Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-93-009

The Cambridge Crystallography Subroutine Library

P J Brown and J C Matthewman

MARK 4 USERS' MANUAL

February 1993

The Cambridge Crystallography Subroutine Library

MARK 4 USERS' MANUAL

February 1993

THE CAMBRIDGE CRYSTALLOGRAPHY SUBROUTINE LIBRARY USERS' MANUAL

P J Brown and J C Matthewman*

* Address for communication: Mrs Judy Matthewman, 24 Courtyards, Little Shelford, Cambridge CB2 5ER, United Kingdom.
Telephone Cambridge (+223) 843220

WELCOME TO C C S L

CCSL is still being developed.

This version of the Manual reflects the state of the Mark 4 system in August 1992.

New and useful facilities will be added, but in consequence we do not guarantee that some of the details given in this Manual may not change.

The authors would welcome reports of any errors or difficulties.

The Manual has been typeset using TEX

CAMBRIDGE CRYSTALLOGRAPHY SUBROUTINE LIBRARY USERS' MANUAL

CONTENTS

APPENDICES AVAILABLE SEPARATELY

DEDICATION

The Library and this Manual are dedicated to the memory of
Dr W.H. Taylor, onetime Reader in Crystallography
in the University of Cambridge.
Will was a strong influence on the lives of many
Cambridge scientists.

Amongst other things, he taught us that computing
for crystallographers is a worthwhile pursuit,
that there is always room for improvement
- and that "data" is a plural word!

Chapter 1

INTRODUCTION TO CCSL

This Manual describes the Cambridge Crystallography Subroutine Library ("CCSL"). It describes the material in the Master File of the program and is intended for distribution to users at sites where CCSL is installed.

CONTENT

The Manual is intended both as an introduction to the Library for new users and as an *aide memoire* for more experienced users. Some familiarity with the use of FORTRAN is assumed, including the use of a computer to make and edit files and to compile, load and run FORTRAN jobs.

SCOPE

All members of CCSL are included except those specifically written for Profile Refinement, for which separate documentation is available.

CHANGES

The Manual is being written in parallel with the programs. Inevitably, some of the information will change as the programs develop. New editions of the Manual will reflect important changes in the system.

REMAINDER OF THE MANUAL

This Chapter ends with a complete sample job, followed through in enough detail to introduce various necessary ideas. The other chapters describe different aspects of the Library and need not necessarily be read in sequence.

Chapter 2 gives an overall view of the Library. It should also help the user to discover what is provided, and to begin to appreciate the system as a whole. At the end is a classified list of all the available routines.

Chapter 3 describes the *Crystal Data File*, which contains data other than lists of reflections, observations etc. The bulk of the chapter provides detailed descriptions of all the input card formats in alphabetical order.

Chapter 4 describes the input of other data, such as lists of observations for Least Squares Refinement, etc. There is also a section on the use of input/output files.

Chapter 5 introduces the Least Squares Refinement facilities available, and explains some of the terminology for users who want to refine something a little different from what is already provided.

Chapter 6 explains how the CCSL deals with magnetic structures.

Chapter 7 describes how to run jobs, with reference to operating systems under which CCSL has been implemented. A list of some currently available main programs is given.

APPENDICES

The Appendices collect together lists of information useful for reference, to obtain further understanding of the system, or for writing new programs. They will usually be available separately from the Manual.

Appendix A lists the routines alphabetically, with explanation of:

prerequisite calls or settings,
what is altered or output by the routine,
what the routine is used for, and, briefly, how it works.

For each routine Appendix B lists which others it calls, and which routines call it.

Appendix C lists all the labelled COMMON, with brief descriptions of its contents. There are also lists of which routines refer to which items in the COMMON.

Appendix D describes some main programs which are available. The new user may at first find what he wants here, and will probably progress by developing his own main programs from existing programs.

All the material for the Appendices is contained within CCSL itself, and there are programs available to create the Appendices from the Master File.

AN INTRODUCTORY EXAMPLE

CCSL is a Library of FORTRAN routines to be used for standard crystallographic calculations. The routines may be thought of as tools with which the crystallographer can make his own programs to do exactly the computations he wants.

It has been written with the needs of the non-standard crystallographer (or the solid state physicist working with crystalline materials) in mind. It is not intended simply for the solution or refinement of crystal structures, for which systems such as XRAY may be more suitable.

Let us first take a simple but realistic small example. The main program:

```
      LOGICAL NOMORE
      DIMENSION H(3),K(3)
      COMMON /IOUNIT/LPT,ITI,ITO,IPLO,LUNI,IOUT
      CALL PREFIN('SIMPLE')
      CALL SYMOP
      CALL OPSYM(1)
      CALL SYMUNI
      CALL RECIP
      CALL SETGEN(0.2)
      NSUM=0
      WRITE (LPT,2000)
2000  FORMAT (//'   h     k    l  Multiplicity')
C
   1  CALL GETGEN(H,NOMORE)
      IF (NOMORE) GO TO 2
      MULT=MULBOX(H)
      NSUM=NSUM+MULT
      CALL INDFIX(H,K)
      IF (MULT .NE. 0) WRITE (LPT,2001) K,MULT
 2001 FORMAT(3I5,I8)
      GO TO 1
C
   2  WRITE (LPT,2002)  NSUM
2002  FORMAT (/' Number of reflections inside sphere =',I5)
      STOP
      END
```

if supplied with a Crystal Data File named SAMPLE.CRY which says:

```
C 7.07107   7.07107   8.66025
S     Y,     Y-X,  1/6+Z
S    -Y,     -X,   1/6-Z
```

and run as a FORTRAN job, given CCSL as a Library, produces the output on page 4.

EXPLANATION OF THE EXAMPLE

The user will naturally need some knowledge of FORTRAN. Throughout, words which are parts of the FORTRAN programs will be written in capitals, as they usually are in the actual program.

This main program should be recognisable to the user as FORTRAN, most of which he can interpret. The nine names which refer to routines of CCSL are:

PREFIN, SYMOP, SYMUNI, RECIP and SETGEN (all for input), OPSYM for symmetry output and GETGEN, MULBOX and INDFIX inside a loop generating h, k, l values.

The interesting routines are GETGEN and MULBOX. GETGEN produces in the array H the "next" set of reflection indices h, k, l, eventually covering the whole of an asymmetric unit in reciprocal space. MULBOX returns the value of the multiplicity of the reflection with indices h, k, l.

GETGEN is a FORTRAN SUBROUTINE (invoked by the word CALL), whereas MULBOX is a FORTRAN FUNCTION, in which a variable called MULBOX is set to the answer. Every member of the CCSL is either a SUBROUTINE or a FUNCTION; for ease of discussion we use the word "routine" to mean "either SUBROUTINE or FUNCTION".

