SUMDSQ1) is read from the .EXP file. A file of interatomic distances and angles (.DIS file), written by program DISAGL is opened. Both the cycle number and sum of squares of differences must match the file contents. If the file does not match or cannot be read, as before the user must choose between exiting and continuing.

Overall CIF Information

The writing to the CIF file then starts. The first (or only, if ONEBLOCK is true) data name is created, from variable EXPRNAME (which is restricted to 20 characters or less to avoid too large data names). Then a block_id is created and written. In the single-block (ONEBLOCK is true) case, the block_id includes the instrument name. In the multi-block case, the first block will have information relevant to all blocks, but the histogram(s) and phase(s) will be in separate blocks. Then a few audit records are created. In the cases where it is unclear if quotes will be needed, subroutine WRVAL is used. This in turn calls subroutine ADDQUOTE to check the string to be written and to add quotes as needed.

The publication information template is then copied using subroutine CPTMPLTE, as is described further below. The overall template is read from file *EXPNAM*_publ.cif, where *EXPNAM*.EXP is the name of the GSAS experiment file. If file *EXPNAM*_publ.cif does not exist, then then it is created using the contents of file template_publ.cif, which is read from the current directory, or if not present, from the distribution version in the GSAS data directory. The template contents is also copied directly into the the output CIF file as well.

Results that pertain to the overall refinement are then written using subroutine OVERALL. Subroutine OVERALL creates CIF entries that describe how the refinement has progressed. For example, _refine_ls_shift/su_max describes the maximum parameter shift in the last cycle of refinement. Powder profile R-factors are written later, when the powder diffraction histograms are written, in subroutine WRPOWDHIST, but if more than one powder histogram is computed, the GSAS also computes overall powder R-factors, for all histograms combined and these overall R-factor values are written out here. Note that in thise case, a multi-block CIF will be created.

Phase Information

GSAS2CIF then loops over phases. Note that if there is is more than one phase, the information for each phase must be placed in a separate block. This is also true if more than one histogram present. Thus, if more than one phase or more than one histogram is present (or both), then the phase information and histogram information will be in separate blocks. However, in the case where there is one histogram and one phase, then variable ONEBLOCK is set to true and both the phase information and histogram information will be included in the same CIF block. This is why a data block is started and a block_id created in this loop only for multiblock CIFs.

The phase information template is then copied using subroutine CPTMPLTE (described further below). This template is read from file *EXPNAM*_phase*N*.cif, where *EXPNAM*.EXP is the name of the GSAS experiment file and *N* is the phase number. If file *EXPNAM*_phase*N* does not exist, then then it is created using the contents of file template_phase.cif, which is read from the current directory, or if not present, from the distribution version in the GSAS data directory. The template contents is also copied directly into the the output CIF file as well.

The next step is to write out the phase information. This is done using subroutine WRITEPHASE, discussed further below.

Histogram Information

After phase processing is complete, then processing of histograms starts. First, the instrument name, input earlier is read from the .EXP file. The program then processes powder diffraction histograms differently from single crystal histograms.

Powder Histograms

The first step in processing powder diffraction histograms is to begin a data block and create a block_id, unless a single block CIF is being created.

The next step is to insert the histogram template file. This is done by creating two file names, *EXPNAM_instnameNN*.cif and *instname*.cif, where *instname* is the instrument name that was input before. Subroutine CPTMPLTE (see below) first attempts to read from file *EXPNAM_instnameNN*.cif in the current directory. If this file is not found, it is created and filled with the more generic histogram template file. If the *EXPNAM_instnameNN*.cif file is not found, subroutine CPTMPLTE attempts to read file *instname*.cif first from the current directory, or if not present, from the GSAS data directory. The *instname*.cif file is intended as a template file that has been customized for a particular instrument. If this file cannot be found, then file template_instrument.cif is read from the current directory, or if not present there, from the distribution version of this file in the GSAS data directory.

Parameters and powder data are written in subroutine WRPOWDHIST, as is described further below

Finally, the reflections are listed using subroutine WRREFLIST.

Single-Crystal Histograms

For single crystal histograms, the only output that is generated is that the reflections are listed using subroutine WRREFLIST.

Copying of Template Files

Subroutine CPTMPLTE is used to copy a template file into a CIF. The strategy is that descriptive information to be included in the output CIF will be placed in a set of project-specific templates files, rather than added directly to the CIF. In this way, GSAS2CIF can be rerun at any point and the descriptive information will be included in the output CIF. The project-specific template files are named similarly to the GSAS experiment file. If any of these project-specific template files are not found, they are created using either customized template files or if not found using a standard version distributed with GSAS. This allows a user to reuse customizes CIF template files, so that, for example, the instrument description can be reused.

Subroutine CPTMPLTE first attempts to read a version of the template file that has been customized for the current project from the current directory. The name of this file is passed to CPTMPLTE in variable LOCALCOPY. If this file is not found, a second file name, found in variable TEMPLATE1, is tried (if this name is non-blank). The subroutine looks first in the current directory and if not there, in the GSAS data directory, which is determined by an environment variable (gsas). If the TEMPLATE1 file is opened, the file LOCALCOPY is created and opened for output. If neither the LOCALCOPY nor the TEMPLATE1 file is found, a third file name, found in variable TEMPLATE2 is opened. The subroutine looks first in the current directory and if not there, in the GSAS data directory, which is determined by an environment variable (gsas). If this file is not found, the program stops, as this implies that the environment variable or required files are not properly installed. If the TEMPLATE2 file is opened, the file LOCALCOPY is created and opened for output.

After either file LOCALCOPY, TEMPLATE1 or TEMPLATE2 is opened, it is copied one line at a time. All lines are copied to the LOCALCOPY file, if TEMPLATE1 or TEMPLATE2 is being read. Each line is checked for a string starting with "data_", lines following the data flag are copied into the output CIF. The template file should not have any lines greater than 80 characters, so if any are noted, a warning message is produced.

Subroutine WRITEPHA

Subroutine WRITEPHA is used to write information about a phase into the CIF output. This information includes the unit cell parameters, symmetry, atomic parameters and refinement parameters that are phase-specific.

The first step in subroutine WRITEPHA is to call the standard GSAS routine DSGREAD, which reads in the coordinates and their uncertainties, as well as unit cell parameters and symmetry information. Note that much of is read into common blocks.

The GSAS phase name is read from the .EXP file and is written out as CIF item _pd_phase_name. Unit cell parameters are then read. from the .EXP file. GSAS subroutine BMATRX is used to compute the reciprocal unit cell parameters for later use. The unit cell parameters are then written out, where only the unique parameters (*i.e. a & c* for a tetragonal cell are given with uncertainties. The unit cell volume is computed (alas, without an uncertainty estimate at present) using GSAS subroutine CELVOL and the unit cell type is written by translating the Laue class.

The space group is written in exactly the same format as used by GSAS, except that the trailing "R" flag, which is used by GSAS to indicate a rhombohedral setting, is removed if present.

The symmetry operations are then written from the matrices generated by subroutine DSGREAD. This requires a bit of extra work, as GSAS does not generate the symmetry operations corresponding to a center of symmetry or lattice centering, if present. Note that offsets applied to symmetry operations to bring them into agreement with the *International Tables*, for example, after a -x,1/2+y,-z is operated on by body center +1/2,+1/2,+1/2, the resulting symmetry operation, 1/2-x,1+y,1/2-z, is conventionally written 1/2-x,y,1/2-z. The offsets applied to symmetry operations are saved in array OFFSET, for later use with interatomic distance and angle listings.

Each symmetry operation is assigned a code (_symmetry_equiv_pos_site_id) which is later referenced in the interatomic distance and angle listings. This corresponds to the GSAS symmetry element number, plus 100 times the centering operation number and multiplied by -1, for elements generated by a center of symmetry. Note that centric space groups in GSAS always have their origin at the center of symmetry (Origin 2, where a choice is offered). So, the center of symmetry operation is always -x,-y,-z.

Atoms are then processed. First, counters used for unit cell contents are zeroed. Then the atom table loop headers are written, and the atom labels are checked, to make sure that all atom labels are unique, since this is required by the CIF standard. It would be confusing if GSAS2CIF changed atom labels, so if any atom have the same labels, a warning message is generated. Users are given the option to produce a CIF that contains duplicate atom labels since few, if any programs that read CIFs will even notice.

The atom table is generated. GSAS subroutine SYTSYM is used to compute the site multiplicity for each atom. The composition of the unit cell is then noted using arrays COMPTBL and FRACTBL, where FRACTBL is used for atoms that have partial occupancy. Note that if no atoms are written a series of "?" values are written to match the table header. If any atoms with anisotropic displacement parameters are noted, a second atom table is generated with the anisotropic U_{ij} values.

The number of atoms of each type are then listed. Note that due to categorization rules, these numbers of atoms and the scattering factor values can only appear in the same loop, if in the same block. Thus, if a single block CIF will be created, this loop is skipped and these numbers are reported in subroutine WRPOWDHIST. information must appear this loop must be combined with the

A value for Z is determined (_cell_formula_units_Z) by dividing the unit composition for all of the fully occupied atoms by 2 and 3 as many times as is possible, without resulting in non-integer values. The chemical formula (_chemical_formula_sum) and the mass (not weight!) of a formula unit (_chemical_formula_weight) are then computed by dividing the total values for a unit cell using the value of Z. Note that the determination of Z is sometimes a matter of style and on occasion users may decide to edit the resulting CIF file to change Z. If done, be sure to change _chemical_formula_sum and _chemical_formula_weight accordingly.

GSAS offers two types of preferred orientation corrections, the traditional March-Dollase correction and a spherical harmonic expansion representation of the orientation distribution function. The March-Dollase terms are set by histogram and phase, while each phase has a single set of spherical harmonic terms for all histograms. In the case where a multiblock CIF is being written, the spherical harmonic terms are written in subroutine WRITEPHASE. In the single block case, the these terms are written in WRPOWDHIST.

Interatomic distances are then written. This is done by reading through the .DIS file and then writing out distances matching the current phase. Note that each pair of atoms has a code that identifies the symmetry operations needed to generate the site from the coordinates in the list. These codes are written by program DISAGL into the .DIS file, but must be corrected with the offsets generated previously. Note that no operations are applied to the first atom, so that its site code is always ".".

Interatomic angles are then written. This is done as before by reading through the .DIS file and then writing out angles matching the current phase. Note that the no operations are applied to the central atom, so that its site code is always ".", but the two outer atoms each have a code that identifies the symmetry operations needed to generate the sites from the coordinates in the list. These codes are written by program DISAGL into the .DIS file, but must be also corrected with the offsets generated previously.

Subroutine WRPOWDHIST

Subroutine WRPOWDHIST is used to write histogram-related information into the output CIF. This information includes the powder data, as well the computed pattern, as well as the many parameters used within GSAS in order to reproduce the experimental data.

Subroutine WRPOWDHIST starts by counting the number of phases present in the histogram and by calling GSAS subroutine OPNPRF, which opens the binary file containing the observed and computed pattern.

In preparation for writing the preferred orientation parameters, the number of March-Dollase & (when needed) spherical harmonic terms are counted. The March-Dollase terms are stored as IMD and the spherical harmonic terms are stored as IODF. The treatment of these preferred orientation parameters is a bit complex, since there are *n* x *m* March-Dollase terms, but only *n* spherical harmonic terms, where there are *n* phases and *m* histograms. If a multiblock CIF will be created, the spherical harmonic terms are included in the phase data block(s), while the March-Dollase terms are included in the histogram data block(s). CIF only defines one term for recording the preferred orientation correction, so in the single-block case, care is taken to make sure that both sets of terms are output together, should both ever be used together. It makes little sense for both types of corrections to be used together, but the goal is that the CIF should reflect how the refinement was performed.

For a multi-block CIF, a phase table is written as the first information recorded in the CIF by WRPOWDHIST for multi-block CIFs. Some of the items contained in the phase table are:

_pd_phase_block_id
: a pointer to the block that defines the phase

_pd_phase_mass_%
: the percentage of the current phase

_pd_proc_ls_profile_function
: the profile function and terms, described as a text item. Much of the text is generated in subroutine LISTPRF.

_pd_proc_ls_pref_orient_corr
: the March-Dollase correction, when needed.

Alternately, in the single-block case, the unit cell contents are determined, so that unit cell contents can be included with the scattering factors. A table of atoms, with scattering factors or scattering lengths is then written, optionally with the unit cell contents, is then written.

The next section writes information about the probe species: x-ray vs. neutron, wavelength(s), polarization & other calibration information. Note that in the case where two wavelengths are present, these values must be placed in a loop and are labeled with _diffrn_radiation_type. This creates a violation of the CIF categorization rules, as the category of _diffrn_radiation_type differs from the _diffrn_radiation_wavelength data items. Alas, there is no other way at present to solve this.

Subsequent sections of subroutine WRPOWDHIST write out different types of histogram information. R-factors are read from the .EXP file and are written to the CIF. Background terms are then written. These terms are written as a text field, as there are no formal definitions for expressing these values yet. Absorption corrections are then written, again as a text field. Then, the maximum and minimum extinction & absorption corrections are written. At present, CIF does not define such terms for extinction, so _gsas_exptl_extinct_corr_T_min and _gsas_exptl_extinct_corr_T_max are used.

While the preferred orientation correction was written in previously described sections, of subroutine WRITEPHASE and WRPOWDHIST for the case of multi-block CIFs, it has not been processed in the case of single-block CIFs. Preferred orientation corrections are written for

March-Dollase and/or spherical harmonic terms Likewise, profile terms were written in previously for the case of multi-block CIFs, profile terms for a single block CIF are written here.

The file is then given a time-stamp and the calculation method is defined as the Rietveld method. There is no particular reason to do this here or anywhere else in this subroutine.

Listing of Powder Data

In the final section of subroutine WRPOWDHIST, the observed and computed data are written. These data are written first to a scratch file and are then read back and are written to the CIF. In this way the numbers can be aligned in columns. There is no requirement within CIF to do this, but it looks nice and makes the numbers much easier to peruse.

However, before the data can be read, a number of flags are set to determine how the data will be stored. The pdCIF dictionary defines two different two-theta data items, one for data with a fixed step size and the other for variable step sizes. In the case of constant wavelength the two-theta values are checked if they are in constant steps. Note that the data are retrieved from the binary histogram file using subroutine READPRF. Also a flag, FIXEDBKG, is set if fixed background points are used to define the background for the pattern.

In certain circumstances, GSAS does not include all the observed diffraction data in the binary histogram file. This can happen when data points are skipped or are averaged. This also happens when sections of the observed histogram at the beginning or end of the pattern are not used. At present, this latter condition is not tested. However, when data points are skipped or are averaged, the initial unprocessed histogram is written in a separate loop from the processed observed and computed patterns. The unprocessed histogram data is written by subroutine WRITERAWDATA. Note that this subroutine uses the _pd_meas_ data items, so if this routine is called, noted by variable MOREOBS set to true, the later part of WRPOWDHIST uses the _pd_proc_ data items.

If the x-axis corresponds to two-theta values in in constant steps, the starting, ending and step values are written to the CIF as _pd_meas_2theta_range_ CIF items, unless variable MOREOBS is true, in which case _pd_proc_2theta_range_ items are used. In the latter case, zero corrections are applied to the values.

Depending on settings previously determined, different options are used to write the header for the observed and computed pattern. Then the data are written, using care to only write the items matching the header entries to the scratch file. Finally, the values are read from the scratch file and are written to the CIF.

Subroutine FESD

Subroutine FESD is used to format numbers for CIF in a variation of crystallographic notation. Note that if the uncertainty value is negative, the uncertainty is not printed, but rather, the uncertainty determines the number of significant digits. This routine does not currently handle numbers in exponential notation.

Subroutine LISTPRF

Subroutine LISTPRF is used to describe the current peak profile function and list some of the profile parameter values.

Subroutine WRITERAWDATA

Subroutine WRITERAWDATA is used to copy the contents of a GSAS raw data file (sometimes named .RAW or .GSAS) directly to a CIF file. Data are read using GSAS subroutine READHST. If the uncertain values match the square root of the intensity values at each point, then it is assumed the intensity values are counts so that uncertanties are not specified. Note that if points are two-theta values and are in constant steps, the _pd_meas_2theta_range CIF items are used in place of _pd_meas_2theta_scan.

Subroutine WRREFLIST

Subroutine WRREFLIST is used to write a table of reflections for both single crystal and powder histograms. The first step in WRREFLIST is to determine if the reflection table corresponds to a powder or single-crystal histogram. This makes a difference when writing the reflection loop header, as a wavelength id, _pd_refln_wavelength_id, is written for multichromatic powder diffraction data, a phase id, _pd_refln_phase_id, is written when more than one phase is present, as well as the reflection d-space, _refln_d_spacing, and a local data item, that defines the relative reflection intensity, _gsas_i100_meas.

The reflection values are written on a scratch file, so that the data can be written into the CIF in nice neat columns. Again, this is not needed, but makes the data more easily read by humans.

Comments, corrections or questions: crystal@NIST.gov

$Revision: $ $Date: $

Publication Information Template: File template_publ.cif

This file is used to insert information relevant to a sample or entire project into the first section or block of a CIF. See the gsas2cif documentation for an explanation of how this is used.

```
# GSAS publication template file

data_Publication_Template
#==============================================================================
# this information describes the project, paper etc. for the CIF             #
# Acta Cryst. Section C papers and editorial correspondence is generated     #
# from the information in this section                                       #
#                                                                            #
#   (from)   CIF submission form for Rietveld refinements (Acta Cryst. C)    #
#                                                 Version 14 December 1998   #
#==============================================================================
# 1. SUBMISSION DETAILS

_publ_contact_author_name            ?   # Name of author for correspondence
_publ_contact_author_address             # Address of author for correspondence
; ?
;
_publ_contact_author_email           ?
_publ_contact_author_fax             ?
_publ_contact_author_phone           ?

_publ_contact_letter
; ?
;

_publ_requested_journal              ?
_publ_requested_coeditor_name        ?
_publ_requested_category             ?   # Acta C: one of CI/CM/CO/FI/FM/FO

#===============================================================================

# 2. PROCESSING SUMMARY (IUCr Office Use Only)

_journal_data_validation_number      ?

_journal_date_recd_electronic        ?
_journal_date_to_coeditor            ?
_journal_date_from_coeditor          ?
_journal_date_accepted               ?
_journal_date_printers_first         ?
_journal_date_printers_final         ?
_journal_date_proofs_out             ?
_journal_date_proofs_in              ?
_journal_coeditor_name               ?
_journal_coeditor_code               ?
```

```
_journal_coeditor_notes
; ?
;
_journal_techeditor_code              ?
_journal_techeditor_notes
; ?
;
_journal_coden_ASTM                   ?
_journal_name_full                    ?
_journal_year                         ?
_journal_volume                       ?
_journal_issue                        ?
_journal_page_first                   ?
_journal_page_last                    ?
_journal_paper_category               ?
_journal_suppl_publ_number            ?
_journal_suppl_publ_pages             ?

#==============================================================================

# 3. TITLE AND AUTHOR LIST

_publ_section_title
; ?
;
_publ_section_title_footnote
; ?
;

# The loop structure below should contain the names and addresses of all
# authors, in the required order of publication. Repeat as necessary.

loop_
        _publ_author_name
        _publ_author_footnote
        _publ_author_address
 ?                                    #<--'Last name, first name'
; ?
;
; ?
;

#==============================================================================

# 4. TEXT

_publ_section_synopsis
;  ?
;
_publ_section_abstract
; ?
```

```
;
_publ_section_comment
; ?
;
_publ_section_exptl_prep      # Details of the preparation of the sample(s)
                              # should be given here.
; ?
;
_publ_section_exptl_refinement
; ?
;
_publ_section_references
; ?
;
_publ_section_figure_captions
; ?
;
_publ_section_acknowledgements
; ?
;

#==============================================================================
# 5. OVERALL REFINEMENT & COMPUTING DETAILS

_refine_special_details
; ?
;
_pd_proc_ls_special_details
; ?
;

# if regions of the data are excluded, the reason(s) are supplied here:
_pd_proc_info_excluded_regions
; ?
;


# The following items are used to identify the programs used.
_computing_molecular_graphics     ?
_computing_publication_material   ?

_refine_ls_weighting_scheme       ?
_refine_ls_weighting_details      ?
_refine_ls_hydrogen_treatment     ?
_refine_ls_extinction_method      ?
_refine_ls_extinction_coef        ?
_refine_ls_number_constraints     ?

_refine_ls_restrained_S_all       ?
_refine_ls_restrained_S_obs       ?
```

Phase Information Template: File template_phase.cif

This file is used to insert information relevant to a specific chemical phase into the phase section or block(s) of a CIF. See the gsas2cif documentation for an explanation of how this is used.

```
# GSAS phase information template file

data_Phase_Template
#==============================================================================
# 6. PREPARATION, CHEMICAL, STRUCTURAL AND CRYSTAL DATA

# The following three fields describe the preparation of the material.
# The cooling rate is in K/min.  The pressure at which the sample was
# prepared is in kPa.  The temperature of preparation is in K.

_pd_prep_cool_rate                ?
_pd_prep_pressure                 ?
_pd_prep_temperature              ?

_pd_char_particle_morphology      ?
_pd_char_colour                   ?       # use ICDD colour descriptions


_chemical_name_systematic
; ?
;
_chemical_name_common             ?
_chemical_formula_moiety          ?
_chemical_formula_structural      ?
_chemical_formula_analytical      ?
_chemical_melting_point           ?
_chemical_compound_source         ?       # for minerals and
                                          # natural products
_symmetry_space_group_name_Hall   ?

_cell_measurement_temperature     ?

_cell_special_details
; ?
;

_geom_special_details             ?

# The following item identifies the program(s) used (if appropriate).
_computing_structure_solution     ?

#==============================================================================

# 7. Phase information from GSAS
```