GETGEN and MULBOX, then, are examples of *crystallographic* routines of CCSL. There are a variety of others, such as LOGICAL FUNCTION ISPABS(H) which tells whether h, k, l in array H is or is not a space goup absence, COMPLEX FUNCTION FCALC(H) which calculates a complex Structure Factor, and SUBROUTINE FOUR1Z which calculates one layer of a Fourier map.

INDFIX is a small *utility* routine of CCSL which takes the 3 numbers in the array H and "fixes" them (converts them from REAL numbers to INTEGERS) into the integer array K. Utility routines exist for any operation which the authors have become tired of writing several times, such as matrix multiplication, finding a number in a given list, vector products, packing small integers into bigger ones, etc.

Except for OPSYM, the remaining CCSL routines in the example (PREFIN, SYMOP, RECIP, SYMUNI and SETGEN) are all involved in the input of data necessary to perform the later calculations and are examples of *setting-up routines.*

SYMOP reads and interprets the data cards starting **S** for the space group symmetry.

RECIP reads the **C** card for the unit cell parameters.

SYMUNI makes a reciprocal space asymmetric unit, using the space group information.

SETGEN sets up the system ready for repeated use of GETGEN. It needs to know the space group, the asymmetric unit and the cell parameters, so SYMOP, SYMUNI and RECIP must all have been obeyed before SETGEN.

OPSYM(1) writes out the space group so found, in real space terms. (OPSYM(2) would use reciprocal space).

This leaves PREFIN. All CCSL programs which read a Crystal Data File should CALL PREFIN with the name of the program as argument at an early stage. It initialises various aspects of the system, and puts the crystal data into a form from which it can subsequently be read and interpreted as often as required.

The remainder of the main program is ordinary FORTRAN. Note that although the Library does the work of producing h, k, l values and multiplicities, it is the main program which controls what is done with them and how they are printed. The labelled COMMON /IOUNIT/ has to be declared in the main program to direct the output of indices to the logical unit LPT which has been set up in PREFIN to be a printable file.

```
                    * * * S I M P L E * * *
        Cambridge Crystallography Subroutine Library      Mark 4.0
                    Job run at 15:59 on 19-AUG-92
Crystal data file  SAMPLE.CRY opened
Data read by PREFIN from file SAMPLE.CRY
       1 card   labelled C
       2 cards  labelled S
Non-centrosymmetric space group with 12 operator(s)
The group is generated by the 2 element(s) numbered:  2  7
Friedel's law NOT assumed - hkl distinct from -h-k-l
General equivalent positions are:
      0                    0                    0       +
 1      x                    y                    z
 2      y                   -x+y                 1/6+z
 3     -x+y                 -x                   1/3+z
 4     -x                   -y                   1/2+z
 5     -y                    x-y                 2/3+z
 6      x-y                  x                   5/6+z
 7     -y                   -x                   1/6-z
 8     -x                   -x+y                 1/3-z
 9      x-y                 -y                   -z
10     -x+y                  y                   1/2-z
11      x                    x-y                 5/6-z
12      y                    x                   2/3-z
Indices  13  11  10 to be used for typical reflection inside asymmetric unit
Asymmetric unit has  2 plane(s):
           k>=0
           h>=k
Symmetry constraints on lattice parameters    b   =  a
                                            alpha = 90
                                             beta = 90
                                            gamma = 120
Real cell          7.0711    7.0711    8.6602   90.00   90.00  120.00
Volume =   375.0001
Reciprocal cell    0.1633    0.1633    0.1155   90.00   90.00   60.00
Volume =   0.2667E-02
Real cell quadratic products:
A (=a  sqrd)     B            C   D (=b c cos alpha)  E          F
    50.00003    50.00003    74.99992     0.00000     0.00000   -25.00002
Reciprocal cell quadratic products:
A*(=a* sqrd)     B*          C*  D*(=b*c*cos alpha*)  E*         F*
     0.02667     0.02667     0.01333     0.00000     0.00000     0.01333
Matrices for transformation of vectors to orthogonal axes
Real space:
   6.1237    0.0000    0.0000
  -3.5355    7.0711    0.0000
   0.0000    0.0000    8.6602
Reciprocal space:
   0.1633    0.0816    0.0000
   0.0000    0.1414    0.0000
   0.0000    0.0000    0.1155
  h     k     l  Multiplicity
   1     1     2      12
   1     1     1      12
   1     1     0       6
   1     0     0       6
   2     0     0       6
   0     0    -1       2
   1     0    -1      12
   2     0    -1      12
   0     0    -2       2
   1     0    -2      12
   2     0    -2      12
   0     0    -3       2
   1     0    -3      12
Number of reflections inside sphere =  108
```

COMMENTS ON THE EXAMPLE OUTPUT OPPOSITE

The output charts the progress of the program. PREFIN notes one **C** card and two **S** cards (without interpreting the data on them yet). SYMOP detects the lack of a centre of symmetry, and counts 12 operators in total in the group. In passing it looks for an **I** ("Instructions") card which could say, e.g.,

```
I FRIE  1
```

and would ask to assume Friedel's Law. It does not find one, and says so.

OPSYM(1) lists the operators (this is separate from SYMOP because once the user is happy with his space group he may well not want to list all operators for every run).

SYMUNI notes that it has not been given a **U** ("Unit") card with typical h, k, l, so it uses (13,11,10), as being a general reflection with $h > k > l$. It finds a suitable asymmetric unit and prints it out.

RECIP needs the space group symmetry, so first prints the relations imposed on the cell parameters by the symmetry. Because of these, it needs only values of cell sides a and c from the **C** card (the value of b is present on the sample Crystal Data but need not be). It reads and interprets the **C** card, filling in various useful quantities connected with the cell parameters. Not all these quantities are printed. It is possible to make RECIP print more if more detail is needed.

SETGEN then devises a scheme for traversing the whole of the asymmetric unit. The argument 0.2 is the maximum value of $\sin\theta/\lambda$ required. Again, nothing is printed, but it could be if it is needed.

The remaining output is generated by the WRITE statements in the user's main program.

Note that one of the arguments of the CALL of GETGEN is a LOGICAL variable, NOMORE, which is eventually set to be .TRUE. when there are no more h, k, l values. This must be declared LOGICAL at the head of the main program. Similarly, because arrays H and K are used, these must be DIMENSIONed at the start, as H(3), K(3).

In more complicated main programs it may be necessary to have access to variables, such as the cell parameters, used in the Library routines. Such variables are held in labelled COMMON blocks. The cell parameters are in

```
COMMON /CELPAR/CELL(3,3,2),V(2),ORTH(3,3,2),CPARS(6,2),KOM4
```

where the array CELL holds both real and reciprocal space sides and angles. If the main program wants to refer to CELL, it should declare COMMON /CELPAR/ at its head, copying it from the Library.

The user will see from the discussion above that it will be easier to read this Manual if he has available a printout of the FORTRAN source code of the Library, although this is very bulky and a copy of Appendix A may suffice.

There are, of course, certain typical computations which many users will require often, and these give rise to standard main programs using CCSL. To the user, such a main program is, for his particular computer, made into a complete job which he can regard as a 'black box'. He feeds his data into this box as if he were using a more conventional system of programs.

A new user will probably approach CCSL this way. It is quite likely, though, that soon the black box will not do exactly what he wants, and he will move on to using CCSL in the way it is intended, writing his own programs, or modifying existing programs, to fit his problem precisely.

A MORE ELABORATE EXAMPLE

Taking the sample program as a basis, let us expand it to illustrate the sort of features to expect in CCSL. Suppose we have data which describe a crystallographic structure, and we want to generate indices as above, but then to take an asymmetric unit full of reflections, sort them in ascending order of $\sin\theta/\lambda$, and list them with calculated structure factors.

The expanded program below uses in addition from CCSL:

crystallographic routines	LATABS - is it a lattice absence? FCALC - form a complex structure factor ISPABS - is it a space group absence?
utility routines	VCTMOD - vector length (for $\sin\theta/\lambda$) INDFLO - convert integers into real numbers SORTX - sort real numbers in ascending order ERRMES - send an error message if necessary
input (*setting-up*) routines	ATOPOS - read a set of atomic positions SETFOR - read a set of form factors SETANI - read anisotropic temperature factors

There are more declarations at the head of the program. The FUNCTION FCALC is COMPLEX, so both it and the variable to which the program sets it must be declared COMPLEX. ISPCE and ISTAR must be declared CHARACTER *1.

ISPABS and LATABS are LOGICAL FUNCTIONs, and must be so declared. SORTX sorts within arrays in store, so these must be DIMENSIONed.

The following, then, is the larger example main program. It should by now be fairly clear to the reader what the various parts of it are, and how they fit together.

```
      COMPLEX FC,FCALC
      LOGICAL NOMORE,ISPABS,LATABS
      DIMENSION H(3),K(3,1000),SINTH(1000),IPNT(1000)
      COMMON /IOUNIT/LPT,ITI,ITO,IPLO,LUNI,IOUT
      CHARACTER *1 ISPCE,ISTAR
      DATA ISPCE,ISTAR/' ','*'/
C
      CALL PREFIN('GETSF')
      WRITE (ITO,2002)
2002  FORMAT (' Value of sin theta/lambda max? ')
      READ (ITI,1000) S
1000  FORMAT (F10.4)
      CALL SYMOP
      CALL OPSYM(1)
      CALL OPSYM(2)
      CALL RECIP
      CALL ATOPOS
      CALL SETFOR
      CALL SETANI
      CALL SYMUNI
      WRITE (LPT,2000)
2000  FORMAT (////'  Sorted structure factors - * indicates space',
     1 'group absence:'/'     No.    h   k   l    Mult     s',
     2'            A           B          FcMod')
      CALL SETGEN(S)
C COMPLAIN AND STOP IF THERE WERE ERRORS IN THE INPUT
      CALL ERRMES(0,0,' to GETSF')
      NSUM=0
      N=0
```

```
   1  CALL GETGEN(H,NOMORE)
      IF (NOMORE) GO TO 2
      IF (LATABS(H)) GO TO 1
      MULT=MULBOX(H)
      IF (MULT .EQ. 0) GO TO 1
      N=N+1
      NSUM=NSUM+MULT
      SINTH(N)=VCTMOD(0.5,H,2)
      CALL INDFIX(H,K(1,N))
      GO TO 1
C
C SORT THE ARRAY IPNT, SO THAT IT POINTS TO THE ELEMENTS
C OF SINTH IN SEQUENCE:
   2  CALL SORTX(SINTH,IPNT,N)
      DO 3 I=1,N
      J=IPNT(I)
      CALL INDFLO(H,K(1,J))
      FC=FCALC(H)
      A=REAL(FC)
      B=AIMAG(FC)
      FCMOD=SQRT(A*A+B*B)
      IC=ISPCE
      IF (ISPABS(H)) IC=ISTAR
      M=MULBOX(H)
      WRITE (LPT,2001) IC,I,(K(L,J),L=1,3),M,SINTH(J),A,B,FCMOD
2001  FORMAT (' ',A1,I5,2X,3I4,2X,I5,F10.5,3X,3F12.5)
   3  CONTINUE
      WRITE (LPT,100)  NSUM,S
 100  FORMAT (/'  Total number of reflections inside sphere',
     1'=',I4/' S max=',F10.4)
      STOP
      END
```

The initial dialogue which has been given as a WRITE and a READ statement for simplicity could be more elegantly rendered as:

```
      CALL ASK('Value of sin theta/lambda max?')
      CALL RDREAL(S,1,IP,80,IE)
```

because the routine ASK puts out the given message on the screen, and reads in a line from the keyboard ready for routine RDREAL to read the value of S.

Similarly, the WRITE statement with FORMAT 2000 could be replaced by:

```
      CALL CENTRE(LPT,4,'Sorted structure factors - * indicates '//
     1 'space group absence;',80)
      CALL MESS(LPT,0,'    No.    h   k   l    Mult     s'//
     2 '            A           B          FcMod')
```

Chapter 2

THE STRUCTURE OF THE SYSTEM

This Chapter is an introduction to the Library as a whole, the language used, and the Master File. A reader interested initially only in the crystallography could skip to the **List of Available Routines**.

CCSL AS A FORTRAN LIBRARY

CCSL is a set of FORTRAN routines. Specifications of all the routines with detailed descriptions are in Appendix A, in alphabetical order of routine. Later in this chapter they are listed in groups under various headings, to give the user a broad view of what is available.

A FORTRAN job always has a *main* program, which usually calls routines. A CCSL user is expected to provide the main program, and then to present CCSL to be scanned as a Library. (The Master File contains some main programs with the Library).

Running a FORTRAN job varies from one computer to another, but where a FORTRAN compiler exists there should be some sort of facilities for:

a) compiling each routine of CCSL separately, and holding the set of such compiled routines as a Library, and

b) when given a CCSL main program, scanning this Library and extracting any routines referred to by the main program (and all routines referred to by them, and so on) so that the program can be run.