Instrument Information Template: File template_instrument.cif

This file is used to insert information relevant to a specific powder diffraction instrument into the dataset section or block(s) of a CIF. See the gsas2cif documentation for an explanation of how this is used.

```
# GSAS instrument template file

data_Instrument_Template
#==============================================================================
# 8. INSTRUMENT CHARACTERIZATION

_exptl_special_details
; ?
;

# The following item is used to identify the equipment used to record
# the powder pattern when the diffractogram was measured at a laboratory
# other than the authors' home institution, e.g. when neutron or synchrotron
# radiation is used.

_pd_instr_location
; ?
;
_pd_calibration_special_details           # description of the method used
                                          # to calibrate the instrument
; ?
;

_diffrn_ambient_temperature       ?
_diffrn_source                    ?
_diffrn_source_target             ?
_diffrn_source_type               ?
_diffrn_measurement_device_type   ?
_diffrn_detector                  ?
_diffrn_detector_type             ?       # make or model of detector

_pd_meas_scan_method              ?       # options are 'step', 'cont',
                                          # 'tof', 'fixed' or
                                          # 'disp' (= dispersive)
_pd_meas_special_details
;  ?
;

# The following two items identify the program(s) used (if appropriate).
_computing_data_collection        ?
_computing_data_reduction         ?

# Describe any processing performed on the data, prior to refinement.
# For example: a manual Lp correction or a precomputed absorption correction
_pd_proc_info_data_reduction      ?

# The following item is used for angular dispersive measurements only.

_diffrn_radiation_monochromator   ?

# The following three items are used for time-of-flight measurements only.

_pd_instr_dist_src/spec           ?
_pd_instr_dist_spec/detc          ?
_pd_meas_2theta_fixed             ?

# 9. Specimen size and mounting information

# The next three fields give the specimen dimensions in mm.  The equatorial
# plane contains the incident and diffracted beam.

_pd_spec_size_axial               ?       # perpendicular to
                                          # equatorial plane

_pd_spec_size_equat               ?       # parallel to
                                          # scattering vector
                                          # in transmission

_pd_spec_size_thick               ?       # parallel to
                                          # scattering vector
                                          # in reflection

_pd_spec_mounting                         # This field should be
                                          # used to give details of the
                                          # container.
; ?
;

_pd_spec_mount_mode               ?       # options are 'reflection'
                                          # or 'transmission'

_pd_spec_shape                    ?       # options are 'cylinder'
                                          # 'flat_sheet' or 'irregular'
```

Program GSAS2CIF

This program is used to create pdCIF files from GSAS Rietveld refinements. See the gsas2cif documentation for an explanation of this code. Dots (●) indicate links to sections in this documentation and underlined subroutines and functions are links to the source code for those routines.

```
      PROGRAM GSAS2CIF

!PURPOSE: Create a CIF file from a GSAS .EXP file

      INCLUDE       '../INCLDS/COPYRIGT.FOR'

!PSEUDOCODE:

!CALLING ARGUMENTS:

●
!INCLUDE STATEMENTS:
      INCLUDE       '../INCLDS/ARRAYSZE.FOR'
      INCLUDE       '../INCLDS/DISAGLCM.FOR'

!LOCAL VARIABLES:

      INTEGER*4     MATRX                   !Pointer to Var-Covar matrix
      INTEGER*4     IUCIF                   !Unit no. for cif file
      CHARACTER*68  DESCR                   !Experiment title
      CHARACTER*255 EXPNAM                  !Experiment file name
      CHARACTER*20  EXPRNAME                !Experiment name = name of data block
      CHARACTER*255 CMTNAM
      CHARACTER*80  TEXT                    !ISAM data string
      CHARACTER*255 MSG                     !
      CHARACTER*100 MSG1                    !
      INTEGER*4     NPHAS(9)                !Phase existance flags
      INTEGER*4     NUMPHAS                 !No. of phases
      LOGICAL*4     IFPWDR                  ! true when powder data are present
      LOGICAL*4     IFSNGL                  ! true when single xtal data are present
      LOGICAL*4     IXST
      LOGICAL*4     ONEBLOCK                ! true if the CIF will have one block
      CHARACTER*4   HTYP(99)                !Histo. types
      CHARACTER*4   HTYPE                   !Current histogram type
      CHARACTER*30  INSTNAME                ! name of instrument (for I.D.)
      CHARACTER*12  KEYVAL                  !ISAM key
      INTEGER*4     IUPRF                   !Unit no. for powder histogram
      CHARACTER*24  DAYTIME
      CHARACTER*50  AUTHOR                  !The dreaded Author name appears again
      CHARACTER*24  SAUTHOR                 !A shortened version of the author name
      CHARACTER*3   MONTH

!SUBROUTINES CALLED:

!FUNCTION DEFINITIONS:
      INTEGER*4     READEXP                 !ISAM file read function
      CHARACTER*6   CRSKEY                  !ISAM key building routine
      CHARACTER*6   HSTKEY                  !ISAM key building routine

!DATA STATEMENTS:

!CODE:


      CALL STRTRN('GSAS2CIF','SHARED','LIST',IUEXP,IULST,IUTRM)
      CALL PROGNAM(IUTRM,'GSAS2CIF','Generate CIF files|'//
     1      'Original design by Brian Toby, NIST')

      INQUIRE(UNIT=IUEXP,NAME=EXPNAM)
      LEXPNM = INDEX(EXPNAM,'.EXP')-1
! drop the directory names
      IST = 1
      DO I = 1,LEXPNM
         IF (EXPNAM(I:I) .EQ. '/' .OR. EXPNAM(I:I) .EQ. '\\') IST = I+1    ! written
for  -fbackslash
      ENDDO
      EXPRNAME = EXPNAM(IST:LEXPNM)
      CALL VSTRNG(EXPRNAME,LENCH(EXPRNAME),.TRUE.,.TRUE.)


      CALL GETUNIT(IUCIF)
      OPEN(UNIT=IUCIF,FILE=EXPNAM(1:LEXPNM)//'.cif',
     1      STATUS='UNKNOWN',
     1      FORM='FORMATTED')


      CALL GSDATE(DAYTIME)
! reformat the date in the preferred CIF format yyyy-mm-ddThh:mm
      MONTH = DAYTIME(1:3)
      CALL UPCASE(MONTH)
      IF (MONTH .EQ. 'JAN') THEN
         MONTH = '01-'
      ELSEIF (MONTH .EQ. 'FEB') THEN
         MONTH = '02-'
      ELSEIF (MONTH .EQ. 'MAR') THEN
         MONTH = '03-'
      ELSEIF (MONTH .EQ. 'APR') THEN
         MONTH = '04-'
      ELSEIF (MONTH .EQ. 'MAY') THEN
         MONTH = '05-'
      ELSEIF (MONTH .EQ. 'JUN') THEN
         MONTH = '06-'
      ELSEIF (MONTH .EQ. 'JUL') THEN
         MONTH = '07-'
      ELSEIF (MONTH .EQ. 'AUG') THEN
         MONTH = '08-'
      ELSEIF (MONTH .EQ. 'SEP') THEN
         MONTH = '09-'
      ELSEIF (MONTH .EQ. 'OCT') THEN
         MONTH = '10-'
      ELSEIF (MONTH .EQ. 'NOV') THEN
         MONTH = '11-'
      ELSE
         MONTH = '12-'
      ENDIF
      IF (DAYTIME(5:5) .EQ. ' ') DAYTIME(5:5) = '0'
      DAYTIME = DAYTIME(17:20)//'-'//MONTH//DAYTIME(5:6)//
     $     'T'//DAYTIME(8:15)
! the software does not expect spaces in the date
      CALL VSTRNG(DAYTIME,LENCH(DAYTIME),.TRUE.,.TRUE.)


C get an author name if one is not saved
      ISAM = READEXP(IUEXP,'CIF AUTHOR  ',TEXT)
      IF ( ISAM.NE.0 ) THEN
         WRITE (MSG,'(A,I2,A)') 'Please enter your name:'
         CALL REDTRML('Enter your name:',AUTHOR)
         CALL WRITEXP(IUEXP,'CIF AUTHOR  ','  '//AUTHOR)
      ELSE
         AUTHOR = TEXT(3:)
      END IF
! the software does not expect spaces in the short version of the Author name
      I = lench(AUTHOR)
      IF (I .GT. 20) THEN
         SAUTHOR = AUTHOR(I-19:)
      ELSE
         SAUTHOR = AUTHOR
      ENDIF
      CALL VSTRNG(SAUTHOR,LENCH(SAUTHOR),.TRUE.,.TRUE.)


C count the number of phases & histograms -- use a single CIF block
C if there is only one of each
      CALL GETNPHAS(IUEXP,NPHAS)
C count phase(s)
      NUMPHAS = 0
      DO I=1,9
        IF ( NPHAS(I).GT.0 ) NUMPHAS = NUMPHAS+1
      END DO
      IFLAG = READEXP(IUEXP,' EXPR  NHST ',TEXT)
      READ (TEXT,'(I5)') NHIST
      CALL RDHTYP(IUEXP,NHIST,HTYP)
C count histogram(s) & set instrument names
      IFLAG = READEXP(IUEXP,' EXPR  NHST ',TEXT)
      READ (TEXT,'(I5)') NHIST
      CALL RDHTYP(IUEXP,NHIST,HTYP)
      IFPWDR = .FALSE.
      IFSNGL = .FALSE.
      NPWDHIST = 0
      INSTNAME = ' '
      DO IHST=1,NHIST
         HTYPE = HTYP(IHST)
         IF ( (HTYPE(1:1).EQ.'S' .OR. HTYPE(1:1).EQ.'P')
     1        .AND. HTYPE(4:4).NE.'*' ) THEN
```

```
            NPWDHIST = NPWDHIST + 1

            ISAM = READEXP(IUEXP,HSTKEY(IHST)//' INAME',TEXT)
            IF ( ISAM.NE.0 ) THEN
C is this a default? Is there something to consider as a default?
               CALL WRHNAM(IUEXP,IHST,HTYPE)
               IF (INSTNAME .EQ. ' ') THEN
                  WRITE (MSG,'(A,I2,A)') 'Histogram',IHST,
     $                 ' has no diffractometer name|'//
     1                 '| Enter a name for the diffractometer:'
               ELSE
                  WRITE (MSG,'(A,I2,4A)') 'Histogram',IHST,
     $                 ' has no diffractometer name|',
     1                 '| Enter a name for the diffractometer (/ = ',
     $                 INSTNAME(:lench(INSTNAME)),'):'
               ENDIF
               CALL REDTRML(MSG(:lench(MSG)),TEXT)
               IF (TEXT .NE. '/') INSTNAME = TEXT
               CALL WRITEXP(IUEXP,HSTKEY(IHST)//' INAME','  '//INSTNAME)
            ELSE
               INSTNAME = TEXT(3:)
            END IF
            IF ( HTYPE(1:1).EQ.'P' ) IFPWDR = .TRUE.
            IF ( HTYPE(1:1).EQ.'S' ) IFSNGL = .TRUE.
         END IF
      END DO

C set the Single-block flag -- if 1 phase & 1 histogram we don't have to
C   break the CIF into more than one block
      IF (NPWDHIST .EQ. 1 .AND. NUMPHAS .EQ. 1) THEN
         ONEBLOCK = .TRUE.
      ELSE
         ONEBLOCK = .FALSE.
      ENDIF


C read the .CMT file
      CMTNAM = EXPNAM(1:LEXPNM)//'.CMT'
      INQUIRE(FILE=CMTNAM,EXIST=IXST)
      IF ( IXST ) THEN
         IUCMT = IUEXP+1
         CALL GETUNIT(IUCMT)
         OPEN(IUCMT,FILE=CMTNAM,STATUS='OLD',
     1        FORM='UNFORMATTED')
         READ (IUCMT) NCYCLE
         READ (IUCMT) MATSIZ
         CLOSE (IUCMT)
         CALL GETVM(MATRX,MATSIZ*4)
         CALL RDCOVAR(IULST,IUEXP,NUMPAR,PARNAMS,MBW,%val(MATRX))
      ELSE
         NUMPAR = 0
      END IF

      IF ( NUMPAR.EQ.0 ) THEN
         CALL REDTRML('The Variance-covarance matrix (.CMT file)'//
     $        ' cannot be read.|'//
     $        'Uncertainty estimates can not be reported.|'//
     $        'Continue anyway? (Y,[N])',MSG)
         CALL UPCASE(MSG(1:1))
         IF (MSG .NE. 'Y') STOP
      END IF

C get the cycle number and SUM(D**2) from the .EXP file
      IFLAG = READEXP(IUEXP,'  GNLS  RUN ',TEXT)                  !Read the cycle
number from the EXP file
      IF ( IFLAG.EQ.0 ) THEN
         READ (TEXT,'(43X,I4)') MCYCLE
         READ (TEXT(50:68),'(F15.0)') SUMDSQ1
      ELSE
         MCYCLE = -1
         SUMDSQ1 = -1
      END IF

C open the Distance & Angle file & check it is current?
      CMTNAM = EXPNAM(1:LEXPNM)//'.DISAGL'
      INQUIRE(FILE=CMTNAM,EXIST=IXST)
      IF (IXST) THEN
         IUDIS = IUEXP+1
         CALL GETUNIT(IUDIS)
         OPEN(IUDIS,FILE=CMTNAM,STATUS='OLD',
     1        FORM='FORMATTED')
         READ (IUDIS,'(4x,I5,G20.5)') NCYCLE,SUMDSQ
         IF (NCYCLE .NE. MCYCLE .OR. SUMDSQ .NE. SUMDSQ1) THEN
            PRINT '(3A)','The DISAGL output file does not match the',
     $           ' current refinement. Run DISAGL again.'
            PRINT '(10x,2A20)', '.DISAGL file','.EXP file'
            PRINT '(A10,2(7x,i6,7x))','Cycle:',NCYCLE,MCYCLE
            PRINT '(A10,2G20.5)','SUM(D**2):',SUMDSQ,SUMDSQ1
            CLOSE(IUDIS)
            IUDIS = 0
         ENDIF
      ELSE
         IUDIS = 0
         NCYCLE = -2
         SUMDSQ1 = -2
      ENDIF

      IF(IUDIS .EQ. 0) THEN
         CALL REDTRML('The distance & angles (.DISAGL file)'//
     $        ' cannot be read. (Was DISAGL run?)|'//
     $        'Distances and angles cannot be Reported.|'//
     $        'Continue anyway? (Y,[N])',MSG)
         CALL UPCASE(MSG(1:1))
         IF (MSG .NE. 'Y') STOP
      ENDIF

C now start creating the CIF
```

```
      PRINT '(A)',' Preparing publ section'

      ILEN = LENCH(EXPRNAME)
      WRITE(IUCIF,'(3A,/)') 'data_',EXPRNAME(1:ILEN),'_publ'

      IF (ONEBLOCK) THEN
C fix up instrument name
         I = 1
         DO WHILE (I .LT. LENCH(INSTNAME) .AND. INSTNAME(I:I) .EQ. ' ')
            I = I + 1
         ENDDO
         IF (I .GT. 1) INSTNAME = INSTNAME(I:)
         CALL VSTRNG(INSTNAME,LENCH(INSTNAME),.TRUE.,.TRUE.)
         WRITE(IUCIF,'(A,/,2x,7A)')   '_pd_block_id',
     $        DAYTIME(1:16),'|',EXPRNAME(1:ILEN),'|',
     $        SAUTHOR(:LENCH(SAUTHOR)),'|',
     $        INSTNAME(:LENCH(INSTNAME))
      ELSE
         WRITE(IUCIF,'(A,/,2x,6A)')   '_pd_block_id',
     $        DAYTIME(1:16),'|',EXPRNAME(1:ILEN),'|',
     $        SAUTHOR(:LENCH(SAUTHOR)),'|Overall'
      ENDIF

      WRITE(IUCIF,'(/A,2X,A)') '_audit_creation_method',
     1  '"from EXP file using GSAS2CIF"'
      CALL WRVAL(IUCIF,'_audit_creation_date',DAYTIME(1:16))
      CALL WRVAL(IUCIF,'_audit_author_name',AUTHOR)
      WRITE (IUCIF,'(A,/,3A,/,A,/)') '_audit_update_record',
     1     '; ',DAYTIME(1:16),'  Initial CIF as created by GSAS2CIF',
     $     ';'

C now insert the publication template
      CALL CPTMPLTE(IUCIF,' ','template_publ.cif',
     $     EXPNAM(1:LEXPNM)//'_publ.cif')

C deal with the overall refinement information
      IF (.NOT. ONEBLOCK) THEN
         WRITE(IUCIF,'(A,/)') 'data_'//EXPRNAME(1:LENCH(EXPRNAME))//
     1        '_overall'
      END IF
      CALL OVERALL(IUEXP,IUCIF,EXPRNAME,IFPWDR,HTYP,NHIST,
     $     NPWDHIST,MBW)


      DO IPHAS=1,9
         IF ( NPHAS(IPHAS).GT.0 ) THEN
            PRINT '(A,I1)',' Processing phase ',IPHAS
            IF (.NOT. ONEBLOCK) THEN
               WRITE(IUCIF,'(/,A,I2)') '# Information for phase',IPHAS
               WRITE(IUCIF,'(A,I1,/)') 'data_'//
     $              EXPRNAME(1:LENCH(EXPRNAME))//
     $              '_phase_',IPHAS
                WRITE(IUCIF,'(A,/,2X,2A,I1,4A)') '_pd_block_id',
     1              DAYTIME(1:16)//'|',
     $              EXPRNAME(1:LENCH(EXPRNAME))//'_phase',IPHAS,'|',
     $              SAUTHOR(:LENCH(SAUTHOR)),'||'
             END IF

C now insert the phase template
             WRITE(MSG,'(2A,I1,A)') EXPNAM(1:LEXPNM),'_phase',
     $            IPHAS,'.cif'
             CALL CPTMPLTE(IUCIF,' ','template_phase.cif',MSG)

             CALL WRITEPHASE(IUCIF,IUEXP,IUTRM,IPHAS,NPHAS,DAYTIME,
     $            ONEBLOCK,%val(MATRX),NUMPAR,MBW,IUDIS)
          END IF
       END DO


       CALL GETUNIT(IUPRF)
       DO IHST=1,NHIST
          PRINT '(A,I2,A)',' Begin processing histogram ',IHST,' data'
          HTYPE = HTYP(IHST)
C get & fix up instrument name
          ISAM = READEXP(IUEXP,HSTKEY(IHST)//' INAME',INSTNAME)
          I = 1
          DO WHILE (I .LT. LENCH(INSTNAME) .AND. INSTNAME(I:I) .EQ. ' ')
             I = I + 1
          ENDDO
          IF (I .GT. 1) INSTNAME = INSTNAME(I:)
          CALL VSTRNG(INSTNAME,LENCH(INSTNAME),.TRUE.,.TRUE.)

          IF ( HTYPE(1:1).EQ.'P' .AND. HTYPE(4:4).NE.'*') THEN

C Process powder histograms
             WRITE(IUCIF,'(/,A,I3)')
     1            '# Powder diffraction data for histogram',IHST
             IF (.NOT. ONEBLOCK) THEN
                WRITE(IUCIF,'(A,I2.2,/)') 'data_'//
     1              EXPRNAME(1:LENCH(EXPRNAME))//
     1              '_p_',IHST
                WRITE(IUCIF,'(A,/,2X,2A,I2.2,4A)') '_pd_block_id',
     1              DAYTIME(1:16)//'|',
     $              EXPRNAME(1:LENCH(EXPRNAME))//'_H_',IHST,'|',
     $              SAUTHOR(:LENCH(SAUTHOR)),'|',
     $              INSTNAME(:LENCH(INSTNAME))
             ENDIF

             WRITE(MSG1,'(3A)') 'template_',
     $            INSTNAME(:LENCH(INSTNAME)),'.cif'
             WRITE(MSG,'(3A,I2.2,A)') EXPNAM(1:LEXPNM),'_',
     $            INSTNAME(:LENCH(INSTNAME)),IHST,'.cif'
             CALL CPTMPLTE(IUCIF,MSG1,'template_instrument.cif',MSG)

             CALL WRPOWDHIST(IUCIF,IUEXP,IUTRM,IHST,HTYPE,IUPRF,
     1           LAM2,DAYTIME,ONEBLOCK,EXPRNAME,SAUTHOR)
             PRINT '(A,I2,A)',' Begin processing histogram ',IHST,
     1           ' reflection data'

             CALL WRREFLIST(IUEXP,IUCIF,IHST,HTYPE,NUMPHAS,LAM2,DAYTIME)
          ELSE IF ( HTYPE(1:1).EQ.'S' .AND. HTYPE(4:4).NE.'*' ) THEN

C Process single histograms
             IF (.NOT. ONEBLOCK) THEN
                WRITE(IUCIF,'(A,I2.2,/)') 'data_'//
     $              EXPRNAME(1:LENCH(EXPRNAME))//
     1              '_s_',IHST
             ENDIF
             CALL WRREFLIST(IUEXP,IUCIF,IHST,HTYPE,NUMPHAS,LAM2,DAYTIME)
          END IF
       END DO
       WRITE(IUCIF,'(21A)') '#--',('eof--',i=1,15),'#'
       STOP 'GSAS2CIF completed successfully'
       END
```

Subroutine VSTRNG for program GSAS2CIF

This subroutine is used to make sure that ASCII strings are valid for CIF -- this means only valid ASCII characters. In some cases one does not want to allow spaces in the string and/or in others one does not want a vertical bar (|) in the name. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE VSTRNG(STRING,LN,NOSPACE,NOBAR)
C subroutine to validate a string to insure there are only ASCII characters
C If NOSPACE is True, spaces are replaced with underscores (_)
C If NOBAR is True, illegal characters (| & slash characters) are replaced with
underscores
      CHARACTER*(*) STRING
      INTEGER*4     LN
      LOGICAL*4     NOSPACE
      LOGICAL*4     NOBAR
      IF (NOSPACE) THEN
        DO I = 1,LN
          IF (ICHAR(STRING(I:I)) .LE. 32 .OR.
     1      ICHAR(STRING(I:I)) .GT. 176) STRING(I:I) = '_'
          IF (NOBAR .AND. (STRING(I:I) .EQ. '|' .OR.
     1      STRING(I:I) .EQ. '/' .OR. STRING(I:I) .EQ. '\\'))
     1      STRING(I:I) = '_'
        END DO
      ELSE
        DO I = 1,LN
          IF (ICHAR(STRING(I:I)) .LT. 32 .OR.
     1      ICHAR(STRING(I:I)) .GT. 176) STRING(I:I) = '_'
          IF (NOBAR .AND. (STRING(I:I) .EQ. '|' .OR.
     1      STRING(I:I) .EQ. '/' .OR. STRING(I:I) .EQ. '\\'))
     1      STRING(I:I) = '_'
        END DO
      END IF
      RETURN
      END
```

Subroutine LENCH for program GSAS2CIF

This subroutine is used to find the length of an ASCII string. Trailing spaces, tab & null characters are ignored. See the gsas2cif documentation for an explanation of this code.

```
      INTEGER*4 FUNCTION LENCH(STR)
C---------------------------------------------------------------------------
c       Function LENCH
c
c This function takes a character string and finds out how long the
c "actual" string is (i.e. not including padded blanks on the right).
c
C---------------------------------------------------------------------------
!Calling arguments:

      CHARACTER*(*) STR

!Local variables:

      CHARACTER*1   NUL,TAB
      LOGICAL*4     DONE

!Data:

      data      nul      /0/
      data      tab      /'      '/

!Code:

      if ( str.ne.' ' .and. str(1:1).ne.nul ) then
        ilench = len(str)+1
        done = .false.
        do while ( .not.done .and. ilench.gt.0 )
          ilench = ilench-1
          if ( str(ilench:ilench).ne.' '
     1      .and. str(ilench:ilench).ne.tab            !Look for trailing tabs as
well
     1      .and. str(ilench:ilench).ne.nul ) done=.true.
        END DO
        lench = ilench
      else
        lench=0
      end if
      return
      end
```

Subroutine WRVAL for program GSAS2CIF

This subroutine is used to write a CIF data item to file. Subroutine ADDQUOTE is used to add quotation marks (if any are needed). Note that this routine does not break lines, so it should not be passed values that are more than 78 characters. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE wrval(IUCIF,lbl,value)
      INTEGER*4     IUCIF
      CHARACTER*(*) LBL,VALUE
      CHARACTER*80  VALUE2
      INTEGER*4     LN1,LN2
      ln1 = max(1,LENCH(lbl))
      CALL ADDQUOTE(value,value2,ln2)
      IF (ln2 .le. 40) then
        write (IUCIF,'(a,t40,a)') lbl(:ln1),value2(:ln2)
      ELSEIF (ln2 .le. 60) then
        write (IUCIF,'(a,/,t20,a)') lbl(:ln1),value2(:ln2)
      ELSEIF (ln2 .le. 75) then
        write (IUCIF,'(a,/,t5,a)') lbl(:ln1),value2(:ln2)
      ELSE
        IF (ln2 .le. 79) write (*,*)
     1    'Error -- value for ',lbl(:ln1),' is too long!'
        write (IUCIF,'(a,/,1x,a)') lbl(:ln1),value2(:ln2)
      END IF
      RETURN
      END
```

Subroutine CPTMPLTE for program GSAS2CIF

This subroutine is used to insert the contents of a CIF template file into a CIF file. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE CPTMPLTE(IUCIF,TEMPLATE1,TEMPLATE2,LOCALCOPY)
C Copy Template File
C This subroutine opens the file referenced by LOCALCOPY and copies
C the contents, line by line to the output CIF file (IUCIF).
C
C If this file does not exist, a master template file named TEMPLATE1
C is opened and a file named LOCALCOPY is created. If that does not exist
C or is blank,  a master template file named TEMPLATE2 is opened.
C
C The master template file is then copied to are copied both to the
C output CIF file (IUCIF) and the LOCALCOPY file.
C
C The master template file will be read from the current default data
C data directory, if it exists, otherwise it is read from the GSAS
C data directory.

      INTEGER*4     IUCIF                   !Unit no. for cif file
      CHARACTER*(*) TEMPLATE1               !Name of template file #1
      CHARACTER*(*) TEMPLATE2               !Name of template file #2
      CHARACTER*(*) LOCALCOPY               !Name of local copy of template file

!LOCAL VARIABLES:
      INTEGER*4     IUIN                    !Unit no. for input file
      INTEGER*4     IUCP                    !Unit no. for output localcopy file (if
needed)
      INTEGER*4     IFLAG                   !Open error flag
      INTEGER*4     ILONG                   !# of too long lines
      INTEGER*4     L

      CHARACTER*100 LINE                    !temp variable
      CHARACTER*255 FULLNAME                !full path for template file
      CHARACTER*255 GSAS                    !location of GSAS files
      LOGICAL*4     DATAFLAG                !set to true after the data_ line is read

!FUNCTION DEFINITIONS:

      INTEGER*4     GSGETENV                !get a environment variable
      INTEGER*4     LENCH                   !length of a character string


      IUCP = 0
      CALL GETUNIT(IUIN)


C first try to open the LOCALCOPY, if it exists
```

```
      OPEN(UNIT=IUIN,FILE=LOCALCOPY,
     1  IOSTAT=IFLAG,                                          !error flag
     1  STATUS='OLD',FORM='FORMATTED')
      IF (IFLAG .EQ. 0) THEN
        PRINT '(2A)',' Copying from file ',LOCALCOPY
      END IF



      IF (IFLAG .NE. 0 .AND. TEMPLATE1 .NE. ' ') THEN
C     open failed, open the 1st template
C       look first for a template in the current directory
        OPEN(UNIT=IUIN,FILE=TEMPLATE1,
     1    IOSTAT=IFLAG,                                          !error flag
     1    STATUS='OLD',FORM='FORMATTED')
        IF ( IFLAG.EQ.0 ) THEN
          PRINT '(2A)',' Reading from current directory, file ',
     1      TEMPLATE1(:LENCH(TEMPLATE1))
          CALL GETUNIT(IUCP)
          OPEN(UNIT=IUCP,FILE=LOCALCOPY,STATUS='NEW',
     1      FORM='FORMATTED')
          PRINT '(2A)',' Creating file ',LOCALCOPY
        ELSE
C       not found, look in the GSAS data directory
          IFLAG = GSGETENV('gsas    ',GSAS)
          IF ( IFLAG.EQ.0 ) STOP
     1      'ERROR - Environment variable GSAS is undefined'
          FULLNAME = GSAS(1:INDEX(GSAS,' ')-1)//'/data/'//TEMPLATE1
          OPEN(UNIT=IUIN,FILE=FULLNAME,
     1      IOSTAT=IFLAG,
     1      STATUS='OLD',FORM='FORMATTED')
          IF ( IFLAG.NE.0 ) THEN
            PRINT '(3A)',' File ',TEMPLATE1(:LENCH(TEMPLATE1)),
     1        ' not found in current or GSAS data directory'
            PRINT '(4A)',' will try generic template'
          ELSE
            PRINT '(2A)',' Reading from GSAS data directory, file ',
     1        TEMPLATE1(:LENCH(TEMPLATE1))
            CALL GETUNIT(IUCP)
            OPEN(UNIT=IUCP,FILE=LOCALCOPY,STATUS='NEW',
     1        FORM='FORMATTED')
            PRINT '(2A)',' Creating file ',LOCALCOPY
          END IF
        END IF
      END IF



      IF (IFLAG .NE. 0) THEN
C     open failed, open the 2nd template
C       look first for a template in the current directory
        OPEN(UNIT=IUIN,FILE=TEMPLATE2,
     1    IOSTAT=IFLAG,                                          !error flag
     1    STATUS='OLD',FORM='FORMATTED')
        IF ( IFLAG.EQ.0 ) THEN
          PRINT '(2A)',' Reading from current directory, file ',
     1      TEMPLATE2(:LENCH(TEMPLATE2))
        ELSE
C       not found, look in the GSAS data directory
          IFLAG = GSGETENV('gsas    ',GSAS)
          IF ( IFLAG.EQ.0 ) STOP
     1      'ERROR - Environment variable GSAS is undefined'
          FULLNAME = GSAS(1:INDEX(GSAS,' ')-1)//'/data/'//TEMPLATE2
          OPEN(UNIT=IUIN,FILE=FULLNAME,
     1      IOSTAT=IFLAG,
     1      STATUS='OLD',FORM='FORMATTED')
          IF ( IFLAG.NE.0 ) THEN
            PRINT '(2A)',' Error: could not find file ',
     1        TEMPLATE2(:LENCH(TEMPLATE2))
            PRINT '(4A)','  This file is missing from GSAS',
     1        ' data directory, ',GSAS(1:INDEX(GSAS,' ')-1),
     1        '/data/'
            STOP 'ERROR - missing template file'
          END IF
          PRINT '(2A)',' Reading from GSAS data directory, file ',
     1      TEMPLATE2(:LENCH(TEMPLATE2))
        END IF
        CALL GETUNIT(IUCP)
        OPEN(UNIT=IUCP,FILE=LOCALCOPY,STATUS='NEW',FORM='FORMATTED')
        PRINT '(2A)',' Creating file ',LOCALCOPY
      END IF



C got the input file, now read from it
      ILONG = 0
      DATAFLAG = .FALSE.
      READ (IUIN,'(A)',IOSTAT=IFLAG) LINE
      DO WHILE (IFLAG .EQ. 0)
        L = LENCH(LINE)                                     ! is the line too
long?
        IF (L .GT. 80) THEN
          L = 80
          ILONG = ILONG + 1
        END IF
C don't copy lines that preceed the data_ token
        IF (DATAFLAG) WRITE (IUCIF,'(A)') LINE(:L)
        IF (IUCP .NE. 0) WRITE (IUCP,'(A)') LINE(:L)
C Did this line contain a data_ block name?
        IF (.NOT. DATAFLAG) THEN
          I = 1
          DO WHILE (LINE(I:I) .EQ. ' ' .AND. I .LT. L)
            I = I + 1
          END DO
          CALL UPCASE(LINE)
          IF (LINE(I:I+4) .EQ. 'DATA_') DATAFLAG = .TRUE.
        END IF
        READ (IUIN,'(A)',IOSTAT=IFLAG) LINE
      END DO

      IF (ILONG .GT. 0) PRINT '(A,I5,A)',' Warning:',ILONG,
     1  ' lines longer than 80 characters were truncated'
      IF (IUCP .NE. 0) CLOSE(IUCP)
      CLOSE(IUIN)
      RETURN
      END
```

Subroutine OVERALL for program GSAS2CIF

This subroutine is used to write overall parameters and results to the output CIF file. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE OVERALL(IUEXP,IUCIF,EXPRNAME,IFPWDR,HTYP,NHIST,
     1     NPWDHIST,MBW)

      INTEGER*4      IUEXP,IUCIF          !Unit nos.
      CHARACTER*8    EXPRNAME              !Experiment name
      LOGICAL*4      IFPWDR               ! true if there is one or more powder 
histogram present
      CHARACTER*4    HTYP(99)             !Histogram type flags
      INTEGER*4      NHIST                !The number of histograms in this experiment
      INTEGER*4      NPWDHIST             ! The number of powder histograms
      INTEGER*4      MBW                  !Matrix bandwidth

!Local variables:

      CHARACTER*68   TEXT                 !ISAM file read buffer
      CHARACTER*20   DAT1                 !Mean value of |Shift/esd|
      CHARACTER*20   DAT2                 !Maximum value of |shift/esd|

!Functions:

      INTEGER*4      READEXP              !ISAM file read routine
      CHARACTER*6    HSTKEY               !ISAM key builder

!Code:

      ISAM = READEXP(IUEXP,'  GNLS SHFTS',TEXT)
      IF ( ISAM.EQ.0 ) THEN
        DAT1 = TEXT(1:10)
        DAT2 = TEXT(11:20)
      ELSE
C if the Shifts are not present -- they are not defined, not unknown
        DAT1 = '.'
        DAT2 = '.'
      END IF
      CALL WRVAL(IUCIF,'_refine_ls_shift/su_max',DAT1)
      CALL WRVAL(IUCIF,'_refine_ls_shift/su_mean',DAT2)
      CALL WRVAL(IUCIF,'_computing_structure_refinement','GSAS')
      ISAM = READEXP(IUEXP,' REFN GDNFT ',TEXT)
C likewise for GOF, etc -- they are not defined, not unknown
      IF ( ISAM.EQ.0 ) THEN
        DAT1 = TEXT(33:37)
        READ (TEXT(18:28),'(F11.0)') GNFT
        WRITE (DAT2,'(F7.2)') SQRT(GNFT)
      ELSE
        DAT1 = '.'
        DAT2 = '.'
      END IF
      CALL WRVAL(IUCIF,'_refine_ls_number_parameters',DAT1)
      CALL WRVAL(IUCIF,'_refine_ls_goodness_of_fit_all',DAT2)
      ISAM = READEXP(IUEXP,' REFN RESTR ',TEXT)
      IF ( ISAM.NE.0 ) TEXT = '      0'
      CALL WRVAL(IUCIF,'_refine_ls_number_restraints',TEXT(1:7))

C things to consider computing
      ! _refine_ls_number_reflns
      ! _refine_ls_goodness_of_fit_obs
      ! _refine_ls_R_factor_all
      ! _refine_ls_R_factor_obs
      ! _refine_ls_wR_factor_all
      ! _refine_ls_wR_factor_obs
      ! _refine_ls_restrained_S_all
      ! _refine_ls_restrained_S_obs



C include an overall profile r-factor, if there is more than one powder histogram
      IF ( IFPWDR .AND. NPWDHIST .GT. 1) THEN
        WRITE(IUCIF,'(/A/)') '# Overall powder R-factors'
        ISAM = READEXP(IUEXP,' REFN RPOWD ',TEXT)
        IF ( ISAM.EQ.0 ) THEN
          CALL WRVAL(IUCIF,'_pd_proc_ls_prof_R_factor',TEXT(11:20))
          CALL WRVAL(IUCIF,'_pd_proc_ls_prof_wR_factor',TEXT(1:10))
        ELSE
          CALL WRVAL(IUCIF,'_pd_proc_ls_prof_R_factor','.')
          CALL WRVAL(IUCIF,'_pd_proc_ls_prof_wR_factor','.')
        END IF
      END IF

      IF (MBW .EQ. 0) THEN
        CALL WRVAL(IUCIF,'_refine_ls_matrix_type','full')
      ELSE
        CALL WRVAL(IUCIF,'_refine_ls_matrix_type','userblocks')
      END IF
      RETURN
      END
```

Subroutine WRITEPHASE for program GSAS2CIF

This subroutine is used to write parameters and results for each phase to the output CIF file. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE WRITEPHASE(IUCIF,IUEXP,IUTRM,IPHAS,NPHAS,DAYTIME,
     1     ONEBLOCK,MATRX,NUMPAR,MBW,IUDIS)
!PURPOSE: write information about the phase

      INCLUDE       '../INCLDS/COPYRIGT.FOR'

!PSEUDOCODE:

!CALLING ARGUMENTS:

      INTEGER*4     IUCIF
      INTEGER*4     IUEXP
      INTEGER*4     IUTRM
      INTEGER*4     IPHAS
      INTEGER*4     NPHAS(9)            !Phase existence flags
      CHARACTER*20  DAYTIME
      LOGICAL*4     ONEBLOCK            ! true if the CIF will have one block
      REAL*4        MATRX(1)
      INTEGER*4     NUMPAR              !Number of refined parameters
      INTEGER*4     MBW                 !Matrix bandwidth

!INCLUDE STATEMENTS:

      INCLUDE       '../INCLDS/ARRAYSZE.FOR'
      INCLUDE       '../INCLDS/SPGCOMI.FOR'
      INCLUDE       '../INCLDS/HEADSCOM.FOR'
      INCLUDE       '../INCLDS/DISAGLCM.FOR'
      INCLUDE       '../INCLDS/CELLCOM.FOR'

!LOCAL VARIABLES:

      INTEGER*4     IOPRTNS(50)
      INTEGER*4     ISAM
      INTEGER*4     JMLT(MAXATM)        !Atom site multiplicities
      INTEGER*4     NSYS(14)
     1             /1,2,3,4,4,5,5,6,6,6,7,7,8,8/
      REAL*4        ANGLES(3)
      REAL*4        ANGSIG(3)
      REAL*4        RM(6)               !Recip. metr. tensor
      REAL*4        VOLUME              !Unit cell volume (=0 for error)
      REAL*4        UB(3,3)             !UB-matrix
      LOGICAL*4     ANIFLAG
      CHARACTER*1   CLBL(3)
     1             /'a','b','c'/
      CHARACTER*5   ALBL(3)
     1             /'alpha','beta ','gamma'/
      CHARACTER*20  STRING(10)
      CHARACTER*80  TEXT                !ISAM file read write buffer
      CHARACTER*12  SYST(8)
     1            /'triclinic   ','monoclinic  ','orthorhombic',
     1             'tetragonal  ','trigonal    ','trigonal    ',
     1             'hexagonal   ','cubic       '/
      CHARACTER*4   XYZLBL(9)
     1            /'-z  ','-y  ','-x  ','x-y ','ERR ','y-x ',
     1             '+x  ','+y  ','+z  '/
      CHARACTER*4   TRA(13)
     1            /'    ','ERR ','+1/6','+1/4','+1/3','ERR ',
     1             '+1/2','ERR ','+2/3','+3/4','+5/6','ERR ',' '/
      CHARACTER*4   OUTL(6,2)
      REAL*4        CONC(MAXELEM)
      LOGICAL*4     NOTDONE
      INTEGER*4     NOFFSET               ! number of symmetry positions
      INTEGER*4     OFFSYMID(192)         ! symmetry ID needing offset correction
      INTEGER*4     OFFSET(192)           ! offset to be added to 100*x+10*y+z
      CHARACTER*1   MSG
      REAL*4        COFF(MAXODF)          ! Spherical harmonic coefficients
      INTEGER*4     INDX(3,MAXODF)        ! Sph. harmonic index
      INTEGER*4     ISAMSYM               ! Sample symmetry number (1-4)
      CHARACTER*12  KEYVAL                ! ISAM key
      CHARACTER*2   ELEMTBL(MAXELEM)      ! Table of unique elements
      CHARACTER*2   ELEM
      REAL*4        COMPTBL(MAXELEM)      ! Number of atoms for each unique
element/cell -- full occupied sites
      REAL*4        FRACTBL(MAXELEM)      ! Number of atoms for each unique
element/cell -- partially occupied sites
      REAL*4        MASSTBL(MAXELEM)      ! Mass for each type of atom
      INTEGER*2     SEQTBL(MAXELEM)       ! Sequence to show elements
      INTEGER*4        Z
      LOGICAL       FLAG
      CHARACTER*1   PUBFLG                ! Y is distances/angles will be published

!SUBROUTINES CALLED:

!FUNCTION DEFINITIONS:

      INTEGER*4     READEXP               !ISAM file read function
      CHARACTER*6   CRSKEY                !ISAM key building routine

!Code:

      NOFFSET = 0
      TEXT = ' '
      IULST = 6


      CALL DSGREAD(IUEXP,IULST,IPHAS,NPHAS(IPHAS),'NOFA',NUMPAR,
     1  MBW,MATRX)            !Read unit cell and atom data


      ISAM = READEXP(IUEXP,CRSKEY(IPHAS)//'  PNAM',TEXT)
      CALL WRVAL(IUCIF, '_pd_phase_name', text)


      ISAM = READEXP(IUEXP,CRSKEY(IPHAS)//'ANGLES',TEXT)
      READ (TEXT,'(3F10.0)') ANGLES
      ISAM = READEXP(IUEXP,CRSKEY(IPHAS)//'ANGSIG',TEXT)
      IF ( ISAM.EQ.0 ) THEN
        READ (TEXT,'(3F10.0)') ANGSIG
      ELSE
        DO I=1,3
          ANGSIG(I) = 0.0
        END DO
      END IF
      CALL BMATRX(ABC(1,IPHAS),ANGLES,UB,
     1  ABCST(1,IPHAS),CANGST(1,IPHAS))
      DO I=1,3
        IF ( (LAUE.GT.3 .AND. I.GT.1) .AND.                        !Laue symmetry
above mmm
     1    (I.EQ.2 .OR.
     1    (((LAUE.EQ.6 .OR. LAUE.EQ.7) .OR.            !Rhombohedral symmetry
     1    LAUE.GT.11) .AND. I.EQ.3)) ) THEN            !Cubic symmetry
          CALL FESD(ABC(I,IPHAS), -CELSIG(I), text, ln)
          CALL WRVAL(IUCIF, '_cell_length_'//clbl(I),text)
        ELSE IF (CELSIG(I) .le. 0.0) THEN
          CALL FESD(ABC(I,IPHAS), 0.0, text, ln)
          CALL WRVAL(IUCIF, '_cell_length_'//clbl(I),text)
        ELSE
          CALL FESD(ABC(I,IPHAS), CELSIG(I), text, ln)
          CALL WRVAL(IUCIF,'_cell_length_'//clbl(I),text)
        END IF
      END DO
      DO I=1,3
        NOTDONE = .TRUE.
        IF ( LAUE.GT.1 ) THEN
          IF ( (LAUE.EQ.6 .OR. LAUE.EQ.7) ) THEN                 !Rhombohedral
setting
            IF (I .gt. 1) THEN
              NOTDONE = .FALSE.
            END IF
          ELSE IF ( (LAUE.GT.7 .AND. LAUE.LT.12) .AND. I.EQ.3 )      !Hexagonal cell,
Gamma angle
     1      THEN
            NOTDONE = .FALSE.
          ELSE IF ( (LAUE.EQ.2 .AND. NAXIS.NE.I) .OR.            !Monoclinic, not the
unique axis
     1      LAUE.GT.2 ) THEN                        !Anything else
            NOTDONE = .FALSE.
          END IF
        END IF
        IF ( NOTDONE .and. ANGSIG(I) .gt. 0) THEN
          CALL FESD(ANGLES(I), ANGSIG(I), text, ln)
          CALL WRVAL(IUCIF, '_cell_angle_'//albl(I),text)
        ELSE IF (ANGSIG(I) .gt. 0) THEN
          CALL FESD(ANGLES(I), -ANGSIG(I), text, ln)
          CALL WRVAL(IUCIF, '_cell_angle_'//albl(I),text)
        ELSE
          CALL FESD(ANGLES(I), 0.0, text, ln)
```

```
            CALL WRVAL(IUCIF, '_cell_angle_'//albl(I),text)
          END IF
        END DO
        CALL CELVOL(ABC(1,IPHAS),ANGLES,RM,VOLUME)
        CALL FESD(VOLUME, 0.0, text, ln)
        CALL WRVAL(IUCIF, '_cell_volume',text)


        CALL WRVAL(IUCIF,'_symmetry_cell_setting',SYST(NSYS(LAUE)))