The point of compiling separate routines into a Library is that not every run needs every routine; the user is unlikely to call for, say, absorption corrections in the same run as a plotted Fourier. So he wishes to load and obey only those FORTRAN routines he actually wants for this run.

LANGUAGE

CCSL is written almost entirely in FORTRAN 77, (ANSI standard X3.9-1978). The reference for what is standard F77 is taken from the VAX-11 FORTRAN Language Reference Manual (AA-D034C-TE) trying to ignore the extensions printed in blue.

PORTABILITY

There will always be problems trying to make the FORTRAN portable.

The areas which are particularly troublesome in portable programs are:

1) input/output
2) use of graphical output
3) what the consequences of an error are

but there are other areas, which tend only to become apparent when the system is actually implemented on a particular machine.

We are aware of the problems of portability, but nevertheless we insist that CCSL must be portable. Some of the FORTRAN used may seem pedestrian as a result.

The best way to get a view of CCSL's FORTRAN is to print out part of the Library. It is intended to be understandable as far as possible.

THE MASTER FILE

The complete set of CCSL routines and main programs is held in a *Master File.* When a new user who does not have CCSL on his computer asks for it, he should be sent a Master File and a separate FORTRAN 77 program, CCSL.FOR, which should generate a version of the Library and a set of main programs to run on his computer.

It would be possible to give to a new user a FORTRAN Library which he could present directly to his compiler. However, one of the features of CCSL.FOR is that it can alter the dimensions of arrays (which are held symbolically in the Master File), so if the user discovers a need for, say, 100 atomic positions, when only 50 are allowed in the standard CCSL, he can remake his Library to cater for this.

It is useful to hold all possible code (all routines and all programs) together, even though the resulting file is large. CCSL.FOR ensures that all copies of a particular labelled COMMON are the same throughout the whole file; getting these out of step is a frequent source of error. All the COMMON blocks are listed in alphabetical order of their names at the start of the Master File, many with symbolic dimensions for their arrays.

The Master File also contains all the alternative code for areas which are system or device dependent. Such code is tagged, e.g. "CVMS' for lines which are relevant only to VAX systems running VMS and "CILL' or "CRAL' for lines that implement preferences used at ILL and RAL respectively. When the program CCSL.FOR is obeyed, it is told on which system or device the resulting code is intended to run. Thus the same Master File can be sent to different computing establishments.

SEQUENCE OF ROUTINES

The Master File contains a sequence of routines in alphabetical order of their names. This ordering is usually the most convenient for reference, but is unsuitable for a library on machines which have one-pass loaders. One of the options of CCSL.FOR is to generate a *sorted* list of routines, which may be correctly compiled for a one-pass loader. It uses the information stored in the first (comment) line of every routine, saying "C LEVEL" with an integer. Those routines which call nothing else from CCSL are LEVEL 1; those only calling level 1 are LEVEL 2, and so on. The sorted version of the Library has all the routines of the highest level first, then all the next level, and so on down to LEVEL 1.

ROUTINES AVAILABLE

Most CCSL routines can be assigned to one of the three broad categories introduced in the small sample main program of Chapter 1. These are:

A. *Setting-up* routines, concerned with input of information and preparation for subsequent calculation;

B. *Crystallographic* routines, doing specific calculations such as a Fourier section, a structure factor or an absorption correction;

C. *Utility* routines, which are of more general use, being small routines to perform tasks such as the solution of spherical triangles, matrix multiplication or a vector product.

The boundaries between categories are not rigid. In general, category C routines can be used in many contexts and do not include labelled COMMON (other than /IOUNIT/ which associates FORTRAN input/output units with physical devices). Category B are useful in more specific contexts; various output routines are included. Category A includes all routines which read from the Crystal Data; most of these routines are used just once at the start of a job.

Within each category there is a wide range of material. Some attempt at further classification follows. Some routines are of obvious use to users who write their own main programs, whereas some belong internally to CCSL.

All the routines available at the time of writing are listed here. The user should also consult the separate Appendix A, which gives details and may be more up-to-date.

Classified List of Subroutines

1. BASIC CRYSTALLOGRAPHY

A Setting Up

FACGRP	Extracts factor groups from a space group.
FACTGP	Finds the factor elements which generate a space group from one of its sub-groups.
FIXUNI	Deals with one potential plane face of asymmetric unit, while the unit is being formed.
INPUTC	Reads one **C** card.
INPUTS	Reads and interprets one **S** card containing a space group operator.
INPUTU	Reads and interprets a **U** card, giving a typical reflection to define the reciprocal asymmetric unit.
MAKGRP	Generates the subgroup of a space group from the given generators.
ORTHG	Calculates matrices for the transformation of vectors in real or reciprocal space, between crystallographic and orthogonal axes.
PLN3AD	A specialist routine used during the formation of the reciprocal unit cell, to offer up 3 planes as boundaries, in cubic space groups.
POLUNI	A specialist routine to "polish" the edges of a found asymmetric unit by specifying exactly how its faces and edges should be treated.
RECIP	Reads the lattice parameters and forms the reciprocal cell.
SETFC	Calls all the setting-up routines needed for nuclear structure factor calculations.
SHUFLE	Reorders the symmetry operators for various specialist applications.
SYMCEN	A specialist routine used during the input of space group symmetry to discover whether there is a centre of symmetry.
SYMFRI	Reads and interprets an item **FRIE** on an **I** card.
SYMGEN	Produces the generators of a space group which has been read by SYMOP.
SYMOP	Reads all the symmetry operators from **S** cards and generates the space group.
SYMTID	Tidies all the arrays connected with the space group symmetry.
SYMUNI	Selects a reciprocal space asymmetric unit fitting the symmetry.
TRYUNI	A specialist routine used in the setting up of the asymmetric unit, to try out a given set of planes as faces of the unit.
UNITID	A specialist routine called when an asymmetric unit has been found, to "tidy" the unit, its faces and edges.

B Crystallographic

ANGRAD	Calculates the angle in radians between two vectors, in either space.
ASUNIT	Produces reflection indices in the asymmetric unit, related to those given.
CELMAT	Sets up the matrix to convert derivatives wrt A, B, C . . (the cell quadratic products in real space) to derivatives wrt A*, B*, C* . . in reciprocal space.
EQOP	Checks whether a rotation matrix and a translation vector of a symmetry operator are already in a list, and adds them if not. Also finds lattice translations.
EULSYM	Finds the Euler angles correponding to a symmetry rotation.
GENELM	Finds the generators of a subgroup of a space group.
ISPABS	Checks space group absences.
LATABS	Checks h,k,l for being a (nuclear) lattice absence.
LATGEN	Generates points on a lattice.