        WRITE(text,'(20a1)') SPG
        ln = LENCH(text)
C a R suffix is a GSAS code for a rhombohedral setting & should be removed
        if (text(ln:ln) .eq. 'R') text(ln:ln) = ' '
        CALL WRVAL(IUCIF,'_symmetry_space_group_name_H-M',
     1  text(1:LENCH(TEXT)))

        WRITE (IUCIF,'(2A)') 'loop_ _symmetry_equiv_pos_site_id',
     1     ' _symmetry_equiv_pos_as_xyz'
        DO ICV=1,NCV                                          !Loop over the lattice
centering
          DO JCEN=0,ICEN                                      !Loop over the identity
and inversion
            DO I=1,NSYM                                       !Loop through the
matrices
              IM = 100
              IOFF = 0
              DO J=1,3
                IJ = 2*JRT(J,1,I)+3*JRT(J,2,I)+4*JRT(J,3,I)+5
                IK1 = JRT(J,4,I)+NINT(CEN(J,ICV)*12.0)+1
                IK = MOD(JRT(J,4,I)+NINT(CEN(J,ICV)*12.0),12)+1
C has a offset been applied to the symmetry operator?
                IF (IK .NE. IK1) THEN
                  IF (JCEN .EQ. 1) THEN
                    IOFF = IOFF - IM*(IK1-IK)/12
                  ELSE
                    IOFF = IOFF + IM*(IK1-IK)/12
                  END IF
                END IF
                IM = IM/10
                IF ( JCEN.EQ.0 ) THEN
                  OUTL(J,1) = XYZLBL(IJ)
                  OUTL(J,2) = TRA(IK)
                ELSE
                  IJ = 10-IJ
                  OUTL(J,1) = XYZLBL(IJ)
                  IK = 14-IK
                  OUTL(J,2) = TRA(IK)
                END IF
              END DO
              I1MX = 3
              TEXT = ' '
              LN = 1
              DO I1=1,I1MX
                DO I2=1,2
                  LNX = LENCH(outl(i1,i2))
                  IF ( LNX.GT.0 ) THEN
                    TEXT(LN:) = outl(i1,i2)(:LNX)
                    LN = LN+LNX
                  END IF
                END DO
                IF ( MOD(I1,3).NE.0 ) THEN
                  TEXT(LN:LN) = ','
                  LN = LN+1
                ELSE
                  K = 100*(ICV-1) + I
                  IF (JCEN .EQ. 1) K = -K
                  WRITE (IUCIF,'(3X,I5,1x,A)') K, TEXT(:LN)
                  IF (IOFF .NE. 0) THEN
                    NOFFSET = NOFFSET +1
                    IF (NOFFSET .GT. 192) THEN
                      PRINT '(A)','More than 192'//
       1                'Offset symmetry ops -- how did this happen!'
                      STOP 'NOFFSET > 192'
                    END IF
                    OFFSYMID(NOFFSET) = K
                    OFFSET(NOFFSET) = IOFF
                  END IF
                  LN = 1
                END IF
              END DO
            END DO
          END DO
        END DO

  C initialize chemical formula arrays
        DO I=1,MAXELEM
          ELEMTBL(I) = '  '
          COMPTBL(I) = 0.
          FRACTBL(I) = 0.
          MASSTBL(I) = 0.
          SEQTBL(I) = 0
        END DO
        Z = 1


        WRITE(IUCIF,'(/A/)') '# ATOMIC COORDINATES'//
       1  ' AND DISPLACEMENT PARAMETERS'

        WRITE(IUCIF,'(/a5)') 'loop_'
        WRITE(IUCIF,'(6x,A)') '_atom_site_type_symbol',
       1  '_atom_site_label',
       1  '_atom_site_fract_x',
       1  '_atom_site_fract_y','_atom_site_fract_z',
       1  '_atom_site_occupancy',
       1  '_atom_site_thermal_displace_type',
       1  '_atom_site_U_iso_or_equiv',
       1  '_atom_site_symmetry_multiplicity'
        ANIFLAG = .false.
```

```
C Warn on duplicate labels
      DO J=2,NATOM
        DO I=1,J-1
          IF (ATMNAM(I)(:LENCH(ATMNAM(I))) .EQ.
     1      ATMNAM(J)(:LENCH(ATMNAM(J)))
     1      .AND. FRACT(I) .NE. 0 .AND. FRACT(J) .NE. 0) THEN
            PRINT '(A,i5,A,I5,2A)','Atoms',I,' and ',J,
     1        ' are both labeled',ATMNAM(I)
            CALL REDTRML('This is not allowed in a CIF'//
     1        'Continue anyway? (,N)',MSG)
            CALL UPCASE(MSG)
            IF (MSG .EQ. 'N') STOP
          END IF
        END DO
      END DO
      NA = 0

      DO I=1,NATOM
        IF (FRACT(I) .NE. 0 .OR. SGFRAC(I) .NE. 0) THEN
          NA = NA + 1
          CALL SYTSYM (XYZ(1,I),ICEN,NSYM,JRT,NCV,CEN,LAUE, !Get site symmetries and
multiplicities
     1      JMLT(I),JSYM,IOPRTNS,STSYM(I))
C  _atom_site_label
C   Note: atom labels must be unique -- if need be, concatenate (_xxx)
C   where xxx is the atom number -- not implemented at this time, instead warn
(above)
          string(1) = ATMNAM(I)(:LENCH(ATMNAM(I)))
          DO K=1,3
            CALL FESD(XYZ(K,I), SXYZ(K,I), string(K+1), ln)
          END DO
          CALL FESD(FRACT(I), SGFRAC(I), string(5), ln)
          IF ( REFCODE(I)(1:1).EQ.'I' ) THEN
            STRING(6) = 'Uiso'
          ELSE
            ANIFLAG = .true.
            STRING(6) = 'Uani'
          END IF
          IF ( REFCODE(I)(1:1).EQ.'I' ) THEN
            CALL FESD(BIJ(1,I), SBIJ(1,I), string(7), ln)
          ELSE
            UEQV = 0.0
            DO IJ=1,3
              IJ1 = IJ+1
              IF ( IJ1.EQ.4 ) IJ1=3
              IJ2 = 1
              IF ( IJ.EQ.3 ) IJ2=2
              UEQV = UEQV
     1          +BIJ(IJ,I)*(ABC(IJ,IPHAS)*ABCST(IJ,IPHAS))**2
     1          +2.0*BIJ(IJ+3,I)*ABC(IJ2,IPHAS)*ABC(IJ1,IPHAS)
     1          *ABCST(IJ2,IPHAS)*ABCST(IJ1,IPHAS)
     1          *CANG(4-IJ,IPHAS)
            END DO
            CALL FESD(UEQV/3.0, -0.0001, string(7), ln)
          END IF
          WRITE(string(8),'(i4)') JMLT(I)
          CALL VSTRNG(ATMTYP(I),LENCH(ATMTYP(I)),.true.,.false.)
          write(IUCIF,'(A)') ATMTYP(I)
          write(IUCIF,'(A6,4a13,a5,a13,a4,A)') (string(J),J=1,8)
C enter the atom into the composition table
          ELEM = ATMTYP(I)(1:2)
C is this a one-letter or two-letter element?
          JCH = ICHAR(ELEM(2:2))
          IF (JCH .LT. ICHAR('A') .OR. JCH .GT. ICHAR('Z')) THEN
             KEYVAL = ' AFAC  '//ELEM(1:1)//'_'
             ELEM(2:2) = ' '
          ELSE
             KEYVAL = ' AFAC '//ELEM(1:2)//'_'
             ELEM(2:2) = CHAR(JCH+32)
          ENDIF
          J = 1
          DO WHILE (ELEMTBL(J) .NE. '  ' .AND. ELEMTBL(J) .NE. ELEM)
            J = J + 1
          END DO
          ELEMTBL(J) = ELEM
          TEXT = ' '
          ISAM = READEXP(IUEXP,KEYVAL,TEXT)
          READ (TEXT,'(F7.0)') MASSTBL(J)
          IF (FRACT(I) .EQ. 1.0) THEN
            COMPTBL(J) = COMPTBL(J) + JMLT(I)
          ELSE
            FRACTBL(J) = FRACTBL(J) + JMLT(I)*FRACT(I)
          ENDIF
        END IF
      END DO

      IF (K .EQ. 0) THEN
        WRITE(IUCIF,'(A)') ' ? ? ? ? ? ? ? ? ?'
      ELSE IF (ANIFLAG) THEN

        WRITE(IUCIF,'(/a5,1x,A)') 'loop_', '_atom_site_aniso_label'
        WRITE(IUCIF,'(6x,A)') '_atom_site_aniso_U_11',
     1    '_atom_site_aniso_U_12','_atom_site_aniso_U_13',
     1    '_atom_site_aniso_U_22','_atom_site_aniso_U_23',
     1    '_atom_site_aniso_U_33'
        DO I=1,NATOM
          IF (FRACT(I) .NE. 0 .OR. SGFRAC(I) .NE. 0) THEN
            IF ( REFCODE(I)(1:1).EQ.'A' ) THEN
              string(1) = ATMNAM(I)(:LENCH(ATMNAM(I)))
              DO K=1,6
                CALL FESD(BIJ(K,I), SBIJ(K,I), string(K+1), ln)
              END DO
              write(IUCIF,'(A6,6a13)') (string(J),J=1,7)
            END IF
          END IF
        END DO
      END IF
C ======================================================================
C Loop over element types -- but only if the histogram info goes in a
```

```
C   separate block. In a single-block histogram, this info is included
C   with the scattering factor information (WRPOWDHI)
C

      IF (.NOT. ONEBLOCK) THEN
C determine unit cell contents
        DO I=1,MAXELEM
          conc(i) = 0.0
        END DO
        DO I=1,NATOM
          J = ID(I)                             !Get the atom type count flag
          conc(J) = conc(j) + JMLT(I)*FRACT(I)
        END DO
        WRITE(IUCIF,'(/a5,1x,A)') 'loop_', '_atom_type_symbol'
        WRITE(IUCIF,'(6x,A)') '_atom_type_number_in_cell'
        DO j=1,MAXELEM
          if (conc(j) .gt. 0.0) then
            string(1) = ATYPE(j)(1:2)
            CALL VSTRNG(string(1),2,.false.,.false.)
            CALL FESD(conc(j), -0.01, string(2), ln)
            write(IUCIF,'(t20,A2,a13)') (string(I),I=1,2)
          END IF
        END DO
      END IF



C process the chemical formula: pick a Z value & generate molecular weight
C   find the maximum possible Z value
      N = 0
      DO I=1,MAXELEM
        IF (ELEMTBL(I) .NE. ' ') N = I
      END DO

C   factors of 2
      FLAG = .TRUE.
      DO WHILE(FLAG)
        DO I=1,N
          IF (Z*2.*INT(COMPTBL(I)/(Z*2.)) .NE. COMPTBL(I))
     1          FLAG = .FALSE.
        END DO
        IF (FLAG) Z = Z * 2
      END DO
C   factors of 3
      FLAG = .TRUE.
      DO WHILE(FLAG)
        DO I=1,N
          IF (Z*3.*INT(COMPTBL(I)/(Z*3.)) .NE. COMPTBL(I))
     1          FLAG = .FALSE.
        END DO
        IF (FLAG) Z = Z * 3
      END DO
C order the elements in "Hill" order: C,H & alphabetical or alphabetical
C   is C present?
      FLAG = .FALSE.
      J = 1
      DO I=1,N
        IF (ELEMTBL(I) .EQ. 'C ') THEN
           FLAG = .TRUE.
           SEQTBL(I) = J
           J = J + 1
        END IF
      END DO
C   if yes, get H
      IF (FLAG) THEN
        DO I=1,N
          IF (ELEMTBL(I) .EQ. 'H ') THEN
            SEQTBL(I) = J
            J = J + 1
          END IF
        END DO
        DO I=1,N
          IF (ELEMTBL(I) .EQ. 'D ') THEN
            SEQTBL(I) = J
            J = J + 1
          END IF
        END DO
      END IF
      DO K=1,N
        ELEM = 'Zz'
        NUMELEM = 100*ICHAR(ELEM(1:1)) + ICHAR(ELEM(2:2))
        NN = 0
        DO I=1,N
          NUMELEM1 = 100*ICHAR(ELEMTBL(I)(1:1)) + ICHAR(ELEMTBL(I)(2:2))
          IF (NUMELEM1 .LT. NUMELEM .AND. SEQTBL(I) .EQ. 0) THEN
            NN = I
            NUMELEM = NUMELEM1
          END IF
        END DO
        IF (NN .NE. 0) THEN
          SEQTBL(NN) = J
          J = J + 1
        END IF
      END DO

      K = 1
      ATMASS = 0
      DO J=1,N
        DO I=1,N
          IF (SEQTBL(I) .EQ. J) THEN
             TEXT(K:) = ELEMTBL(I)
             IF (ELEMTBL(I)(2:2) .EQ. ' ') THEN
               K = K + 1
             ELSE
               K = K + 2
             ENDIF
             IF (FRACTBL(I) .NE. 0) THEN
                WRITE(KEYVAL,'(F12.2)')  (COMPTBL(I) + FRACTBL(I))/Z
             ELSE
                WRITE(KEYVAL,'(I12)')  NINT(COMPTBL(I)/Z)
             ENDIF
             ATMASS = ATMASS + MASSTBL(I) * (COMPTBL(I) + FRACTBL(I))/Z

             NN = 1
             DO WHILE (KEYVAL(NN:NN) .EQ. ' ')
               NN = NN + 1
             END DO
             IF (KEYVAL(NN:) .NE. '1') THEN     ! values of 1 are assumed
               TEXT(K:) = KEYVAL(NN:)
               K = K + 13 - NN
             END IF
          END IF
        END DO
        TEXT(K:K) = ' '          ! leave a blank space
        K = K + 1
      END DO
      WRITE(IUCIF,'(A)') ' ',
     1     '# If you change Z, be sure to change all 3 of the following'
      CALL WRVAL(IUCIF, '_chemical_formula_sum',text)
      WRITE(TEXT,'(F15.2)') ATMASS
      CALL WRVAL(IUCIF, '_chemical_formula_weight',text)
      WRITE(TEXT,'(I4)') Z
      CALL WRVAL(IUCIF, '_cell_formula_units_Z',text)


C Spherical harmonic ODF
      IODF = 0
      IF (.NOT. ONEBLOCK) THEN
        KEYVAL = CRSKEY(IPHAS)//'ODF   '
        ISAM = READEXP(IUEXP,KEYVAL,TEXT)
        READ (TEXT,'(3I5)') NORD,NODFCOF,ISAMSYM
        IF (NODFCOF .GT. 0) THEN
          WRITE(IUCIF,'(/A)') '_pd_proc_ls_pref_orient_corr'
          WRITE(IUCIF,'(2A)') ';','   Spherical Harmonic ODF'
          WRITE(IUCIF,'(A,I2,A,I3)')
     1      ' PHASE',I,' spherical harmonic order=',NORD
          IF ( ISAMSYM.EQ.1 ) THEN
            WRITE(IUCIF,'(A)') ' No sample symmetry'
          ELSE IF ( ISAMSYM.EQ.2 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' 2/m (shear texture)'
          ELSE IF ( ISAMSYM.EQ.3 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' mmm (rolling texture)'
          ELSE IF ( ISAMSYM.EQ.0 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' cylindrical (fiber texture)'
          END IF
          NREC = 0
          IF ( NODFCOF.GT.0 ) NREC = (NODFCOF-1)/6+1
          IBEG = 1
          DO IREC=1,NREC
            WRITE(KEYVAL(10:12),'(I2,A)')IREC,'A'
            IFIN = MIN(IBEG+5,NODFCOF)
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ (TEXT,'(6(I4,2I3))') ((INDX(K,J),K=1,3),
     1        J=IBEG,IFIN)
            KEYVAL(12:12) = 'B'
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ (TEXT,'(6(F10.0))') (COFF(K),K=IBEG,IFIN)
            IBEG = IBEG+6
          END DO
          DO J=1,NODFCOF
            WRITE(IUCIF,'(A,3I3,3x,A,F10.4)')
     1        ' Index =',(INDX(K,J),K=1,3),
     1        'Coeff=',COFF(J)
          END DO
          WRITE(IUCIF,'(a/)')  ';'
        END IF
      END IF


C now loop over interatomic distances for this phase
      WRITE(IUCIF,'(/a)')  '# MOLECULAR GEOMETRY'
      WRITE(IUCIF,'(/a5)') 'loop_'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_atom_site_label_1'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_atom_site_label_2'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_distance'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_site_symmetry_1'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_site_symmetry_2'
      WRITE(IUCIF,'(6x,A)') '_geom_bond_publ_flag'
      IDIS = 0
      IF (IUDIS .NE. 0) THEN
        REWIND(IUDIS)
        READ (IUDIS,'(A)')                            ! skip the first record
        KPHAS = 0
        DO WHILE(KPHAS .LE. IPHAS)
 1        READ (IUDIS,'(A1,2I2,2F10.4,7I5)',ERR=1,END=2)
     1     PUBFLG,KPHAS,ITYP,D,STD,I,J,ISYM,IOFF
          IF (KPHAS .EQ. IPHAS .AND. ITYP .EQ. 0 .AND.
     1      (FRACT(I) .NE. 0 .OR. SGFRAC(I) .NE. 0) .AND.
     1      (FRACT(J) .NE. 0 .OR. SGFRAC(J) .NE. 0)) THEN
            IF (STD .LE. 0) STD = -0.0001
            CALL FESD(D,STD, text, ln)
            DO K=1,NOFFSET
              IF (ISYM .EQ. OFFSYMID(K)) THEN
                IOFF = OFFSET(K) + IOFF
                  GOTO 3
              END IF
            END DO
 3          CONTINUE
            CALL UPCASE(PUBFLG)
            IF (PUBFLG .NE. 'Y') THEN
              PUBFLG = 'N'
            END IF
            WRITE (IUCIF,'(2(2x,A),2x,A16,2x,A1,2X,I5,A1,I3,2x,A1)')
     1        ATMNAM(I),ATMNAM(J),text(:ln),'.',ISYM,'_',IOFF,PUBFLG
            IDIS = IDIS + 1
          END IF
        END DO
 2        CONTINUE
      END IF
      IF (IDIS .EQ. 0) WRITE(IUCIF,'(A)') '   ?   ?   ?   ?   ?   ?'

C now loop over interatomic angles for this phase
      WRITE(IUCIF,'(/a5)') 'loop_'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_atom_site_label_1'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_atom_site_label_2'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_atom_site_label_3'
      WRITE(IUCIF,'(6x,A)') '_geom_angle'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_site_symmetry_1'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_site_symmetry_2'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_site_symmetry_3'
      WRITE(IUCIF,'(6x,A)') '_geom_angle_publ_flag'
      IANG = 0
      IF (IUDIS .NE. 0) THEN
        REWIND(IUDIS)
        READ (IUDIS,'(A)')                            ! skip the first record
        KPHAS = 0
        DO WHILE(KPHAS .LE. IPHAS)
 11       READ (IUDIS,'(A1,2I2,2F10.4,7I5)',ERR=11,END=12)
     1      PUBFLG,KPHAS,ITYP,D,STD,I,J,K,ISYM1,IOFF1,ISYM3,IOFF3
          IF (KPHAS .EQ. IPHAS .AND. ITYP .EQ. 1 .AND.
     1      (FRACT(I) .NE. 0 .OR. SGFRAC(I) .NE. 0) .AND.
     1      (FRACT(K) .NE. 0 .OR. SGFRAC(K) .NE. 0) .AND.
     1      (FRACT(J) .NE. 0 .OR. SGFRAC(J) .NE. 0)) THEN
            IF (STD .LE. 0) STD = -0.001
            CALL FESD(D,STD, text, ln)
            DO K1=1,NOFFSET
              IF (ISYM1 .EQ. OFFSYMID(K1)) THEN
                IOFF1 = OFFSET(K) + IOFF1
              END IF
              IF (ISYM3 .EQ. OFFSYMID(K1)) THEN
                IOFF3 = OFFSET(K) + IOFF3
              END IF
            END DO
            CALL UPCASE(PUBFLG)
            IF (PUBFLG .NE. 'Y') THEN
              PUBFLG = 'N'
            END IF
            WRITE (IUCIF,'(3(2x,A),2x,A16,2x,
     1        I5,A1,I3,2x,A1,2X,I5,A1,I3,2X,A1)')
     1        ATMNAM(I),ATMNAM(J),ATMNAM(K),text(:ln),
     1        ISYM1,'_',IOFF1,'.',ISYM3,'_',IOFF3,PUBFLG
            IANG = IANG + 1
          END IF
        END DO
 12     CONTINUE
      END IF
      IF (IANG .EQ. 0)
     1     WRITE(IUCIF,'(A)') '   ?   ?   ?   ?   ?   ?   ?   ?'
      RETURN
      END
```

Subroutine WRPOWDHIST for program GSAS2CIF

This subroutine is used to write parameters and results for each powder histogram to the output CIF file. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE WRPOWDHIST(IUCIF,IUEXP,IUTRM,IHST,HTYP,IUPRF,
     1  LAM2,DAYTIME,ONEBLOCK,EXPRNAME,SAUTHOR)
C write the obs & calc powder diffractogram & reflection list
C***********************************************************************
C

!PSEUDOCODE:

!CALLING ARGUMENTS:

      INTEGER*4     IHST,IUCIF,IUEXP,IUTRM
      INTEGER*4     IUPRF
      CHARACTER*4   HTYP
      REAL*4        LAM2                    !Alpha_2 lambda
      CHARACTER*20  DAYTIME
      LOGICAL*4     ONEBLOCK                ! true if the CIF will have one block
      CHARACTER*20  EXPRNAME                !Experiment name = name of data block
      CHARACTER*24  SAUTHOR                 !A shortened version of the author name

!INCLUDE STATEMENTS:
      INCLUDE       '../INCLDS/ARRAYSZE.FOR'
      INCLUDE       '../INCLDS/SPGCOMI.FOR'
      INCLUDE       '../INCLDS/DISAGLCM.FOR'

!LOCAL VARIABLES:

      INTEGER*4     NPHAS(9)                !Phase existance flags
      INTEGER*4     NPHASES                 ! number of phases in histogram
      REAL*4        LAM1,ZERO,POLA,DIFC,DIFA
      LOGICAL*4     MOREOBS                 ! true if there are more OBS points than calc
      LOGICAL*4     FIXEDSTEP               ! true for fixed step data
      LOGICAL*4     FIXEDBKG                ! true if fixed background points are used
      LOGICAL*4     NEEDESD                 ! true if the ESD's are not SQRT(I)
      REAL*4        DYDBK(99)
      CHARACTER*80  TEXT
      CHARACTER*80  TEXT1
      CHARACTER*8   TYP
      CHARACTER*20  BUFFER(10)
      INTEGER*4     BUFLEN(10)
      REAL*4        VALUE                   !General use value
      REAL*4        FIRSTPT,LASTPT          !Data range
      REAL*4        STEPMIN,STEPMAX,STEP     !Min, Max & avg step size
      INTEGER*4     OFFSET                  !No. channels to be omitted at start of
profile
      INTEGER*4     ICLMP                   !Data compression factor
      INTEGER*4     CHEKHST                 !Check sum of this histogram
      INTEGER*4     ISAMP                   !Data samping factor
      INTEGER*4     BAKTYP                  !Background function number
      INTEGER*4     NUMBAK              !Background number of terms
      REAL*4        CONC(MAXELEM)
      INTEGER*4     JMLT(MAXATM)        !Atom site multiplicities
      CHARACTER*12  KEYVAL              !ISAM key
      REAL*4        ATWT                !Atomic weight
      REAL*4        BLEN                !Neutron scattering length
      REAL*4        FFAC(9)             !Xray form factor
      REAL*4        FFAN(2,5)           !Xray dispersion terms
      REAL*4        ABSCO(7)            !Absorption coefficients
      REAL*4        MFAC(9)             ! Neutron magnetic form factor
      REAL*4        NFAC(9)             ! Neutron magnetic form factor
      REAL*4        BACKCOF(MAXBAK)     !background coeffients
      INTEGER*4     PRETRM              !No. terms before diffuse terms in #9
      INTEGER*4     TRMTYP(12)          !Diffuse term types
      CHARACTER*1   IAB                 !Absorption refinement flag
      REAL*4        PHKL(3),RATIO,FRAC  !M-D pref. orient.
      REAL*4        COFF(MAXODF)        !Spherical harmonic coefficients
      INTEGER*4     INDX(3,MAXODF)      !Sph. harmonic index
      INTEGER*4     ISAMSYM             !Sample symmetry number (1-4)
      REAL*4        PAXIS(3)            !Aniso. broadening axis
      CHARACTER*1   SAXIS               !=Y if stacking fault model is needed
      REAL*4        UAXIS(3),VAXIS(3)   !Stacking fault subcell vectors
      INTEGER*4     PTYP,NPRF,PTA,PTB   !Profile type & no. of coefficients
      REAL*4        PCOF(36)            !Profile coefficients
      REAL*4        CTOF                !Peak cutoff

!FUNCTION DEFINITIONS:

      INTEGER*4     READEXP             !ISAM file read function
      CHARACTER*6   HSTKEY              !ISAM key building routine
      CHARACTER*6   HAPKEY              !ISAM key building routine
      CHARACTER*6   CRSKEY              !ISAM key building routine
      INTEGER*4     READPRF
      LOGICAL*4     BTEST
      INTEGER*4     LENCH               !LENGTH of a character string


      call OPNPRF(IUEXP,IHST,NCHAN,'SHARED',.FALSE.,IUPRF)
      ISAM = READEXP(IUEXP,HSTKEY(IHST)//' NPHAS',TEXT)
      READ (TEXT,'(9I5)') NPHAS
      NPHASES = 0
      do i=1,9
        IF ( NPHAS(I).NE.0 ) THEN
          NPHASES = NPHASES + 1
          IPHAS = I
        END IF
      END DO


C=======================================================================
C prepare for March-Dollase Preferred Orientation correction
C do any phases have a M-D correction?
      IMD = 0
      DO I = 1,9
        IF ( NPHAS(I).NE.0 ) THEN
```

```
          KEYVAL = HAPKEY(I,IHST)//'NAXIS '
          IERR = READEXP(IUEXP,KEYVAL,TEXT)
          READ(TEXT,'(I5)') NAXIS
          JAX = 0
          NUMF = 0
          DO IAX=1,NAXIS
            JAX = JAX+1
            KEYVAL = HAPKEY(I,IHST)//'PREFO'//CHAR(48+JAX)
            IERR = READEXP(IUEXP,KEYVAL,TEXT)
            IF ( IERR.EQ.0 ) THEN
              READ (TEXT,'(5F10.0)') RATIO,FRAC,(PHKL(K),K=1,3)
              IF (RATIO .NE. 1) IMD = 1
            END IF
          END DO
        END IF
      END DO
C in the single-block case, need to also check for a spherical harmonic
      IODF = 0
      IF (ONEBLOCK) THEN
        DO I = 1,9
          IF ( NPHAS(I).NE.0 ) THEN
            KEYVAL = CRSKEY(I)//'ODF   '
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ (TEXT,'(2I5)') NORD,NODFCOF
            IF (NODFCOF .GT. 0) IODF = 1
          END IF
        END DO
      END IF

C====================================================================

      IF (.NOT. ONEBLOCK) THEN
        WRITE(IUCIF,'(A)') '# phase table'
        WRITE(IUCIF,'(A)') 'loop_     _pd_phase_id'
        WRITE(IUCIF,'(10X,A)') '_pd_phase_block_id'
        WRITE(IUCIF,'(10x,A)') '_pd_phase_mass_%'
        IF (IMD .GT. 0) WRITE(IUCIF,'(10X,A)')
     1    '_pd_proc_ls_pref_orient_corr'
        WRITE(IUCIF,'(10X,A)') '_pd_proc_ls_profile_function'
        WRITE(IUCIF,'(10X,A)') '_pd_proc_ls_peak_cutoff'
        DO I=1,9
          IF ( NPHAS(I).NE.0 ) THEN
C_pd_phase_block_id
            WRITE(IUCIF,'(2X,I1,2X,2A,I1,4A)') I,
     1        DAYTIME(1:16)//'|',
     1        EXPRNAME(1:LENCH(EXPRNAME))//'_phase',I,'|',
     1        SAUTHOR(:LENCH(SAUTHOR)),'||'
C_pd_phase_mass_% (from phase fractions)
            TEXT = ' '
            KEYVAL = HAPKEY(I,IHST)//'MASSFR'
            IERR = READEXP(IUEXP,KEYVAL,TEXT)
            IF (TEXT .NE. ' ') THEN
              READ(TEXT,'(2F10.4)') WTFR,SIGW
              CALL FESD(WTFR*100., SIGW*100., TEXT, LN)
              WRITE (IUCIF,'(10x,A)') TEXT(:LN)
            ELSE
              WRITE (IUCIF,'(10x,A)') '?'
            ENDIF
C_pd_proc_ls_pref_orient_corr
            IF (IMD .GT. 0) THEN
              WRITE(IUCIF,'(2A)') ';','   March-Dollase'
              TEXT = ' '
              KEYVAL = HAPKEY(I,IHST)//'NAXIS '
              IERR = READEXP(IUEXP,KEYVAL,TEXT)
              READ(TEXT,'(I5)') NAXIS
              JAX = 0
              NUMF = 0
              DO IAX=1,NAXIS
                JAX = JAX+1
                KEYVAL = HAPKEY(I,IHST)//'PREFO'//CHAR(48+JAX)
                IERR = READEXP(IUEXP,KEYVAL,TEXT)
                IF ( IERR.EQ.0 ) THEN
                  READ (TEXT,'(5F10.0)') RATIO,FRAC,
     1              (PHKL(K),K=1,3)
                  IF (NAXIS .EQ. 1) THEN
                    WRITE(IUCIF,'(A,I2,A,F10.5,3(2x,A,F6.3))')
     1                ' AXIS ',IAX,' Ratio=',RATIO,
     1                'h=',PHKL(1),'k=',PHKL(2),'l=',PHKL(3)
                  ELSE
                    WRITE(IUCIF,'(A,I2,2(A,F10.3),3(2x,A,F6.3))')
     1                ' AXIS ',IAX,
     1                ' Ratio=',RATIO,' Frac',FRAC,
     1                'h=',PHKL(1),'k=',PHKL(2),'l=',PHKL(3)
                  END IF
                END IF
              END DO
              WRITE(IUCIF,'(A))') ';'
            END IF
C_pd_proc_ls_profile_function
            WRITE(IUCIF,'(A))') ';'
            CALL SYMMINP(IUEXP,I,ICEN,NSYM,LCENT,NCV,LAUE,NPOL,
     1        NAXIS,JRT,CEN,SPG)
            KEYVAL = HAPKEY(I,IHST)//'PRCF  '
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ(TEXT,'(2I5,F10.5,I5)') PTYP,NPRF,CTOF,IDAMP
            DO K=1,36
              PCOF(K) = 0.0
            END DO
            NREC = (MPRF-1)/4+1
            DO IREC=1,NREC
              WRITE(KEYVAL(12:12),'(I1)') IREC
              ISAM = READEXP(IUEXP,KEYVAL,TEXT)
              IBEG = (IREC-1)*4+1
              READ(TEXT,'(4E15.6)') (PCOF(K),K=IBEG,IBEG+3)
            END DO
C now for some pain... list the profile terms for all of Bob's masterpieces
            CALL LISTPRF(IUCIF,NPRF,PTYP,PCOF,LAUE,NAXIS,HTYP,CTOF)
            KEYVAL = CRSKEY(I)//'SPAXIS'
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            IF ( ISAM.EQ.0 ) THEN
```

```
              READ(TEXT,'(3F5.0,4X,A,6F5.0)') PAXIS,SAXIS,UAXIS,VAXIS
              IF ( SAXIS.EQ.'Y' ) THEN
                WRITE(IUCIF,3) PAXIS,UAXIS,VAXIS
 3              FORMAT('   Stacking fault sublattice vectors:',/,
     1            10x,2(3F6.1,';'),3F6.1)
              ELSE
                WRITE(IUCIF,'(A,3F6.1)')
     1            '   Aniso. broadening axis',PAXIS
              END IF
            END IF
            WRITE(IUCIF,'(A)') ';'
C_pd_proc_ls_peak_cutoff
            WRITE(IUCIF,'(10X,F8.5)') CTOF
          END IF
        END DO
      END IF



C====================================================================
C loop over atom types and report the scattering factor info
C include atom amounts, if one histogram and one phase (note that phase info
C was loaded in WRITEPHASE)
      IF (ONEBLOCK) THEN
C determine unit cell contents
        ISAM = READEXP(IUEXP,' EXPR  NATYP',TEXT)
        READ (TEXT,'(9I5)') NELEM
        DO I=1,NELEM
          conc(i) = 0.0
        END DO
        DO I=1,NATOM
          CALL SYTSYM (XYZ(1,I),ICEN,NSYM,JRT,NCV,CEN,LAUE,         !Get site
symmetries and multiplicities
     1      JMLT(I),JSYM,IOPRTNS,STSYM(I))
          J = ID(I)                            !Get the atom type count flag
          conc(J) = conc(j) + JMLT(I)*FRACT(I)
        END DO
      END IF

C loop header
      WRITE(IUCIF,'(/a5,1x,A)') 'loop_', '_atom_type_symbol'
      IF (ONEBLOCK) THEN
        WRITE(IUCIF,'(6x,A)') '_atom_type_number_in_cell'
      END IF
      IRAD = -1
      IF (HTYP(2:2) .eq. 'X') THEN
        WRITE(IUCIF,'(6x,A)') '_atom_type_scat_dispersion_real'
        WRITE(IUCIF,'(6x,A)') '_atom_type_scat_dispersion_imag'
        WRITE(IUCIF,'(6x,''_atom_type_scat_Cromer_Mann_'',A)') 'a1',
     1    'a2','a3', 'a4', 'b1', 'b2', 'b3', 'b4', 'c'
        ISAM = READEXP(IUEXP,HSTKEY(IHST)//' IRAD ',TEXT)
        READ (TEXT,'(I5)') IRAD
      ELSEIF (HTYP(2:2) .eq. 'N') THEN
        WRITE(IUCIF,'(6x,A)') '_atom_type_scat_length_neutron'
      END IF
      WRITE(IUCIF,'(6x,A)') '_atom_type_scat_source'
C _atom_type_description
      ISAM = READEXP(IUEXP,' EXPR  NATYP',TEXT)
      READ (TEXT,'(9I5)') NELEM
      IF (NELEM .LE. 0) THEN
        PRINT '(A)','Error -- This experiment contains no atom types.'
        STOP 'EXPR  NATYP Error'
      END IF
      DO j=1,NELEM
        CALL VSTRNG(ATYPE(j),LENCH(ATYPE(j)),.true.,.false.)
        text = ATYPE(j)(1:8)
        ln = 10
        IF (ONEBLOCK) THEN
          CALL FESD(conc(j), -0.01, text(ln:), ln)
          ln = ln + 2
        END IF
        CALL RDTYPDT(IUEXP,ATYPE(j),ATWT,BLEN,FFAC,FFAN,ABSCO,MFAC,
     1    NFAC,GFAC)
        IF (HTYP(2:2) .eq. 'X') THEN
          IF (IRAD .GE. 1 .and. IRAD .LE. 5) THEN
            write(IUCIF,'(2x,a,2f9.3)') text(:LENCH(text)),
     1        (FFAN(I,IRAD),I=1,2)
          ELSE IF (IRAD .EQ. 0) THEN
            FF1 = 0
            FF2 = 0
            IREC = 0
            ISAM = 0
C     loop through the anomolous f' & f'' for values
            DO WHILE (ISAM .EQ. 0 .AND. IREC .LT. 9)
              IREC = IREC + 1
              KEYVAL = HSTKEY(IHST)//'FFANS '
              WRITE(KEYVAL(12:12),'(I1)') IREC
              ISAM = READEXP(IUEXP,KEYVAL,TEXT1)
              IF ( ISAM.EQ.0 ) THEN
                READ(TEXT1,'(2X,A8,2F10.0)') TYP,FF1A,FF2A
                IF (TYP .EQ. ATYPE(j)) THEN
                  FF1 = FF1A
                  FF2 = FF2A
                  ISAM = -1
                END IF
              END IF
            END DO
            write(IUCIF,'(2x,a,2f9.3)') text(:LENCH(text)),FF1,FF2
          END IF
C     now write the coeff. for the scattering curve
          LN = 1
          TEXT = ' '
          DO I=1,9
            IF (FFAC(I) .GE. 10000 .OR. FFAC(I) .LE. -1000) THEN
              WRITE(TEXT(LN:),'(F8.1)') FFAC(I)
            ELSEIF (FFAC(I) .GE. 1000 .OR. FFAC(I) .LE. -100) THEN
              WRITE(TEXT(LN:),'(F8.2)') FFAC(I)
            ELSEIF (FFAC(I) .GE. 100 .OR. FFAC(I) .LE. -10) THEN
              WRITE(TEXT(LN:),'(F8.3)') FFAC(I)
            ELSEIF (FFAC(I) .GE. 10 .OR. FFAC(I) .LE. 0) THEN
              WRITE(TEXT(LN:),'(F8.4)') FFAC(I)
            ELSE
              WRITE(TEXT(LN:),'(F8.5)') FFAC(I)
            ENDIF
            LN = LN + 8
          END DO
          WRITE(IUCIF,'(A)') TEXT(:LN)
        ELSE IF (HTYP(2:2) .eq. 'N') THEN
          write(IUCIF,'(2x,a,1x,F8.4)') text(:LENCH(text)),BLEN
        END IF
C_atom_type_scat_source
        write(IUCIF,'(2x,a)') 'International_Tables_Vol_C'
      END DO
C====================================================================

      IF (HTYP(2:2) .eq. 'X') THEN
        CALL WRVAL(IUCIF,'_diffrn_radiation_probe','x-ray')
      ELSE IF (HTYP(2:2) .eq. 'N') THEN
        CALL WRVAL(IUCIF,'_diffrn_radiation_probe','neutron')
      ELSE
        PRINT '(A)','Unexpected data type for _diffrn_radiation'//
     1    '_probe histogram #',IHST
        CALL WRVAL(IUCIF,'_diffrn_radiation_probe','?')
      END IF

      ISAM = readexp(IUEXP, HSTKEY(IHST)//' CHANS',text)
      READ(TEXT,'(20x,I10,10x,i10)') nchans,mchans
      ISAM = READEXP(IUEXP, HSTKEY(IHST)//' ICONS',TEXT)
      IF (HTYP(3:3) .eq. 'T') THEN
        READ(TEXT,'(3F10.0)') DIFC,DIFA,ZERO
      ELSE IF (HTYP(2:2) .eq. 'N') THEN
        READ (TEXT,'(3f10.0,25x,f10.0)') lam1,lam2,zero,ratio
        CALL FESD(LAM1, -.00001, text, ln)
        CALL WRVAL(IUCIF, '_diffrn_radiation_wavelength' ,text)
      ELSE
        READ (TEXT,'(3f10.0,10x,f10.0,i5,f10.0)') lam1,lam2,zero,
     1    pola,ipola,ratio
C Bob -- I don't know how to treat case of IPOLA != 0
        IF (IPOLA .EQ. 0) THEN
          CALL FESD(POLA, -.01, text, ln)
          CALL WRVAL(IUCIF, '_diffrn_radiation_polarisn_ratio' ,text)
        ELSE
          CALL WRVAL(IUCIF, '_diffrn_radiation_polarisn_ratio' ,'?')
        END IF
        IF (LAM2 .eq. 0.0) then
          CALL FESD(LAM1, -.00001, text, ln)
          CALL WRVAL(IUCIF, '_diffrn_radiation_wavelength' ,text)
          CALL WRVAL(IUCIF, '_diffrn_radiation_type', '?')
        ELSE
C always assume Ka1 & Ka2 if two wavelengths are present
          WRITE(IUCIF,'(/a)') 'loop_'
          WRITE(IUCIF,'(6x,A)') '_diffrn_radiation_wavelength'
          WRITE(IUCIF,'(6x,A)') '_diffrn_radiation_wavelength_wt'
          WRITE(IUCIF,'(6x,A)') '_diffrn_radiation_type'
          WRITE(IUCIF,'(6x,A)') '_diffrn_radiation_wavelength_id'
          WRITE(IUCIF,'(20x,f10.6,5x,f6.3,3x,A,i3)')
     1      LAM1,1.0,'K\\a~1~',1,
     1      LAM2,ratio,'K\\a~2~',2
        END IF
      END IF


      TEXT = '         ?         ?'
      ISAM = READEXP(IUEXP,HSTKEY(IHST)//' RPOWD',TEXT)
      CALL WRVAL(IUCIF,'_pd_proc_ls_prof_R_factor',TEXT(11:20))
      CALL WRVAL(IUCIF,'_pd_proc_ls_prof_wR_factor',TEXT(1:10))
      TEXT = '         ?'
      ISAM = READEXP(IUEXP,HSTKEY(IHST)//' WREXP',TEXT)
      CALL WRVAL(IUCIF,'_pd_proc_ls_prof_wR_expected',TEXT(1:10))

      TEXT = '         ?'
      ISAM = READEXP(IUEXP,HSTKEY(IHST)//' R-FAC',TEXT)
      CALL WRVAL(IUCIF,'_refine_ls_R_Fsqd_factor',TEXT(6:15))


C document the background function used
      KEYVAL = HSTKEY(IHST)//' TRNGE'
      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
      IF ( ISAM.NE.0 ) TEXT = '       1.0     100.0'
      READ(TEXT,'(2F10.0)') TTMIN,TTMAX
      KEYVAL = HSTKEY(IHST)//'BAKGD '
      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
      IF (ISAM .EQ. 0) THEN
        READ(TEXT,'(2I5,15X,I5,12I1)') BAKTYP,NUMBAK,PRETRM,TRMTYP
        WRITE(IUCIF,'(/a)') '_pd_proc_ls_background_function'
        WRITE(IUCIF,'(2a,I2,a,I3,a)') ';',
     1    '   GSAS Background function number',BAKTYP,
     1    ' with',NUMBAK,' terms.'
        IF ( BAKTYP.EQ.1 ) THEN
          WRITE(IUCIF,'(A)') ' Shifted Chebyshev function of 1st kind'
        ELSE IF ( BAKTYP.EQ.2 ) THEN
          WRITE(IUCIF,'(A)') ' Cosine Fourier series'
        ELSE IF ( BAKTYP.EQ.3 ) THEN
          WRITE(IUCIF,'(A)') ' Real space distribution function'
        ELSE IF ( BAKTYP.EQ.4 ) THEN
          WRITE(IUCIF,'(A)') ' Power series in Q**2n/n!'
        ELSE IF ( BAKTYP.EQ.5 ) THEN
          WRITE(IUCIF,'(A)') ' Power series in n!/Q**2n'