LATVEC	Tests for the presence of a lattice vector.
OPSYM	Prints out the symmetry operators in either real or reciprocal space.
ORTHO	Carries out conversions between crystallographic and orthogonal axes.
RECELL	Makes real or reciprocal space cell parameters from the others.
ROTOSM	Calculates the effect of the rotation matrix of a symmetry operator, on a vector given on orthogonal axes.
ROTSYM	Rotates the vector H into RH by the given symmetry operator.
SCLPRD	Forms the scalar product of two vectors referred to crystal axes.
SUBSYM	Replaces all the symmetry parameters by those of a subgroup.
SYMBAK	Restores the original symmetry operators after a call to SUBSYM.
SYMREF	Generates a set of equivalent reflections and related phases.
VCTMOD	Calculates the modulus of the vector H, in either space.

C Utility

AXIS	Finds the axis of a given rotation matrix.
FCTOR	Finds the highest common factor of a set of indices and reduces them by that factor.
FRAC3	Makes all 3 elements of a vector fractional.
INVENT	Given a plane U and an axis H, produces a direction ANS which is in the plane but not parallel to the axis.
NORDER	Returns the order of the Jth symmetry operator.

2. DATA COLLECTION AND REDUCTION

A Setting Up

ARPRIN	A multi-mode routine to produce the output in ARRNGE type main programs.
ARREAD	Multiple entry routine to deal with reading items for ARRNGE type programs.
GAUSPT	Sets up Gauss points and weights for use in 3D integration.
ICDFMT	Interprets **I** card for **RNUM** and **MULP**.
IICD3	Interprets **I** cards for ARRNGE-type main programs, and sets them up.
INPUTD	Reads and interprets all **D** cards, for general diffraction information.
INPUTE	Reads and interprets an **E** card, and sets up extinction calculations.
INPUTG	Reads and interprets all **G** cards, for Gaussian integration of various integrals over a crystal defined by its plane faces.
SETABS	Sets up data for the calculation of absorption (or related) integrals.
SETDC	Reads **D** cards to set up crystal orientation and diffraction geometry.
SETGAU	Sets up the COMMON /GAUSS/ for Gaussian integration (for use in absorption correction type integrals).
SETLP	Prepares to calculate (X-ray) Lorentz and polarisation corrections.
SETPOL	Reads the polarisation and flipping efficiencies (for polarised neutrons).

B Crystallographic

ABMULT	Forms transmission (=1/absorption) factors or related integrals.
ABSCOR	Applies absorption corrections to groups of equivalent reflections.
ABSOR	Forms a transmission (=1/absorption) factor or related integrals.
AINOUT	Multiple function routine called by ARRNGE type programs to process sort items.

ALRPOL Calculates the reciprocals of Lorentz and polarisation factors.

ANGDIR Calculates D3 or 4-circle angles from direction cosines.

BIGGAM Deals with gammas whose value is greater than one, in SORGAM. There is also the entry SETGAM to set up the calculation.

EXTINC Multi-entry routine to deal with all aspects of single crystal extinction corrections.

FGAMMA Calculates gamma, the ratio of magnetic and nuclear scattering, and its standard deviation. FGAMMS does this in the presence of spin-independent multiple scattering.

GAMEX Calculates an extinction corrected gamma from a flipping ratio.

GETDC Calculates direction cosines of the incident and diffracted beams, used by absorption correction type integrals.

QARRIN In the Library, simply a dummy routine. If the user needs some special new input format for ARRNGE type main programs, he provides a new version of QUARRIN.

STATIS Calculates means and other statistics for a set of measurements of a single quantity.

C Utility

INPUAR Reads the data from files made by ARRNGE and other similar programs.

3. MANIPULATION OF REFLECTION INDICES

A Setting Up

PRMTIV A specialist routine for use in generating h,k,l indices where the natural stepping vectors do not define a primitive cell.

SETGEN Sets up the generation of a complete set of reflection indices.

B Crystallographic

GENMUL Gives next useful set of h,k,l and multiplicity, scanning the asymmetric unit of reciprocal space.

GETGEN Gives the next useful set of h,k,l scanning the asymmetric unit of reciprocal space.

INBOX Determines whether the given reflection is inside, on, or outside the reciprocal space asymmetric unit.

C Utility

INDFIX Converts 3 real items in H to be integers in K.

INDFLO Converts 3 integers in K into floating point reals in H.

MATCH Checks to see whether two sets of reflection indices match.

MATCHF Checks to see whether two sets of floating point reflection indices match.

MULBOX Tests indices for being in the asymmetric unit, and gives multiplicity.

4. STRUCTURE FACTOR CALCULATIONS

A Setting Up

ATOPOS Reads and interprets all given **A** cards.

INPUTA Reads an **A** card and prepares it for further processing.

INPUTF Reads and partially interprets an **F** card.

INPUTT Reads and interprets one **T** card.

INPUTW	Reads one W card as far as atom label and word.
RADFUN	Reads coefficients for the expansion of an atomic wave function.
SETANI	Reads T cards to set up for calculation of anisotropic temperature factors.
SETFOR	Sets up data for scattering or form factor calculations.

B Crystallographic

ANITF	Forms the contribution to the anisotropic temperature factor on an atom N from indices H.
ATOMS	Makes a real space unit cell full of related atomic positions.
CONATF	Produces for a single coefficient of an anisotropic temperature factor, its conversion factor from the internally used betas, in order to communicate with the user.
FCALC	Calculates the COMPLEX nuclear structure factor for the reflection H.
FORMFA	Calculates form or scattering factors.
FORMFC	Calculates form factor integrals from radial wave functions.

5. FOURIER CALCULATIONS AND MAP PLOTTING

A Setting Up

INPUTM	Reads and interprets all M cards.
MAJUST	A specialist routine used in the input of a Crystal Data File needing previously stored Fourier maps.
NEXCON	Sets up the next contour value to plot for Fouriers.
SETFOU	Sets up data for Fourier map calculations.
USYM	Transforms all the symmetry operators by pre- and post-multiplying them by U, the orientation matrix for a Fourier map.