        ELSE IF ( BAKTYP.EQ.6 ) THEN
          WRITE(IUCIF,'(A)') ' Power series in Q**2n/n! and n!/Q**2n'
        ELSE IF ( BAKTYP.EQ.7 ) THEN
          WRITE(IUCIF,'(A)') ' Linear interpolation'
        ELSE IF ( BAKTYP.EQ.8 ) THEN
          WRITE(IUCIF,'(A)') ' Reciprocal interpolation'
        ELSE IF ( BAKTYP.EQ.9 ) THEN
          WRITE(IUCIF,'(A)') ' Diffuse scattering function'
        END IF
        NREC = (NUMBAK-1)/4+1
        IBEG = 1
        DO IREC=1,NREC
          WRITE(KEYVAL(12:12),'(I1)')IREC
          IFIN = MIN(IBEG+3,NUMBAK)
          ISAM = READEXP(IUEXP,KEYVAL,TEXT)
          READ (TEXT,'(4E15.6)') (BACKCOF(I),I=IBEG,IFIN)
          WRITE(IUCIF,'(5x,4(I2,A,1PG15.6))')
     1      (I,':',BACKCOF(I),I=IBEG,IFIN)
          IBEG = IBEG+4
        END DO
        WRITE(IUCIF,'(a)') ';'
      END IF

C handle absorption/roughness correction
      KEYVAL = HSTKEY(IHST)//'ABSCOR'
      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
      IF (ISAM .EQ. 0 .AND.  TEXT(20:20).EQ.'.' ) THEN          !Ignore old record 
format
        READ(TEXT,'(2E15.6,4X,A,2I5)') ABSCOR1,ABSCOR2,IAB,
     1    IDAMP,IABTYP
        WRITE(IUCIF,'(/a)') '_exptl_absorpt_process_details'
        WRITE(IUCIF,'(2a,I2)') ';   GSAS Absorption/surface roughness',
     1    ' correction: function number',IABTYP
        IF ( IABTYP.EQ.0 .AND. ABSCOR1 .EQ. 0) THEN
          WRITE(IUCIF,'(A)') ' No correction is applied.'
        ELSE IF ( IABTYP.EQ.0 ) THEN
          WRITE(IUCIF,'(A)') ' Debye-Scherrer absorption correction'
          WRITE(IUCIF,'(A,G15.5)') 'Term (= MU.r/wave) = ',ABSCOR1
        ELSE IF ( IABTYP.EQ.1 ) THEN
          WRITE(IUCIF,'(A)') ' Linear absorption correction'
          WRITE(IUCIF,'(A,2G15.5)') 'Term = ',ABSCOR1
        ELSE IF ( IABTYP.EQ.2 ) THEN
          WRITE(IUCIF,'(A)') ' Surface roughness abs. correction'//
     1      ' (Pitschke, et al.)'
          WRITE(IUCIF,'(A,2G15.5)') 'Terms = ',ABSCOR1,ABSCOR2
        ELSE IF ( IABTYP.EQ.3 ) THEN
          WRITE(IUCIF,'(A)') ' Surface roughness abs. correction'//
     1      ' (Suortti)'
          WRITE(IUCIF,'(A,2G15.5)') 'Terms = ',ABSCOR1,ABSCOR2
        ELSE IF ( IABTYP.EQ.4 ) THEN
          WRITE(IUCIF,'(A)') ' Flat plate transmission absorption'//
     1      ' correction'
          WRITE(IUCIF,'(A,2G15.5)') 'Terms = ',ABSCOR1,ABSCOR2
        END IF
        IF (IAB .EQ. 'Y') THEN
          WRITE(IUCIF,'(A,2G15.5)') 'Correction is refined.'
        ELSE IF (IABTYP.NE.0 .OR. ABSCOR1 .NE. 0) THEN
          WRITE(IUCIF,'(A,2G15.5)') 'Correction is not refined.'
        END IF
        WRITE(IUCIF,'(a)') ';'
      END IF
C probably not needed
C    _exptl_absorpt_correction_type      'shelx76 gaussian'
C not exactly appropriate
```

```
C    _exptl_absorpt_coefficient_mu       0.59 (unknown in GSAS? -- empirical?)
C    _pd_char_atten_coef_mu_obs

C show range of applied corrections
C absorption
      TEXT = ' '
      ISAM = READEXP(IUEXP, HSTKEY(IHST)//'TRMNMX',TEXT)
      IF (TEXT .NE. ' ') THEN
        CALL WRVAL(IUCIF,'_exptl_absorpt_correction_T_min',TEXT(1:10))
        CALL WRVAL(IUCIF,'_exptl_absorpt_correction_T_max',TEXT(11:20))
      ELSE
        CALL WRVAL(IUCIF,'_exptl_absorpt_correction_T_min','?')
        CALL WRVAL(IUCIF,'_exptl_absorpt_correction_T_max','?')
      ENDIF
C extinction
      TEXT = ' '
      ISAM = READEXP(IUEXP, HSTKEY(IHST)//'EXMNMX',TEXT)
      IF (TEXT .NE. ' ') THEN
        WRITE(IUCIF,'(A)') '# Extinction correction'
        CALL WRVAL(IUCIF,'_gsas_exptl_extinct_corr_T_min',TEXT(1:10))
        CALL WRVAL(IUCIF,'_gsas_exptl_extinct_corr_T_max',TEXT(11:20))
      ENDIF


      IF (ONEBLOCK .AND. IMD+IODF .GT. 0) THEN
        I = IPHAS
        WRITE(IUCIF,'(A)') '_pd_proc_ls_pref_orient_corr'
        IF (IMD .GT. 0) THEN
          WRITE(IUCIF,'(2A)') ';','   March-Dollase'
        ELSE
          WRITE(IUCIF,'(2A)') ';','   Spherical Harmonic ODF'
        END IF
        IF (IMD .GT. 0) THEN
          TEXT = ' '
          KEYVAL = HAPKEY(I,IHST)//'NAXIS '
          IERR = READEXP(IUEXP,KEYVAL,TEXT)
          READ(TEXT,'(I5)') NAXIS
          JAX = 0
          NUMF = 0
          DO IAX=1,NAXIS
            JAX = JAX+1
            KEYVAL = HAPKEY(I,IHST)//'PREFO'//CHAR(48+JAX)
            IERR = READEXP(IUEXP,KEYVAL,TEXT)
            IF ( IERR.EQ.0 ) THEN
              READ (TEXT,'(5F10.0)') RATIO,FRAC,
     1          (PHKL(K),K=1,3)
              IF (NAXIS .EQ. 1) THEN
                WRITE(IUCIF,'(A,I2,A,F10.5,3(2x,A,F6.3))')
     1            ' AXIS ',IAX,' Ratio=',RATIO,
     1            'h=',PHKL(1),'k=',PHKL(2),'l=',PHKL(3)
              ELSE
                WRITE(IUCIF,'(A,I2,2(A,F10.3),3(2x,A,F6.3))')
     1            ' AXIS ',IAX,
     1            ' Ratio=',RATIO,' Frac',FRAC,
     1            'h=',PHKL(1),'k=',PHKL(2),'l=',PHKL(3)
              END IF
            END IF
          END DO
        END IF
        IF (IODF .GT. 0 .and. IMD .NE. 0) THEN
          WRITE(IUCIF,'(/A)') ' **** Spherical Harmonic ODF ****'
        END IF
        IF (IODF .GT. 0) THEN
          KEYVAL = CRSKEY(I)//'ODF   '
          ISAM = READEXP(IUEXP,KEYVAL,TEXT)
          READ (TEXT,'(3I5)') NORD,NODFCOF,ISAMSYM
          WRITE(IUCIF,'(A,I3)')
     1      ' Spherical harmonic order=',NORD
          IF ( ISAMSYM.EQ.1 ) THEN
            WRITE(IUCIF,'(A)') ' No sample symmetry'
          ELSE IF ( ISAMSYM.EQ.2 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' 2/m (shear texture)'
          ELSE IF ( ISAMSYM.EQ.3 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' mmm (rolling texture)'
          ELSE IF ( ISAMSYM.EQ.0 ) THEN
            WRITE(IUCIF,'(2A)') ' The sample symmetry is:',
     1        ' cylindrical (fiber texture)'
          END IF
          NREC = 0
          IF ( NODFCOF.GT.0 ) NREC = (NODFCOF-1)/6+1
          IBEG = 1
          DO IREC=1,NREC
            WRITE(KEYVAL(10:12),'(I2,A)')IREC,'A'
            IFIN = MIN(IBEG+5,NODFCOF)
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ (TEXT,'(6(I4,2I3))') ((INDX(K,J),K=1,3),
     1        J=IBEG,IFIN)
            KEYVAL(12:12) = 'B'
            ISAM = READEXP(IUEXP,KEYVAL,TEXT)
            READ (TEXT,'(6(F10.0))') (COFF(K),K=IBEG,IFIN)
            IBEG = IBEG+6
          END DO
          DO J=1,NODFCOF
            WRITE(IUCIF,'(A,3I3,3x,A,F10.4)')
     1        ' Index =',(INDX(K,J),K=1,3),
     1        'Coeff=',COFF(J)
          END DO
        END IF
        TEXT = ' '
        ISAM = READEXP(IUEXP, HSTKEY(IHST)//'ODMNMX',TEXT)
        IF (TEXT .NE. ' ') THEN
           WRITE(IUCIF,'(4A)')
     1          ' Prefered orientation correction range: Min=',
     1          TEXT(1:10),', Max=',TEXT(11:20)
        ENDIF
        WRITE(IUCIF,'(A)') ';'
      END IF
```

```
C_pd_proc_ls_profile_function
      IF (ONEBLOCK) THEN
        I = IPHAS
        WRITE(IUCIF,'(2(/,A))') '_pd_proc_ls_profile_function',';'
        CALL SYMMINP(IUEXP,I,ICEN,NSYM,LCENT,NCV,LAUE,NPOL,
     1    NAXIS,JRT,CEN,SPG)
        KEYVAL = HAPKEY(I,IHST)//'PRCF  '
        ISAM = READEXP(IUEXP,KEYVAL,TEXT)
        READ(TEXT,'(2I5,F10.5,I5)') PTYP,NPRF,CTOF,IDAMP
        DO K=1,36
          PCOF(K) = 0.0
        END DO
        NREC = (MPRF-1)/4+1
        DO IREC=1,NREC
          WRITE(KEYVAL(12:12),'(I1)') IREC
          ISAM = READEXP(IUEXP,KEYVAL,TEXT)
          IBEG = (IREC-1)*4+1
          READ(TEXT,'(4E15.6)') (PCOF(K),K=IBEG,IBEG+3)
        END DO
C now for some pain... list the profile terms for all of Bob's masterpieces
        CALL LISTPRF(IUCIF,NPRF,PTYP,PCOF,LAUE,NAXIS,HTYP,CTOF)
        KEYVAL = CRSKEY(I)//'SPAXIS'
        ISAM = READEXP(IUEXP,KEYVAL,TEXT)
        IF ( ISAM.EQ.0 ) THEN
          READ(TEXT,'(3F5.0,4X,A,6F5.0)') PAXIS,SAXIS,UAXIS,VAXIS
          IF ( SAXIS.EQ.'Y' ) THEN
            WRITE(IUCIF,3) PAXIS,UAXIS,VAXIS
          ELSE
            WRITE(IUCIF,'(A,3F6.1)')
     1        '   Aniso. broadening axis',PAXIS
          END IF
        END IF
        WRITE(IUCIF,'(A)') ';'
        WRITE(IUCIF,'(A,F8.5)') '_pd_proc_ls_peak_cutoff',CTOF
      END IF

C use current time/date here
      CALL WRVAL(IUCIF, '_pd_proc_info_datetime', daytime)
      CALL WRVAL(IUCIF, '_pd_calc_method', 'Rietveld Refinement')


      DO I=1,10
        BUFLEN(I) = 0
      END DO
C put the intensity data on a scratch file, so that we can find the length
C of each column; then we can line up numbers so they look pretty
      CALL GETUNIT(IUSCRT)
      OPEN(IUSCRT)

C is this time-of-flight?
      IF (HTYP(3:3) .eq. 'T') THEN
```

```
        FIXEDSTEP = .false.                        ! true for fixed step data
        NEEDESD = .true.
      ELSE
C check through the data to check if the step size is fixed and
C get the starting, ending angles & step angles/channel while doing this
        J = 1
        IREC = 0
        STEPMIN = 0
        STEPMAX = 0
        LASTPT = -1
C are the intensity values scaled? Assume no & scan through the histogram
        NEEDESD = .false.
        do while (J .ne. 0)
          IREC = IREC + 1
          J = READPRF(IUPRF,IREC,ICODE,FIRSTPT,YO,YC,YI,YB,YW,CHWDT,
     1      MIN1,MIN2)
        END DO
        K = 1
        IF (YW .GT. 0 .AND.
     1    (YO*YW .LT. .95 .OR. YO*YW .GT. 1.05))
     1    NEEDESD = .true.
        ISAM = readexp(IUEXP, HSTKEY(IHST)//' CHANS',text)
        IF ( ISAM.EQ.0 ) READ(text,'(5I10,I5)') OFFSET,ICLMP,
     1    NCHANS,CHEKHST,MCHANS,ISAMP
        IF ( ISAMP.EQ.0 ) ISAMP = 1
        DO I = IREC+1,NCHANS
          J = READPRF(IUPRF,I,ICODE,T2,YO,YC,YI,YB,YW,CHWDT,MIN1,MIN2)
          if (j .eq. 0) then
            IF (YW .GT. 0 .AND.
     1        (YO*YW .LT. .95 .OR. YO*YW .GT. 1.05))
     1        NEEDESD = .true.
C     Is this the second defined point?
            IF (LASTPT .EQ. -1) THEN
              STEPMIN = ABS(T2 - FIRSTPT)
              STEPMAX = ABS(T2 - FIRSTPT)
            ELSE
              STEPMIN = MIN(STEPMIN,ABS(T2 - LASTPT))
              STEPMAX = MIN(STEPMAX,ABS(T2 - LASTPT))
            END IF
            irec = I
            LASTPT = t2
            k = k + 1
          END IF
        END DO
C treat a <1% variation in stepsize as fixed step size
        IF ( (STEPMAX-STEPMIN)/STEPMAX .GT. 0.01) THEN
          FIXEDSTEP = .false.
        ELSE
C round step to nearest .001 degree
          STEP = NINT(100.*((LASTPT - FIRSTPT)/(k-1.)))/100.
          FIXEDSTEP = .true.
        END IF
      END IF

C do we have any fixed background points?
```

```
      FIXEDBKG = .false.                  ! true if fixed background points are used
      ISAM = READEXP(IUEXP,HSTKEY(IHST)//'FXB  1',TEXT)
      IF ( ISAM.NE.0 ) FIXEDBKG = .true.


C do we have a different number of experimental data points then in the
C refined histogram? Caused by data compression or sampling
C for now, ignore dropped points at the beginning of the diffraction pattern
      IF (ICLMP .GT. 1 .OR. ISAMP .GT. 1) THEN
        MOREOBS = .true.                ! there are more OBS points than calc
      ELSE
        MOREOBS = .false.               ! same number of OBS & CALC points
      END IF
C***********************************************************************
C loop for raw data (if needed)
      IF (MOREOBS) THEN
        WRITE(IUCIF,'(/,A)') '#---- raw data loop -----'
        CALL WRITERAWDATA(IUEXP,IUCIF,IHST,HTYP,FIXEDSTEP)
        WRITE(IUCIF,'(/,A)') '#---- calculated data loop -----'
      ELSE
        WRITE(IUCIF,'(/,A)') '#---- raw/calc data loop -----'
      END IF
C***********************************************************************

C loop over computed histogram & optionally list the observed as well
      IF (FIXEDSTEP) THEN
C starting, ending angles & step for OBS & CALC -- N.B. Always 2theta
        IF (.not. MOREOBS) then
          CALL FESD(FIRSTPT/100., -abs(STEP/10000.), text,ln)
          CALL WRVAL(IUCIF, '_pd_meas_2theta_range_min', text)
          CALL FESD(LASTPT/100., -abs(STEP/10000.), text,ln)
          CALL WRVAL(IUCIF, '_pd_meas_2theta_range_max', text)
          CALL FESD(STEP/100., -abs(STEP/10000.), text,ln)
          CALL WRVAL(IUCIF, '_pd_meas_2theta_range_inc', text)
        END IF
C zero correct & convert from centidegrees
        FIRSTPT = (FIRSTPT - zero)/100.
        LASTPT = (LASTPT - zero)/100.
        CALL FESD(FIRSTPT, -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_proc_2theta_range_min', text)
        CALL FESD(LASTPT, -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_proc_2theta_range_max', text)
        CALL FESD(STEP/100., -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_proc_2theta_range_inc', text)
      END IF
C***********************************************************************

C write the header for the main data loop
      WRITE(IUCIF,'(/,A)') 'loop_'
      IF (HTYP(3:3) .eq. 'T') THEN
        IF (.not. MOREOBS) WRITE(IUCIF,'(6x,A)')
     1    '_pd_meas_time_of_flight'
        WRITE(IUCIF,'(6x,A)') '_pd_proc_d_spacing'
      ELSE IF (.not. FIXEDSTEP) THEN
```

```
          IF (.not. MOREOBS) WRITE(IUCIF,'(6x,A)') '_pd_meas_2theta_scan'
          IF (MOREOBS .or. ZERO .ne. 0.)
     1      WRITE(IUCIF,'(6x,A)') '_pd_proc_2theta_corrected'
        END IF
C which intensity variable is needed?
        IF (MOREOBS) THEN
          WRITE(IUCIF,'(6x,A)') '_pd_proc_intensity_total'
          NEEDESD = .TRUE.                            ! this requires SU's since _total
is assumed to not be counts
        ELSE IF (NEEDESD) THEN
          WRITE(IUCIF,'(6x,A)') '_pd_meas_intensity_total'
        ELSE
          WRITE(IUCIF,'(6x,A)') '_pd_meas_counts_total'
        END IF
        WRITE(IUCIF,'(6x,A)') '_pd_proc_ls_weight'
        WRITE(IUCIF,'(6x,A)') '_pd_proc_intensity_bkg_calc'
C for now ignore fixed background points
!       IF (FIXBACK) WRITE(IUCIF,'(6x,A)') '_pd_proc_intensity_fix_bkg'
        WRITE(IUCIF,'(6x,A)') '_pd_calc_intensity_total'

        KEYVAL = HSTKEY(IHST)
        IF ( HTYP(3:3).EQ.'T' ) THEN                       !TOF neutron data
          ISAM = READEXP(IUEXP,KEYVAL(1:6)//'BNKPAR',TEXT)
          READ(TEXT,'(10X,F10.0)') WAVE
          DIFC1000 = DIFC/1000.
          ISAM = READEXP(IUEXP,KEYVAL(1:6)//'I ITYP',TEXT)
          READ(TEXT,'(15X,F10.4)') TMAX
          TMAX = 180.0/TMAX
        ELSE IF ( HTYP(3:3).EQ.'C' ) THEN
          WAVE = LAM1
          TMAX = 180.0
        ELSE IF ( HTYP(3:3).EQ.'E' ) THEN
          WAVE = LAM1
          TMAX = 250.0
        END IF


C now loop through the data
        npoint = 0
        DO I=1,nchans
          j = 0
          k = READPRF(IUPRF,I,ICODE,T1,YO,YC,YI,YB,YW,CHWDT,MIN1,MIN2)
          if (k .eq. 0) then
C compute the background
            npoint = npoint + 1
            IF ( HTYP(3:3).EQ.'T' ) THEN             ! TOF
              T1A = T1/1000.
            ELSE IF (HTYP(3:3).EQ.'C') THEN          ! constant wavelength
              T1A = T1/100.
            ELSE IF ( HTYP(3:3).EQ.'E' ) THEN       ! energy dispersive x-ray
              T1A = T1
            END IF
            CALL CALCBAK(HTYP,TMAX,DIFC1000,WAVE,TTMIN,TTMAX,
     1        BAKTYP,NUMBAK,BACKCOF,PRETRM,TRMTYP,T1A,YB1)
C add the fixed and computed background
```

```
          YB1 = YB + YB1
C------------------------------------------------------------------------
C report the scan variable, if TOF or not fixed step
          IF (HTYP(3:3) .eq. 'T') THEN
C _pd_meas_time_of_flight
            IF (.not. MOREOBS) THEN
              J = J + 1
              CALL FESD(t1, -0.01, buffer(j),ln)      !The LANSCE T-O-F clocks run @ 
20M Hz
              BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            END IF
C _pd_proc_d_spacing
            TMP = TOFTOD(HTYP,DIFC,DIFA,ZERO,1.0,T1,VALUE)
            T1 = TMP
            J = J + 1
            LN = -1
            CALL FESD(t1, -t1*.0001, buffer(j),ln)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
          ELSE IF (.not. FIXEDSTEP) THEN
C _pd_meas_2theta_scan
            IF (.not. MOREOBS) THEN
              J = J + 1
              LN = -1
              CALL FESD(t1/100., -0.001, buffer(j),ln)
              BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            END IF
C _pd_proc_2theta_corrected
            IF (MOREOBS .or. ZERO .ne. 0.) THEN
              J = J + 1
              LN = -1
              CALL FESD((t1-ZERO)/100., -0.001, buffer(j),ln)
              BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            END IF
          END IF
C------------------------------------------------------------------------
C intensity info:
          IF (NEEDESD) THEN
C _pd_proc_intensity_total or _pd_meas_intensity_total
            ESD = 0
            IF (YW .gt. 0) ESD = 1./SQRT(YW)
            J = J + 1
            LN = -1
            CALL FESD(YO, ESD, buffer(j),ln)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
          ELSE
C _pd_meas_intensity_counts
            J = J + 1
            LN = -1
            CALL FESD(YO, -1., buffer(j),ln)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
          END IF
C------------------------------------------------------------------------
C _pd_proc_ls_weight
```

```
          J = J + 1
          IF ( BTEST(ICODE,1) ) THEN
            LN = -1
            CALL FESD(0.0, 0.0, buffer(j),ln)
          ELSE
            LN = -1
            CALL FESD(YW, -yw/100., buffer(j),ln)
          END IF
          BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
C-----------------------------------------------------------------------
C _pd_proc_intensity_calc_bkg
          J = J + 1
          LN = -1
          CALL FESD(YB1, -ESD/10., buffer(j),ln)
          BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
!          if (j .ge. 5) then
!             write (IUCIF,'(5(2x,a))')
!      1      (buffer(jj)(:LENCH(buffer(jj))),jj=1,j)
!             j = 0
!          END IF
C for now ignore fixed background points
!      IF (FIXBACK) WRITE(IUCIF,'(6x,A)') '_pd_proc_intensity_fix_bkg'

C-----------------------------------------------------------------------
C _pd_calc_intensity_total
          J = J + 1
          IF(BTEST(ICODE,1)) THEN
            buffer(j) = ' .'        ! undefined Y(calc)
          ELSE
            LN = -1
            CALL FESD(YC, -esd/10., buffer(j),ln)
          END IF
          BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
C-----------------------------------------------------------------------
C write the line to the scratch file
          write (IUSCRT,'(9A)') (buffer(jj),jj=1,j)
          JMAX = J
        END IF
      END DO
      REWIND(IUSCRT)
C copy from the scratch file to the table
      DO I=1,NPOINT
        READ(IUSCRT,'(9A)') (BUFFER(JJ),JJ=1,JMAX)
        TEXT = ' '
        LN = 1
        DO JJ=1,JMAX
          IF (LN + BUFLEN(JJ) .GT. 80) THEN
            WRITE(IUCIF,'(A)') TEXT(:LENCH(TEXT))
            TEXT = ' '
            LN = 5
          END IF
          TEXT(LN+1:) = BUFFER(JJ)(1:BUFLEN(JJ))
          LN = LN + BUFLEN(JJ) + 2
        END DO
```

```
          WRITE(IUCIF,'(A)') TEXT(:LENCH(TEXT))
        END DO
        CLOSE(IUSCRT,STATUS='DELETE')
        IF (.not. MOREOBS) THEN
          write (text,'(I9)') npoint
          CALL WRVAL(IUCIF, '_pd_meas_number_of_points', text)
        END IF
        write (text,'(I9)') npoint
        CALL WRVAL(IUCIF, '_pd_proc_number_of_points', text)
        CLOSE(IUPRF)
        RETURN
        END
```

Subroutine WRREFLIST for program GSAS2CIF

This subroutine is used to write a reflection list for both single crystal and powder diffraction data. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE WRREFLIST(IUEXP,IUCIF,IHST,HTYPE,NPHASES,LAM2,DAYTIME)

      INCLUDE       '../INCLDS/COPYRIGT.FOR'

!Calling arguments:

      INTEGER*4     IUEXP                   !Experiment file unit number
      INTEGER*4     IUCIF                   !CIF file unit number
      INTEGER*4     IHST                    !Histogram number
      CHARACTER*4   HTYPE                   !Histogram type
      INTEGER*4     NPHASES                 !Number of phases present in the experiment
      REAL*4        LAM2                    !2nd wave length in the powder pattern
      CHARACTER*20  DAYTIME

!Local variables:

      INTEGER*4     MINHKL(3)
      INTEGER*4     MAXHKL(3)
      REAL*4        HKL(3)
      REAL*4        INCDNT,DSPACE,LAM,FOSQ,PEAKI,FOTSQ,FCSQ,FCTSQ,
     1      PHAS,TRANS,EXTCOR,PROFLP,TF
      CHARACTER*80  TEXT
      CHARACTER*20  BUFFER(11)
      INTEGER*4     BUFLEN(11)
      LOGICAL*4     PWDR                    !.TRUE. for powder data
      LOGICAL*4     SNGL                    !.TRUE. if single crystal data

!Functions defined:

      CHARACTER*6   HSTKEY
      INTEGER*4     READEXP
      INTEGER*4     HEXTOINT
      INTEGER*4     REDREFP
      INTEGER*4     REDREFS

!Code:

      PWDR = .FALSE.
      SNGL = .FALSE.
      DO I=1,11
        BUFLEN(I) = 0
      END DO
      IF ( HTYPE(1:1).EQ.'P' ) THEN
        PWDR = .TRUE.
      ELSE
        SNGL = .TRUE.
      END IF

      WRITE(IUCIF,'(/a5,1x,A)') 'loop_'
      IF ( PWDR ) THEN
        WRITE(IUCIF,'(6x,A)') '_refln_index_h'
        WRITE(IUCIF,'(6x,A)') '_refln_index_k'
        WRITE(IUCIF,'(6x,A)') '_refln_index_l'
        IF (LAM2 .ne. 0) WRITE(IUCIF,'(6x,A)')
     1    '_pd_refln_wavelength_id'
        IF (NPHASES .GT. 1) WRITE(IUCIF,'(6x,A)') '_pd_refln_phase_id'
      ELSE
        WRITE(IUCIF,'(6x,A)') '_refln_index_h'
        WRITE(IUCIF,'(6x,A)') '_refln_index_k'
        WRITE(IUCIF,'(6x,A)') '_refln_index_l'
      END IF
      WRITE(IUCIF,'(6x,A)') '_refln_observed_status'
      WRITE(IUCIF,'(6x,A)') '_refln_F_squared_meas'
      WRITE(IUCIF,'(6x,A)') '_refln_F_squared_calc'
      WRITE(IUCIF,'(6x,A)') '_refln_phase_calc'
      IF ( PWDR ) THEN
        WRITE(IUCIF,'(6x,A)') '_refln_d_spacing'
        WRITE(IUCIF,'(6x,A)') '_gsas_i100_meas'
        ISAM = READEXP(IUEXP,HSTKEY(IHST)//' BIGFO',TEXT)
        IF ( ISAM.EQ.0 ) THEN
          READ(TEXT,'(F15.0)') BIGFO
          IF ( BIGFO.LE.0.0 ) BIGFO = 1.0
        ELSE
          BIGFO = 1.0
        END IF
        ISAM = readexp(IUEXP,HSTKEY(IHST)//' NREF ',text)
      ELSE
        ISAM = readexp(IUEXP,HSTKEY(IHST)//' NREFM',text)
        IF ( ISAM.NE.0 ) THEN
          ISAM = readexp(IUEXP,HSTKEY(IHST)//' NREF ',text)
        END IF
      END IF
      READ(text,'(I5,F10.0,4X,A1)') NREF,DMIN,IFOBS
      numobs = 0
      do i=1,3
        minhkl(i) = 999
        maxhkl(i) = -999
      END DO
      dmax = 0.
      dmin = 9999.
      NREFI = 100000*IHST
      IF ( HTYPE(1:3).EQ.'PNT' ) NREFI=NREFI+NREF+1
      CALL GETUNIT(IUSCRT)
      OPEN(IUSCRT)
      MREF = 0

      DO K=1,NREF
        J = 0
        IF ( PWDR ) THEN
          IF ( HTYPE(1:3).EQ.'PNT' ) THEN
```

```
            KREF = NREFI - K
          ELSE
            KREF = NREFI + K
          END IF
          IS = HEXTOINT('0000FFFF')                  !1111 1111 1111 1111
          I = REDREFP(IUEXP,KREF,IS,HKL,MUL,ICODE,
     1        PRFOCOR,DSPACE,LAM,FOSQ,PEAKI,
     1        FOTSQ,FCSQ,FCTSQ,PHAS,TRANS,
     1        EXTCOR,PROFLP,TF)
          IPHAS = MOD(ICODE/1000,10)
          ILAM  = MOD(ICODE/100,10)
          PEAKI = 100.0*PEAKI/BIGFO
        ELSE
          KREF = NREFI + K
          IS = HEXTOINT('00006FFD')                   !'00 0000 0000 0110 1111 1111
1101'
          I = REDREFS(IUEXP,KREF,IS,HKL,0,ICODE,
     1        INCDNT,DSPACE,LAM,FOSQ,SIGFO,
     1        FOTSQ,FCSQ,FCTSQ,PHAS,0,
     1        EXTCOR,WTFO,0,0,0,
     1        0,0,0,0,0, 0,0,0)
          IPHAS = MOD(ICODE/10000,10)
          IELEM = MOD(ICODE/1000,10)
          IMAG  = MOD(ICODE/100,10)
        END IF
        do i=1,3
          minhkl(i) = min(minhkl(i), nint(hkl(I)))
          maxhkl(i) = max(maxhkl(i), nint(hkl(I)))
        END DO
        J = J + 1
        WRITE(BUFFER(J),'(3I4)') (NINT(HKL(i)),i=1,3)
        BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
        IF ( (PWDR .AND. PROFLP.GT.0.5) .OR. SNGL ) THEN
          MREF = MREF+1
          DMAX = MAX(DMAX, DSPACE)
          DMIN = MIN(DMIN, DSPACE)
          IF (LAM2 .NE. 0) THEN
            J = J + 1
            WRITE(BUFFER(J),'(I2)') ILAM + 1
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
          END IF
          IF (NPHASES .GT. 1) THEN
             J = J + 1
             WRITE(BUFFER(J),'(I2)') IPHAS
             BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
          ENDIF
          J = J + 1
          IF ( (SNGL .AND. WTFO.GT.0.0) .OR. PWDR ) THEN
            BUFFER(J) = 'o'
            NUMOBS = NUMOBS+1
          ELSE
            BUFFER(J) = '<'
          END IF
```

```
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            J = J + 1
            ESD = -0.01
            IF ( SNGL .AND. FOSQ.GT.0.0 ) ESD = SIGFO*FOTSQ/FOSQ
            LN = -1
            CALL FESD(FOTSQ, ESD, BUFFER(J),LN)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            J = J + 1
            LN = -1
            CALL FESD(FCTSQ, -ABS(ESD), BUFFER(J), LN)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            J = J + 1
            LN = -1
            CALL FESD(PHAS, -0.1, BUFFER(J), LN)
            BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            IF ( PWDR ) THEN
              J = J + 1
              LN = -1
              CALL FESD(DSPACE, -0.0001, BUFFER(J), LN)
              BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
              J = J + 1
              LN = -1
              CALL FESD(PEAKI, -0.1, BUFFER(J), LN)
              BUFLEN(J) = MAX(BUFLEN(J),LENCH(BUFFER(J)))
            END IF
            WRITE (IUSCRT,'(11A)') (BUFFER(JJ),JJ=1,J)
            JMAX = J
          END IF
        END DO
        REWIND(IUSCRT)
        DO I=1,MREF
          READ(IUSCRT,'(11A)') (BUFFER(JJ),JJ=1,JMAX)
          WRITE(IUCIF,'(11(A,:,1x))') (BUFFER(JJ)(1:BUFLEN(JJ)),JJ=1,JMAX)
        END DO
        CLOSE(IUSCRT,STATUS='DELETE')
        write (text,'(i9)') numobs
        CALL WRVAL(IUCIF, '_reflns_number_observed', text)
        write (text,'(i5)') minhkl(1)
        CALL WRVAL(IUCIF, '_reflns_limit_h_min', text)
        write (text,'(i5)') maxhkl(1)
        CALL WRVAL(IUCIF, '_reflns_limit_h_max', text)
        write (text,'(i5)') minhkl(2)
        CALL WRVAL(IUCIF, '_reflns_limit_k_min', text)
        write (text,'(i5)') maxhkl(2)
        CALL WRVAL(IUCIF, '_reflns_limit_k_max', text)
        write (text,'(i5)') minhkl(3)
        CALL WRVAL(IUCIF, '_reflns_limit_l_min', text)
        write (text,'(i5)') maxhkl(3)
        CALL WRVAL(IUCIF, '_reflns_limit_l_max', text)
        write (text,'(f8.3)') dmin
        CALL WRVAL(IUCIF, '_reflns_d_resolution_high', text)
```

```
      write (text,'(f8.3)') dmax
      CALL WRVAL(IUCIF, '_reflns_d_resolution_low', text)
C?    _reflns_number_total                   1592
C?    _reflns_observed_criterion             F_>_6.0_\s(F)
      RETURN
      END
```

Subroutine ADDQUOTE for program GSAS2CIF

This subroutine is used to surround an ASCII string with quotes, when needed. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE addquote(lbl,lbl1,ln)
      CHARACTER*(*) LBL,LBL1
      INTEGER*4     LN
      INTEGER*4     I,J
      ln = LENCH(lbl)
      j = 1
C remove initial blanks
      do while (lbl(j:j) .eq. ' ' .and. j .le. ln)
        j = j + 1
      END DO
      lbl1 = lbl(j:)
      ln = LENCH(lbl1)
C is there a blank in the string?
      i = J
      do while (lbl(I:I) .ne. ' ')
        i = i + 1
        if (i .gt. ln+J-1) return ! no
      END DO
C yes
      lbl1 = '"'//lbl(J:ln+J-1)//'"'
      ln = ln + 2
      RETURN
      END
```

Subroutine FESD for program GSAS2CIF

This subroutine is used to format numbers for CIF in a variation of crystallographic notation. Note that if the uncertainty value is negative, the uncertainty is not printed, but rather, the uncertainty determines the number of significant digits. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE FESD(value,esd,string,ln)

C-----------------------------------------------------------------------
C format a value & esd as a string in crystallographic notation
C Use a negative esd to indicate the level of significance:
C  value 123.456, error=0.01 ==> 123.46(1)
C  value 123.456, error=-.01 ==> 123.46
C-----------------------------------------------------------------------
      REAL          VALUE,ESD
      CHARACTER*(*) STRING
      INTEGER*4     LN                 !if <0 input then fixed field
      INTEGER*4     IDEC,IFLD
      CHARACTER*20  FMTSTR
      LOGICAL*4     IFXD