B Crystallographic

ATMPLO	Plots atom positions on a map.
ATOGEN	Generates a set of equivalent positions.
ERRMAP	Calculates the standard deviation of the density in a Fourier map.
FORIER	Controls Fourier calculations.
FOUR1D	Calculates a Fourier along a general line.
FOUR1Z	Calculates one layer of Fourier sum : a section if 3D, a projection if 2D, or a bounded section if "4D".
FOURGP	Calculates a Fourier on a general plane.
GETMAP	Retrieves one Fourier map previously filed using SAVMAP.
MAPCON	After a Fourier map has been plotted, sends to the plotter the list of contours which were plotted, with a frame.
MAPDRW	Draws an unframed contour map, in predetermined place.
MAPFRA	Draws a black frame round a potential contoured map, adding the labels X Y and a 1A scale.
MAPKEY	Plots a key to the atoms found by ATMPLO.
MAPTIT	Writes a title over a plotted map, with a frame.
PLTTXT	Plots a block of explanatory text under a plotted Fourier map.
PRNTMP	Prints out a Fourier projection or one layer of a 3D Fourier.
QFOUIN	In the Library, simply a dummy routine. If the user wishes some special new input format for Fourier routines, he provides a new version of QFOUIN.
READMP	Reads into the array DENS a map previously written to file, unformatted.
RESOL	Calculates a resolution function for use with Fourier inversion.

SAVMAP Writes one layer of map to given file, plus information for later retrieval.

SYMEQU Generates new indices and a phase, in Fourier calculations.

TBOUND A specialist routine used duing the plotting of atomic positions in routine ATMPLO.

C Utility

CHOOSF Chooses modulus and phase for a particular Fourier.

DOSIDE A specialist routine for contour plotting, to help to decide where a contour crosses the side of a square.

FOUINP Reads one data item for a given type of Fourier, in a given format.

6. LEAST SQUARES REFINEMENT - GENERAL

A Setting Up

CELREL Transfers any relations which exist between cell parameters from their own COMMON to the general contraint/fixing COMMON.

FIXPAR Records an instruction to fix a LSQ parameter.

FIXREL Takes a temporary set of fix/relate information and adds it to the permanent information.

FUDGIN Interprets all **L FUDG** cards.

F2VAR8 Records varying information for a particular family 2 (structure) parameter.

GEOMIN Reads **L** cards for bond slack constraints.

IICD1 Interprets basic **I** cards to drive any LSQ.

LLSCAL Multiple entry routine which deals with scale factors in Least Squares.

LLTFAC Multiple entry routine which deals with overall isotropic temperature factors in Least Squares.

LSETSF Sets up vocabulary for a specific LSQ problem involving structure factors.

LSETUP Sets up a specific LSQ problem; copies the vocabulary to standard COMMON.

RDFV Reads all the user's **L FIX** and **L VARY** cards in sequence.

RDRELA Reads and interprets all user-supplied **L RELA** cards for constraints.

RELPAR In the setting up of LSQ applications, relates two parameters by a simple linear relationship.

RELSM3 Forms a complete set of relations imposed by symmetry on the given 3 parameters of a LSQ application.

RELSM6 Forms a complete set of relations imposed by symmetry on the given 6 parameters of a LSQ application.

SCLCHN A specialist routine used during the setting up of constraints on LSQ variables. Scales a chain of connected variables by the given constant.

VARMAK Makes variables for a LSQ cycle from given fix/vary and constraint lists.

VOCAB Adds a given set of vocabulary and meanings to the Least Squares total vocabulary.

B Crystallographic

CELDER From h,k,l calculates d* squared and its derivatives, and sets SSQRD.

F2NEW Outputs a new LSQ family 2 (structure parameters) card (for **A**, **T** or **F** cards).

F2RELA Collects all structure factor type constraints implied by the symmetry.

F2SHFT Applies a shift to a particular family 2 (structure) parameter.

GEOMLS Calculates bond lengths and derivatives for geometrical slack constraints.

LFCALC	Calculates a nuclear structure factor and its derivatives.
MATCEL	After a cycle of LSQ gives variance-covariance matrix for the cell quadratic products A*, B* etc in both real and reciprocal space, and the same for the cell sides and angles a, b, c, alpha, beta, gamma.
OTPUTI	Outputs a new **I** card after a LSQ refinement, updating the cycle number.
QLSQIN	In the Library, simply a dummy routine. If the user wishes some special new input format for Least Squares routines, he provides a new version QLSQIN.
RFACS	A multiple entry routine to deal with all aspects of R Factor calculations and statistics for single crystal observations, and
XYZREL	Collects all position parameter constraints implied by the symmetry.

C Utility

ADDCON	Adds a constraint to the list held in LSQ programs.
ADJUST	Applies a (possibly fudged) shift to a given LSQ parameter.
CELNEW	Writes out a new **C** card after cell parameter refinement.
CELSDP	Prints the esds of real cell parameters after a cycle of refinement.
CELSHF	Applies a shift to a cell quadratic product.
DEPRIN	Decodes an integer which describes the frequency of LSQ printing required, and outputs this frequency.
ELEMAT	Gets a matrix element from the triangular LSQ matrix.
FETSHF	Fettles a shift and esd for printing and counts, in the application of shifts in LSQ.
FIXVAR	Adds a request to fix (or vary) a parameter to the lists held in setting up LSQ environments.
FUDGET	Reads a fudge factor from an **L FUDG** card having already read a parameter specification.
GEOMCO	Multiple entry routine for geometric slack constraints.
KPAK	Pack a LSQ parameter specification on to integer.
KSAME	Decides whether two LSQ parameter specifications are the same, allowing wild card elements.
KUNPAK	Unpacks a LSQ parameter specification from single integer.
KWHOLE	Decides whether KK is a whole packed parameter specification, or whether there are wild card elements.
MATCOR	After a Least Squares cycle, prints correlations from the inverse matrix.
MATINV	Inverts the matrix in ALSQ, of which the upper triangle only is held.
MATSET	Sets up pointers into a Least Squares matrix, and clears the matrix and the corresponding right hand side vector.
MATSHF	From an inverted Least Squares matrix, calculates shifts in basic variables.
MATTOT	Add in contributions to LSQ matrix and RHS for one observation.
NEWCD	Opens a file on which to write a new Crystal Data File after a Least Squares Refinement. In VAX VMS terminology, the file has extension .CCN at RAL or .CRY at ILL.
PARNAM	Obtains the printing name of a LSQ parameter.
PARRD	Reads a LSQ parameter specification from a given card at given point.
PARRUN	Controls the cycling over all parameters in LSQ (not for Profile Refinement).
PRBLOK	Prints a block of shifts in parameters all relating to the same atom in LSQ applications involving structure parameters.
PRIVAR	Prints a list of basic variables, and constraint relations, for LSQ.
PRIWRD	Finds the name of the packed (possibly part) LSQ parameter from the built-in table of parameter names.

PRNCYC Decides whether printing (of various different quantities in LSQ) is needed during the current LSQ cycle.

PUNPAK Unpacks a parameter specification from single integer.

RELATE In LSQ programs, converts a vector of derivatives wrt variables into the vector of derivatives wrt basic variables.

SERROR Called after VA05A to give the standard deviations of the parameters.

SHFESD During the application of LSQ shifts, calculate the shift and ESD for a "redundant" variable.

SYMFIX Says whether a LSQ parameter is fixed by symmetry or not.

TBLFND Looks for an A4 NAME in every table it can find, trying to identify it as part of a LSQ parameter name.

VA05A Minimises the sum of squares of given functions without requiring the explicit calculation of derivatives.

WGHTLS Performs various operations to do with weights for LSQ, either for PR or simpler applications.

WGHTSF Deals with weights of LSQ observations for single crystal.

7. LEAST SQUARES REFINEMENT - SPECIFIC

A Setting Up

DFLTMG Called as a substitute for DEFALT via VARMAK, giving default fix/vary for otherwise unspecified parameters for magnetic structures.

DFLTMP Called as a substitute for DEFALT out of VARMAK, giving defaults fix/vary for otherwise unspecified parameters of multipole refinements.

DFLTSF Called as a substitute for DEFALT out of VARMAK, giving default fix/vary information for structure parameters.

DFTRUE Called as a substitute for DEFALT out of VARMAK in main programs, to vary an otherwise unspecified parameter.

INOBGR Performs a preliminary pass of observations for GRLSQ, (group refinement, integrated intensity LSQ).

INPLSF Reads in (nearly) all L cards for the particular LSQ type "SF", for single crystal work.

MPOVAR Records whether each multipole parameter is fixed or varied.

PARSDS Collects together all parameter fixing and varying information for LSQ refinement of cell parameters using d-spacings.

PARSFW An older routine to set up variables from parameters for FWLSQ.

STLSFW Sets up the main program FWLSQ to perform LSQ on Forsyth & Wells scattering factor coefficients.

STLSSF Sets up any main program which does single crystal structure factor LSQ.

VARSDS Makes variables for cell parameters for d-spacing LSQ.

VARSMG Records pointers for all variables in magnetic structure-factor LSQ.

VARSMP Records pointers for all variables in multipole LSQ.

VARSSF Records pointers for all variables in structure-factor LSQ.

B Crystallographic

APSHDS Applies shifts for reciprocal cell parameters during d-spacing refinement.

APSHFW Applies shifts for Forsyth & Wells scattering factor coefficient refinement.

APSHMP Applies shifts to all variables in multipole LSQ, and prints the results.

APSHSF Applies shifts to all variables in single crystal structure factor based LSQ, and prints the results.

APSHT2 Applies shifts for reciprocal cell quadratic products and a zero point during 2theta refinement.

CALCDS Calculates d* squared, and its derivatives wrt reciprocal cell quadratic products.

CALCFW Calculates the Forsyth & Wells exponential function which approximates to observed scattering factor curves.

CALCGR Gives the calculated function for grouped single crystal Least Squares.

CALCMP Makes the calculated function and derivatives for single crystal LSQ with multipoles.

CALCSF Makes the calculated function and derivatives for single crystal LSQ.

CALCMG Calculates GCALC and its derivatives for structures which may have mixed magnetic and nuclear reflections.

CALCT2 Calculates a function to match an observation for 2theta-type refinement of reciprocal cell parameters and a zero point.

CALPOL Calculates scattered polarisations and their derivatives for structures which may have mixed magnetic and nuclear reflections.

CONVMP Converts between user values and LSQ parameters for multipoles.

JMPOL Writes to unit NEWIN all **J MPOL** cards after a multipole refinement.

LMPCAL Calculates a structure factor and its derivatives using a multipole description of the form factors.

NWINDS Outputs to unit NEWIN a new input dataset at the end of a refinement of cell parameters using d spacing values.

NWINFW Writes out a new Crystal Data File for main program FWLSQ.

NWINMP Writes to unit NEWIN a replacement Crystal Data File after multipole LSQ.

NWINSF Outputs a replacement Crystal Data File after single crystal refinement.

NWINT2 Writes out a new Crystal Data File for main program T2LSQ.

PARSSF Collects all parameter fix and vary information for single crystal LSQ.

PRMBLK Applies shifts to multipole parameters, in both program and user units.

CALMMP Make calculated function and derivatives for single crystal LSQ with magnetic multipoles.

8. CRYSTAL GEOMETRY

A Setting Up

ADDANG Finds an angle in the tables for geometric constraints, or adds it if absent.

ADDATM Finds an atom in the tables for geometric constraints, or adds it if absent.

ADDBON Finds a bond in the tables for geometric constraints, or adds it if absent.

ADDTOR Finds a torsion angle in the tables for geometric constraints, or adds it if absent.

RDANGL Reads the specification of a bond angle, by reading the names of 2 intersecting bonds; makes the third bond involved.

RDATOM Reads the specification of an atom for slack constraint purposes.

RDBOND Reads a specification of a bond, by reading the names of the atoms at each end.

B Crystallographic

ANGLST Lists all angles at one source atom made by a given list of bonds.

BNDLST Writes out and saves a list of bonds from one atomic position.

BONCOS Given 3 bonds forming a triangle, calculates the angle opposite the first, its sine and cosine, and its derivatives wrt all 3 bonds.

BONDA Calculates the angle between two bonds.

BONDER Calculate a bond and its derivatives for slack constraints.

BONTRI Given two bonds with a common atom, completes the triangle and identifies the atom.

INCELL Sets P to be the position in the central cell equivalent to X.

XROOT Finds the symmetry operations which take the given source atom into the given coordinates.

XTRANS Transforms a given atomic position by given symmetry operator and lattice and cell translations.

C Utility

ATSPEC Makes the 16-character specification of a symmetry related atom from its packed specification.

9. MATHEMATICAL FUNCTIONS

A Setting Up

SPLINE Sets up a cubic spline to fit a curve.

C Utility

ALNINT Performs linear interpolation, suitable for profile backgrounds etc.

BJ Calculates Bessel functions of order 1 or 2.

BRILL Returns the value of the Brillouin function.

CGAMMA Solves the quadratic equation for gamma in terms of the flipping ratio. The ENTRY CGAMMS is for when there is spin-independent multiple scattering.

DROT Calculates the matrix D(i,j) for the Euler rotations alpha, beta, gamma of the eigenfunctions of angular momentum l.

ERFNC Calculates the error function accurate to 3E−7, for + and − X.

EXPINT Calculates an exponential radial integral.

FACT Calculates factorial K.

FF01A Modified Harwell routine for zero order Bessel functions.