      IFXD = .FALSE.
      IF ( LN.LT.0 ) IFXD = .TRUE.
      IF (VALUE .eq. 0 .and. esd .eq. 0) then
        IDEC = 1
        IFLD = 5
      ELSE IF (VALUE .eq. 0) then
        IDEC = max(0.,1.545-LOG10(ABS(ESD)))
        IFLD = 4+IDEC
      ELSE IF (esd .eq. 0) then
        IDEC = 5
        IFLD = max(1.,LOG10(abs(VALUE)))+3+IDEC
      ELSE
        IDEC = max(0.,1.545-LOG10(MAX(0.000001*ABS(VALUE),ABS(ESD))))
        IFLD = max(1.,LOG10(MAX(abs(ESD),abs(VALUE))))+3+IDEC
      END IF
      IF (esd .le. 0) then
        ISIGW = 0
      ELSE
        ISIG = NINT(ESD * (10.0**IDEC))
        ISIGW = 1. + LOG10(1.*ISIG)
      END IF
C remove insignificant figures to the left of the decimal
      if (ISIGW .gt. 2) THEN
        xmult = 10.**(isigw-2)
        value = xmult*NINT(value/xmult)
        isig = xmult*NINT(isig/xmult)
      END IF
      IF ( ISIGW .eq. 0 ) THEN
        WRITE(FMTSTR,'(A,I2,A,I1,A)') '(F',IFLD,'.',IDEC,')'
        WRITE (string,FMTSTR) VALUE
        ln = lench(string)
C remove trailing zeros
        IF ( .NOT.IFXD ) THEN
          DO WHILE (string(ln:ln) .eq. '0'
     1      .AND. STRING(LN-1:LN-1).NE.'.' )
            string(ln:ln) = ' '
            ln = ln - 1
          END DO
        END IF
      ELSE IF (IDEC .gt. 0) THEN
        WRITE(FMTSTR,'(A,I2,A,I1,A,I2,A)') '(F',IFLD,'.',IDEC,
     1    ',1H(,I',ISIGW,',1H))'
        WRITE (string,FMTSTR) VALUE,ISIG
        ln = lench(string)
      ELSE
        WRITE(FMTSTR,'(A,I2,A,I2,A)') '(I',IFLD,',1H(,I',ISIGW,',1H))'
        WRITE (string,FMTSTR) NINT(VALUE),ISIG
        ln = lench(string)
      END IF
      RETURN
      END
```

Subroutine LISTPRF for program GSAS2CIF

This subroutine is used to describe the current peak profile function and list parameter values. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE LISTPRF(IUCIF,NPRF,PTYP,PCOF,LAUE,NAXIS,HTYPE,CTOF)
!Purpose: List powder profile type 1 parameters

      INCLUDE       '../INCLDS/COPYRIGT.FOR'

!PSEUDOCODE:

!CALLING ARGUMENTS:

      INTEGER*4     IUCIF                   !Output file
      INTEGER*4     NPRF                    !Number of coefficients
      INTEGER*4     PTYP                    !Profile function type
      REAL*4        PCOF(36)                !Profile coefficients
      INTEGER*4     LAUE                    !Laue class
      INTEGER*4     NAXIS                   !Unique axis for monoclinic
      CHARACTER*4   HTYPE                   !histogram type code
      REAL*4        CTOF                    !Peak cutoff

!INCLUDE STATEMENTS:

!LOCAL VARIABLES:

      INTEGER*4     NSTR(14)                !No. Shkl strain terms

!SUBROUTINES CALLED:

!FUNCTION DEFINITIONS:

!DATA STATEMENTS:

      DATA  NSTR/15,9,6,5,4, 5,4,5,4,4, 3,3,2,2/

!CODE:

      IF ( HTYPE(2:3).EQ.'NT' ) THEN
        WRITE(IUCIF,'(A,I2,A,I3,A)') ' TOF Profile function number',
     1    PTYP,' with ',NPRF,' terms'
C taken from SUBROUTINE EDTPTP1
        IF ( ABS(PTYP).EQ.1 ) THEN
          NPRF = 12
          WRITE(IUCIF,'(A)') ' Profile coefficients for Von Dreele,',
    1      ' Jorgensen & Windsor convolution function'
         WRITE(IUCIF,'(A)') '  J. Appl. Cryst., 15,581-589(1982)'
         WRITE(IUCIF,'(A)')
    1      '  Modified by Von Dreele - unpublished (1983).'
         WRITE(IUCIF,2) (PCOF(I),I=1,5,2),(PCOF(I),I=2,6,2),PCOF(8),
    1      PCOF(9),PCOF(7),(PCOF(I),I=10,12)
2        FORMAT(
    1      ' #1(alp-0) = ',F8.4,
    1      ' #3(bet-0) = ',1PE12.4,
    1      ' #5(sig-0) = ',0PF10.3,/,
    1      ' #2(alp-1) = ',F8.4,
    1      ' #4(bet-1) = ',1PE12.4,
    1      ' #6(sig-1) = ',0PF10.3,/,
    1      ' #8(rstr ) = ',F8.3,
    1      ' #9(rsta ) = ',F12.3,
    1      ' #7(sig-2) = ',F10.3,/
    1      ' #10(rsca) = ',F8.3,
    1      ' #11(s1ec) = ',F12.2,
    1      ' #12(s2ec) = ',F10.2)
       ELSE IF ( ABS(PTYP).EQ.2 ) THEN
         NPRF = 15
         WRITE(IUCIF,'(A)') ' Profile coefficients for W.I.F. David',
    1      ' function; a convolution of the'
         WRITE(IUCIF,'(A)')
    1      ' Ikeda-Carpenter and Pseudo-Voight functions'
         WRITE(IUCIF,'(A)')
    1      '  W.I.F. David, J. Appl. Cryst., 19,63-64,(1986)'
         WRITE(IUCIF,'(A)')
    1      '  W.I.F. David - unpublished (1986).'
         WRITE(IUCIF,3) (PCOF(I),PCOF(I+4),
    1      PCOF(I+7),I=1,3),PCOF(4),
    1      (PCOF(I),I=11,15)
3        FORMAT(        ' #1 (alp-0) = ',F8.4,
    1      ' #5 (sig-0) = ',F8.2,
    1      ' #8 (gam-0) = ',F8.2,/,
    1      ' #2 (alp-1) = ',F8.4,
    1      ' #6 (sig-1) = ',F8.2,
    1      ' #9 (gam-1) = ',F8.2,/,
    1      ' #3 (beta ) = ',F8.2,
    1      ' #7 (sig-2) = ',F8.2,
    1      ' #10(gam-2) = ',F8.2,/,
    1      ' #4(switch) = ',F8.2,
    1      ' #11(ptec ) = ',F8.2,
    1      ' #12(stec ) = ',F8.2,/,
    1      ' #13(difc ) = ',F8.2,
    1      ' #14(difa ) = ',F8.2,
    1      ' #15(zero ) = ',F8.2)
       ELSE IF ( ABS(PTYP).EQ.3 ) THEN
```

```
         NPRF = 21
         WRITE(IUCIF,'(A)')' Profile coefficients for exponential'//
    1      ' pseudovoigt'
         WRITE(IUCIF,'(A)')
    1      '  convolution Von Dreele, 1990 (unpublished)'
         WRITE(IUCIF,4)(PCOF(I),I=1,21)
4        FORMAT(' #1 (alp  ) = ',F8.4,
    1      ' #2 (bet-0) = ',F8.6,
    1      ' #3 (bet-1) = ',F8.6,/,
    1      ' #4 (sig-0) = ',F8.1,
    1      ' #5 (sig-1) = ',F8.1,
    1      ' #6 (sig-2) = ',F8.1,/,
    1      ' #7 (gam-0) = ',F8.2,
    1      ' #8 (gam-1) = ',F8.2,
    1      ' #9 (gam-2) = ',F8.2,/,
    1      ' #10(gsf  ) = ',F8.2,
    1      ' #11(g1ec ) = ',F8.2,
    1      ' #12(g2ec ) = ',F8.2,/,
    1      ' #13(rstr ) = ',F8.3,
    1      ' #14(rsta ) = ',F8.3,
    1      ' #15(rsca ) = ',F8.3,/,
    1      ' #16(L11)   = ',F8.3,
    1      ' #17(L22)   = ',F8.3,
    1      ' #18(L33)   = ',F8.3,/,
    1      ' #19(L12)   = ',F8.3,
    1      ' #20(L13)   = ',F8.3,
    1      ' #21(L23)   = ',F8.3)
       ELSE IF ( ABS(PTYP).EQ.4 ) THEN
         NPRF = 12+NSTR(LAUE)
         WRITE(IUCIF,'(A)') ' Profile coefficients for exponential'//
    1      ' pseudovoigt'
         WRITE(IUCIF,'(A)')
    1      '  convolution Von Dreele, 1990 (unpublished)'
         WRITE(IUCIF,'(A)')
    1      ' Microstrain broadening by P.W. Stephens, '//
    1      ' (1999). J. Appl. Cryst.,32,281-289.'
         WRITE(IUCIF,5) (PCOF(I),I=1,12)
5        FORMAT(' #1 (alp  ) = ',F8.4,
    1      ' #2 (bet-0) = ',F8.6,
    1      ' #3 (bet-1) = ',F8.6,/,
    1      ' #4 (sig-1) = ',F8.1,
    1      ' #5 (sig-2) = ',F8.1,/,
    1      ' #6 (gam-2) = ',F8.2,
    1      ' #7 (g2ec ) = ',F8.2,
    1      ' #8 (gsf  ) = ',F8.2,/,
    1      ' #9 (rstr ) = ',F8.3,
    1      ' #10(rsta ) = ',F8.3,
    1      ' #11(rsca ) = ',F8.3,/,
    1         ' #12(eta  ) = ',F8.4)
          IF ( LAUE.GE.13) THEN                                   !cubic
            WRITE(IUCIF,10)(i,PCOF(I),I=13,NPRF)
10          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S220 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.11 .OR. LAUE.EQ.12 ) THEN     !hexagonal
            WRITE(IUCIF,11)(I,PCOF(I),I=13,NPRF)
11          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S004 ) = ',1PE8.1,
    1         ' #',i2,'(S202 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.10 ) THEN                             !trigonal 3bar1m
            WRITE(IUCIF,12)(I,PCOF(I),I=13,NPRF)
12          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S004 ) = ',1PE8.1,/,
    1         ' #',i2,'(S202 ) = ',1PE8.1,
    1         ' #',i2,'(S211 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.9 ) THEN                              !trigonal 3barm1
            WRITE(IUCIF,13)(I,PCOF(I),I=13,NPRF)
13          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S004 ) = ',1PE8.1,/,
    1         ' #',i2,'(S202 ) = ',1PE8.1,
    1         ' #',i2,'(S301 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.8 ) THEN                              !trigonal 3bar
            WRITE(IUCIF,14)(I,PCOF(I),I=13,NPRF)
14          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S004 ) = ',1PE8.1,
    1         ' #',i2,'(S202 ) = ',1PE8.1,/,
    1         ' #',i2,'(S310 ) = ',1PE8.1,
    1         ' #',i2,'(S211 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.7 ) THEN                              !rhombohedral 3m
            WRITE(IUCIF,15)(I,PCOF(I),I=13,NPRF)
15          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S220 ) = ',1PE8.1,/,
    1         ' #',i2,'(S310 ) = ',1PE8.1,
    1         ' #',i2,'(S211 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.6 ) THEN                              !rhombohedral 3
            WRITE(IUCIF,16)(I,PCOF(I),I=13,NPRF)
16          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S220 ) = ',1PE8.1,
    1         ' #',i2,'(S310 ) = ',1PE8.1,/,
    1         ' #',i2,'(S301 ) = ',1PE8.1,
    1         ' #',i2,'(S211 ) = ',1PE8.1)
          ELSE IF ( LAUE.EQ.5 ) THEN                             !tetragonal 4/mmm
            WRITE(IUCIF,17)(I,PCOF(I),I=13,NPRF)
17          FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1         ' #',i2,'(S004 ) = ',1PE8.1,/,
    1         ' #',i2,'(S220 ) = ',1PE8.1,
    1         ' #',i2,'(S202 ) = ',1PE8.1)
```

```
         ELSE IF ( LAUE.EQ.4 ) THEN                      !tetragonal 4/m
           WRITE(IUCIF,18)(I,PCOF(I),I=13,NPRF)
18         FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1        ' #',i2,'(S004 ) = ',1PE8.1,
    1        ' #',i2,'(S220 ) = ',1PE8.1,/,
    1        ' #',i2,'(S202 ) = ',1PE8.1,
    1        ' #',i2,'(S310 ) = ',1PE8.1)
         ELSE IF ( LAUE.EQ.3 ) THEN                      !orthorhombic
           WRITE(IUCIF,19)(I,PCOF(I),I=13,NPRF)
19         FORMAT(' #',i2,'(S400 ) = ',1PE8.1,
    1        ' #',i2,'(S040 ) = ',1PE8.1,
    1        ' #',i2,'(S004 ) = ',1PE8.1,/,
    1        ' #',i2,'(S220 ) = ',1PE8.1,
    1        ' #',i2,'(S202 ) = ',1PE8.1,
    1        ' #',i2,'(S022 ) = ',1PE8.1)
         ELSE IF ( LAUE.EQ.2 ) THEN                      !monoclinic
           WRITE(IUCIF,19)(I,PCOF(I),I=13,18)
           IF ( NAXIS.EQ.1) THEN
             WRITE(IUCIF,20) (I,PCOF(I),I=19,NPRF)
20           FORMAT(' #',i2,'(S013 ) = ',1PE8.1,
    1          ' #',i2,'(S031 ) = ',1PE8.1,
    1          ' #',i2,'(S211 ) = ',1PE8.1)
           ELSE IF ( NAXIS.EQ.2 ) THEN
             WRITE(IUCIF,21)(I,PCOF(I),I=19,NPRF)
21           FORMAT(' #',i2,'(S301 ) = ',1PE8.1,
    1          ' #',i2,'(S103 ) = ',1PE8.1,
    1          ' #',i2,'(S121 ) = ',1PE8.1)
           ELSE
             WRITE(IUCIF,22)(I,PCOF(I),I=19,NPRF)
22           FORMAT(' #',i2,'(S130 ) = ',1PE8.1,
    1          ' #',i2,'(S310 ) = ',1PE8.1,
    1          ' #',i2,'(S112 ) = ',1PE8.1)
           END IF
         ELSE IF ( LAUE.EQ.1 ) THEN                      !triclinic
           WRITE(IUCIF,19)(I,PCOF(I),I=13,18)
           WRITE(IUCIF,23)(I,PCOF(I),I=19,NPRF)
23         FORMAT(' #',i2,'(S310 ) = ',1PE8.1,
    1        ' #',i2,'(S103 ) = ',1PE8.1,
    1        ' #',i2,'(S031 ) = ',1PE8.1,/,
    1        ' #',i2,'(S130 ) = ',1PE8.1,
    1        ' #',i2,'(S301 ) = ',1PE8.1,
    1        ' #',i2,'(S013 ) = ',1PE8.1,/,
    1        ' #',i2,'(S211 ) = ',1PE8.1,
    1        ' #',i2,'(S121 ) = ',1PE8.1,
    1        ' #',i2,'(S112 ) = ',1PE8.1)
         END IF
       ELSE
         WRITE(IUCIF,'(A)') ' Profile option not installed.'
```

```
          WRITE(IUCIF,'(A)') ' This is an error & should not happen!'
        END IF
      ELSE IF ( HTYPE(2:3).EQ.'NC' .OR. HTYPE(2:3).EQ.'XC' ) THEN
C taken from SUBROUTINE EDTPTP3
        WRITE(IUCIF,'(A,I2,A,I3,A)') ' CW Profile function number',
     1    PTYP,' with ',NPRF,' terms'
        IF ( PTYP.EQ.1 ) THEN
          NPRF = 6
          WRITE(IUCIF,'(A)')
     1      ' Profile coefficients for Simpson''s rule'//
     1      ' integration of Gaussian function'
          WRITE(IUCIF,'(A)')
     1      ' C.J. Howard (1982). J. Appl. Cryst.,15,615-620.'
          WRITE(IUCIF,'(A)')
     1      ' Cooper & Sayer, J. Appl. Cryst., 8, 615-618'//
     1      ' (1975).'
          WRITE(IUCIF,'(A)')
     1      ' Thomas, J. Appl. Cryst., 10, 12-13(1977).'
          WRITE(IUCIF,32)(PCOF(I),I=1,6)
 32       FORMAT(        ' #1(U)    = ',F8.3,
     1      ' #2(V)    = ',F8.3,
     1      ' #3(W)    = ',F8.3,/,
     1      ' #4(asym) = ',F8.4,
     1      ' #5(F1)   = ',F8.3,
     1      ' #6(F2)   = ',F8.3)
        ELSE IF ( PTYP.EQ.2 ) THEN
          NPRF = 18
          WRITE(IUCIF,'(A)')
     1      ' Profile coefficients for Simpson''s rule'//
     1      ' integration of pseudovoigt function'
          WRITE(IUCIF,'(A)')
     1      ' C.J. Howard (1982). J. Appl. Cryst.,15,615-620.'
          WRITE(IUCIF,'(A)')
     1      ' P. Thompson, D.E. Cox & J.B. Hastings (1987).'//
     1      ' J. Appl. Cryst.,20,79-83.'
          WRITE(IUCIF,33)(PCOF(I),I=1,18)
 33       FORMAT(        ' #1(GU)   = ',F8.3,
     1      ' #2(GV)   = ',F8.3,
     1      ' #3(GW)   = ',F8.3,/,
     1      ' #4(LX)   = ',F8.3,
     1      ' #5(LY)   = ',F8.3,
     1      ' #6(trns) = ',F8.3,/,
     1      ' #7(asym) = ',F8.4,
     1      ' #8(shft) = ',F8.4,
     1      ' #9(GP)   = ',F8.3,/,
     1      ' #10(stec)= ',F8.2,
     1      ' #11(ptec)= ',F8.2,
     1      ' #12(sfec)= ',F8.2,/,
     1      ' #13(L11) = ',F8.3,
     1      ' #14(L22) = ',F8.3,
     1      ' #15(L33) = ',F8.3,/,
     1      ' #16(L12) = ',F8.3,
     1      ' #17(L13) = ',F8.3,
     1      ' #18(L23) = ',F8.3)
        ELSE IF ( PTYP.EQ.3 ) THEN
          NPRF = 19
          WRITE(IUCIF,'(A)') ' Pseudovoigt profile coefficients as'//
     1      ' parameterized in'
          WRITE(IUCIF,'(A)')
     1      ' P. Thompson, D.E. Cox & J.B. Hastings (1987).'//
     1      ' J. Appl. Cryst.,20,79-83.'
          WRITE(IUCIF,'(A)')
     1      ' Asymmetry correction of L.W. Finger, D.E.'//
     1      ' Cox & A. P. Jephcoat (1994).',
     1      ' J. Appl. Cryst.,27,892-900.'
          WRITE(IUCIF,34)(PCOF(I),I=1,19)
 34        FORMAT(' #1(GU)   = ',F8.3,
     1      ' #2(GV)   = ',F8.3,
     1      ' #3(GW)   = ',F8.3,/,
     1      ' #4(GP)   = ',F8.3,
     1      ' #5(LX)   = ',F8.3,
     1      ' #6(LY)   = ',F8.3,/,
     1      ' #7(S/L)  = ',F8.4,
     1      ' #8(H/L)  = ',F8.4,/,
     1      ' #9(trns) = ',F8.2,
     1      ' #10(shft)= ',F8.4,/,
     1      ' #11(stec)= ',F8.2,
     1      ' #12(ptec)= ',F8.2,
     1      ' #13(sfec)= ',F8.2,/
     1      ' #14(L11) = ',F8.3,
     1      ' #15(L22) = ',F8.3,
     1      ' #16(L33) = ',F8.3,/,
     1      ' #17(L12) = ',F8.3,
     1      ' #18(L13) = ',F8.3,
     1      ' #19(L23) = ',F8.3)
        ELSE IF ( PTYP.EQ.4 ) THEN
          NPRF = 12+NSTR(LAUE)
          WRITE(IUCIF,'(A)')
     1      ' Pseudovoigt profile coefficients as'//
     1      ' parameterized in'
          WRITE(IUCIF,'(A)')
     1      ' P. Thompson, D.E. Cox & J.B. Hastings (1987).'//
     1      ' J. Appl. Cryst.,20,79-83.'
          WRITE(IUCIF,'(A)')
     1      ' Asymmetry correction of L.W. Finger, D.E.'//
     1      ' Cox & A. P. Jephcoat (1994).',
     1      ' J. Appl. Cryst.,27,892-900.'
          WRITE(IUCIF,'(A)')
     1      ' Microstrain broadening by P.W. Stephens, '//
     1      ' (1999). J. Appl. Cryst.,32,281-289.'
          WRITE(IUCIF,35)(PCOF(I),I=1,12)
 35       FORMAT(' #1(GU)   = ',F8.3,
     1      ' #2(GV)   = ',F8.3,
     1      ' #3(GW)   = ',F8.3,/,
     1      ' #4(GP)   = ',F8.3,
     1      ' #5(LX)   = ',F8.3,
     1      ' #6(ptec) = ',F8.2,/,
     1      ' #7(trns) = ',F8.2,
     1      ' #8(shft) = ',F8.4,
     1      ' #9(sfec) = ',F8.2,/,
     1      ' #10(S/L) = ',F8.4,
     1      ' #11(H/L) = ',F8.4,
     1      ' #12(eta) = ',F8.4)
          IF ( LAUE.GE.13) THEN                              !cubic
            WRITE(IUCIF,10)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.11 .OR. LAUE.EQ.12 ) THEN        !hexagonal
            WRITE(IUCIF,11)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.10 ) THEN                        !trigonal 3bar1m
            WRITE(IUCIF,12)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.9 ) THEN                         !trigonal 3barm1
            WRITE(IUCIF,13)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.8 ) THEN                         !trigonal 3bar
            WRITE(IUCIF,14)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.7 ) THEN                         !rhombohedral 3m
            WRITE(IUCIF,15)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.6 ) THEN                         !rhombohedral 3
            WRITE(IUCIF,16)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.5 ) THEN                        !tetragonal 4/mmm
            WRITE(IUCIF,17)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.4 ) THEN                         !tetragonal 4/m
            WRITE(IUCIF,18)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.3 ) THEN                         !orthorhombic
            WRITE(IUCIF,19)(I,PCOF(I),I=13,NPRF)
          ELSE IF ( LAUE.EQ.2 ) THEN                         !monoclinic
            WRITE(IUCIF,19)(I,PCOF(I),I=13,18)
            IF ( NAXIS.EQ.1) THEN
              WRITE(IUCIF,20)(I,PCOF(I),I=19,NPRF)
            ELSE IF ( NAXIS.EQ.2 ) THEN
              WRITE(IUCIF,21)(I,PCOF(I),I=19,NPRF)
            ELSE
              WRITE(IUCIF,22)(I,PCOF(I),I=19,NPRF)
            END IF
          END IF
        ELSE
```

```
          WRITE(IUCIF,'(A)') ' Profile function not installed.'
          WRITE(IUCIF,'(A)') ' This is an error & should not happen!'
        END IF
      ELSE IF ( HTYPE(2:3).EQ.'XE' ) THEN
C taken from SUBROUTINE EDTPTP4
        WRITE(IUCIF,'(2A,I2,A,I3,A)') ' Energy Dispersive X-ray',
     1    ' Profile function number',PTYP,' with ',NPRF,' terms'
        IF ( PTYP.EQ.1 ) THEN
          NPRF = 5
          WRITE(IUCIF,'(A)')
     1      ' Profile coefficients for Gaussian function'
          WRITE(IUCIF,42) (PCOF(I),I=1,5)
 42       FORMAT(       ' #1(A)    = ',F8.4,
     1      ' #2(B)    = ',F8.4,
     1      ' #3(C)    = ',F8.4,/
     1      ' #4(ds)   = ',F8.4,
     1      ' #5(cds)  = ',F8.4)
        ELSE
          WRITE(IUCIF,'(A)') ' Profile function not installed.'
          WRITE(IUCIF,'(A)') ' This is an error & should not happen!'
        END IF
      END IF
      WRITE(IUCIF,'(2A,F7.4,A)') ' Peak tails are ignored ',
     1  ' where the intensity is below',CTOF,
     1  ' times the peak'
      RETURN
      END
```

Subroutine WRITERAWDATA for program GSAS2CIF

This subroutine is used to copy "raw" data from the GSAS .RAW file for a powder histogram to the output CIF file. See the gsas2cif documentation for an explanation of this code.

```
      SUBROUTINE WRITERAWDATA(IUEXP,IUCIF,IHST,HTYPE,FIXEDSTEP)

!PURPOSE: Read .RAW powder files and write to CIF -- used only when the
!         there are more raw data than calc points

      INCLUDE       '../INCLDS/COPYRIGT.FOR'

!PSEUDOCODE:

!CALLING ARGUMENTS:

      INTEGER*4     IUEXP                    !Unit no. for .EXP file
      INTEGER*4     IUCIF                    !Unit no. for .CIF file
      INTEGER*4     IHST                     !Histogram no. to be read
      CHARACTER*4   HTYPE                    !Histogram type
      LOGICAL*4     FIXEDSTEP                ! true for fixed step data

!INCLUDE STATEMENTS:

!LOCAL VARIABLES:

      CHARACTER*12  KEYVAL                   !ISAM key
      CHARACTER*66  RAWNAM                   !.RAW file name
      INTEGER*4     IURAW                    !Unit no. for .RAW file
      CHARACTER*66  INSNAM                   !Incident spectrum name
      INTEGER*4     IUINS                    !Unit no. for incident spectrum
      LOGICAL*4     NMCHG                    !=.TRUE. if a new name was read
      INTEGER*4     ISAM                     !ISAM error flag
      INTEGER*4     BANK                     !Bank no. requested
      INTEGER*4     NCHANS                   !No. of channels written on .Pnn file
      INTEGER*4     MCHANS                   !No. of channels from .RAW file
      INTEGER*4     OFFSET                   !No. of channels to be skipped
      INTEGER*4     CHKSUM                   !Intensity check sum for .RAW data
      REAL*4        YO(90000)                !Observed intensities
!      REAL*4        YI(90000)                !Incident intensities
      REAL*4        YW(90000)                !Variances on YO
!      REAL*4        IW(90000)                !Variances on YI
      REAL*4        TOF(90000)               !Positions
      LOGICAL*4     IERR                     !Time map error flag
      REAL*4        TMAX                     !Max. TOF or TTH allowed for incident
function
      CHARACTER*68  TEXT                     !ISAM data string
      CHARACTER*80  ITITL                    !Title on raw file
      LOGICAL*4     IXST                     !File exist flag
      LOGICAL*4     NEEDESD             ! true if the ESD's are not SQRT(I)

!SUBROUTINES CALLED:

!FUNCTION DEFINITIONS:

      INTEGER*4     READEXP             !ISAM read function
      CHARACTER*6   HSTKEY              !'HST' key maker

!DATA STATEMENTS:

!CODE:

      KEYVAL = HSTKEY(IHST)//' BANK '
      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
      READ(TEXT,'(I5)') BANK
      ISAM = READEXP(IUEXP,KEYVAL(1:6)//' CHANS',TEXT)
      READ(TEXT,'(5I10,I5)') OFFSET,ICLMP,NCHANS,CHKSUM,MCHANS,ISAMP
      IF ( ISAMP.EQ.0 ) ISAMP = 1
      ISAM = READEXP(IUEXP,KEYVAL(1:6)//'  HFIL',TEXT)
      RAWNAM = TEXT(3:68)
      ISAM = READEXP(IUEXP,KEYVAL(1:6)//' NEXC ',TEXT)
      READ (TEXT,'(I5)') NEXC
      WRITE (KEYVAL(7:12),'(A,I3)') 'EXC',NEXC
      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
      READ (TEXT,'(F10.0)') TMAX
      IF ( HTYPE(3:3).EQ.'C' ) THEN
        TMAX = TMAX*100.0
      ELSE IF ( HTYPE(3:3).EQ.'T' ) THEN
        TMAX = TMAX*1000.0
      END IF

      CALL OPNRAW(IURAW,.FALSE.,RAWNAM,NMCHG)
      READ(IURAW,'(A)',REC=1) ITITL
      CALL READHST(IURAW,BANK,HTYPE,TEXT,MCHANS,TOF,YO,.TRUE.,YW,IERR)
      CLOSE(IURAW)

      LEN=INDEX(RAWNAM,' ')-1
      WRITE(IULST,3) BANK,RAWNAM(1:LEN)
3     FORMAT(' Data for bank ',I2,' read from file ',A)

C at least for right now, ignore the incident spectrum
C Bob, do you want to change this?
!      KEYVAL = KEYVAL(1:6)//'I ITYP'
!      ISAM = READEXP(IUEXP,KEYVAL,TEXT)
!      READ(TEXT,'(I5)') ITYP
!      IF ( ITYP.GE.10 ) THEN
!        ISAM = READEXP(IUEXP,KEYVAL(1:6)//'  MFIL',TEXT)
!        IF ( ISAM.EQ.0 ) THEN
!          INSNAM = TEXT(3:68)
!           INQUIRE(FILE=INSNAM,EXIST=IXST)
!           IF ( IXST ) THEN
!             PRINT '(A,/,1X,A)',' Incident spectrum read from file:',
!     1         INSNAM
!             CALL OPNRAW(IUINS,NEW,INSNAM,NMCHG)
!             READ(IURAW,'(A)',REC=1) ITITL
!             CALL WRITEXP(IUEXP,KEYVAL(1:6)//'  INAM','  '//ITITL(1:66))
!             CALL READHST(IUINS,BANK,HTYPE,TEXT,MCHANS,TOF,YI,
!     1         .TRUE.,IW,IERR)
!             CLOSE(IUINS)
!           ELSE
!             PRINT '(A,/,A)',' File '//INSNAM,' not found'
!             STOP 'Error in HSTREAD'
!           END IF
!         END IF
!       END IF

C test to see if SU's can be eliminated since su = sqrt(I)
      NEEDESD = .false.
      DO ICH=1+OFFSET,MCHANS
        IF (YW(ICH) .GT. 0 .AND.
     1    (YO(ICH)/YW(ICH) .LT. .95 .OR. YO(ICH)/YW(ICH) .GT. 1.05))
     1    NEEDESD = .true.
      END DO

      IF (HTYPE(3:3) .eq. 'T') THEN
C     write the detector 2theta angle
        ISAM = READEXP(IUEXP,HSTKEY(IHST)//'BNKPAR',TEXT)
        CALL WRVAL(IUCIF, '_pd_meas_2theta_fixed',TEXT(11:20))
        WRITE(IUCIF,'(/a5,1x,A)') 'loop_', '_pd_meas_time_of_flight'
      ELSE IF (FIXEDSTEP) THEN
        STEP = (TOF(MCHANS) - TOF(1+OFFSET))/(MCHANS-(1+OFFSET))
        CALL FESD(TOF(1+OFFSET)/100., -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_meas_2theta_range_min', text)
        CALL FESD(TOF(MCHANS)/100., -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_meas_2theta_range_max', text)
        CALL FESD(STEP/100., -abs(STEP/10000.), text,ln)
        CALL WRVAL(IUCIF, '_pd_meas_2theta_range_inc', text)
        WRITE(IUCIF,'(/a5,1x,A)') 'loop_'
      ELSE
        WRITE(IUCIF,'(/a5,1x,A)') 'loop_', '_pd_meas_2theta_scan'
      END IF
      WRITE(IUCIF,'(6x,A)') '_pd_meas_intensity_total'

      DO ICH=1+OFFSET,MCHANS
        ln = 1
        IF (HTYPE(3:3) .eq. 'T' .OR. .NOT. FIXEDSTEP) THEN
          CALL FESD(TOF(ICH), -TOF(ICH)*.0001, text,ln)
          text(ln+1:) = ' '
          ln = ln + 2
        END IF
        IF (NEEDESD) THEN
          ESD = 0
          IF (YW(ICH) .GT. 0) ESD = SQRT(YW(ICH))
          CALL FESD(YO(ICH), ESD, text(ln:),ln)
        ELSE
          CALL FESD(YO(ICH), -10., text,ln)
        END IF
        write (IUCIF,'(5x,A)') text(:LENCH(text))
      END DO

      write (text,'(I9)') MCHANS-OFFSET
      CALL WRVAL(IUCIF, '_pd_meas_number_of_points', text)
      RETURN
      END
```