FF02A Modified Harwell routine for first order Bessel functions.

FRACT Forms the fractional part of a real number.

FT01A Modification of Harwell Fast Fourier Transform routine.

JTERMS Calculates the terms in the 2D averaged spherically symmetric form factor summation which depend on S, K, and R only.

NB01A Harwell routine NB01A.

SPHARM Calculates spherical harmonics.

SPLINT Evaluates a cubic spline given spline values and first derivative values at the given knots.

STERMS Calculates a term in the spherically symmetric form factor summation, for a 3D averaged form-factor involving Sk and r only.

TB02A Interpolates in non-equal interval table.

TQLI Performs the QL algorithm for eigenvalues and vectors of a real symmetric matrix previously reduced to tridiagonal form.

TRED2 Performs the Householder reduction of a real symmetric matrix to tridiagonal form.

VECOUP Calculates Clebsch-Gordon vector coupling coefficients.

WTMEAN Multiple entry routine for the calculation of weighted averages.

10. TRIGONOMETRY

C	Utility
ARCCOS	Calculates an arc cosine.
DEGREE	Converts from radians to degrees.
RADIAN	Converts from degrees to radians.
SINCOS	Calculates sin from cos or vice-versa.
TRIAN1	Applies the cosine formula for the solution of spherical triangles.
TRIG	Sets up cos(nx) and sin(nx) for a range of n by recursion.

11. TESTS

C	Utility
BINDIG	Tests for the presence of a given binary digit within an integer.
EQPOS	Checks whether the given atom position already occurs in a given list, and adds the new one if not.
EQPPOS	Checks whether the given position vector is related by lattice translation to one already in the given list.
EQRLV	Checks whether vectors differ by a reciprocal lattice vector.
EQVEC	Finds a given vector in given table of vectors, or adds it as a new one.
EXCLD	Determines whether a number occurs within any of a set of given ranges.
GMSAME	Determines whether one vector is the same as another, to a given tolerance.
IATOM	Identifies an atom name in a given list.
ISCAT	Searches for a potential scattering factor name in the table.
LMATCH	Matches an A4 item in given table, adding it if it is not there already.
MINIM	Finds the position and value of the minimum in a list of integers.
NCFIND	Searches for a particular word in a table of words.
NFIND	Searches for integer N in a table.
PARITY	Finds out whether N is odd or even.
RANGE	Puts a number into a given range.
REJECT	Decides for ARRNGE-type main programs whether the record number of a reflection occurs in a list of those to be rejected.
SAID	Decides whether the character variables INCHAR and WANT are the same, ignoring any distinction between upper and lower case.
TESTOV	Tests a floating division for potential overflow.

12. MATRICES AND VECTORS

C	Utility
CGMADD	Sets matrix C = matrix A + matrix B, where A, B and C are all COMPLEX.
CGMEQ	Sets COMPLEX matrix B equal to COMPLEX matrix A.
CGMPRD	Multiplies together two COMPLEX matrices.
CGMSCA	Multiplies every element of a COMPLEX matrix by a COMPLEX scale.
CGMSUB	Sets COMPLEX matrix C = COMPLEX matrix A − COMPLEX matrix B.
CGMUNI	Clears a COMPLEX square matrix to contain the unit matrix.
CGMZER	Clears to zero a COMPLEX matrix.
CMCONJ	Gives the conjugate of a COMPLEX matrix.
CMIMAG	Gives the imaginary parts of a COMPLEX matrix.

CMREAL	Gives the real parts of a COMPLEX matrix.
CMRSCA	Multiplies every element of a COMPLEX matrix by a REAL scale.
CRMPRD	Performs COMPLEX by REAL matrix multiplication.
CRSCLP	Finds the COMPLEX scalar product of a COMPLEX with a REAL vector.
C1MSCA	Multiplies every element of the REAL matrix A by a COMPLEX scalar.
DETER3	Forms the determinant of a 3 x 3 matrix.
GMADD	Sets matrix C = matrix A plus matrix B.
GMEQ	Sets matrix B = matrix A.
GMINV	Inverts matrix A into matrix B.
GMNORM	Normalises the rows of a matrix.
GMTRAN	Transposes a JJxII matrix A into B.
GMPRD	Sets matrix C = matrix A x matrix B.
GMREV	Reverses the signs of the elements of an NI x NJ matrix.
GMSCA	Multiplies every element of the matrix A by the scalar SCALE.
GMSUB	Sets matrix C = matrix A − matrix B.
GMUNI	Writes a unit matrix into the square matrix A.
GMZER	Clears to zero the matrix A.
JGMADD	In integers, sets matrix C = matrix A + matrix B.
JGMEQ	Equates an integer matrix to a given integer matrix.
JGMPRD	In integers, sets matrix C = matrix A x matrix B.
JGMREV	Reverses the signs of the elements of an integer matrix.
JGMSUB	In integers, sets matrix C = matrix A − matrix B.
JGMZER	Clears an integer matrix to zero.
MB11A	Inverts a rectangular matrix whose order is the smaller dimension; used by the Harwell refinement routine VA05A.
RCMPRD	Performs the multiplication of a COMPLEX by a REAL matrix.
RSCALP	Forms the scalar product of two COMPLEX vectors.
SCALPR	Forms the scalar product of two orthogonal vectors (or of one from real space with one from reciprocal space).
SID	Solves a set of simultaneous linear equations.
SUMVEC	Calculates the sum of elements of a real array.
TRANSC	Replaces a COMPLEX square matrix by its transposed conjugate.
TRANSQ	Replaces a REAL square matrix by its transpose.
TRINV3	Replaces a 3x3 matrix by the transpose of its inverse.
UNIVEC	Replaces a vector by a parallel unit vector, and gives its length.
VECPRD	Calculates the vector product of two 1x3 vectors.

13. CCSL INPUT/OUTPUT ROUTINES

A Setting Up

PREFIN	Makes the Crystal Data File readable in a random order by writing it to a scratch file.

C Utility

ASK	Writes a message on unit ITO expecting an interactive answer.
BIGCHA	Writes on unit LPT up to 9 characters in large letters.
CARDIN	Finds the record number IDEN in the DIRECT ACCESS file on unit IO10, which is a copy of the Crystal Data File.

CDFIN	Reads in one Crystal Data File and copies it to a DIRECT ACCESS unit number IO10, starting at record number ID.
CDSCAN	Finds the next card which starts with the letter given in CH, and has then a word which is one of the collection given in WORDS.
CENTRE	Writes a centred message on a given unit, preceded by N empty lines.
CLOFIL	Closes the FORTRAN unit LUN and returns the CCSL unit to the pool.
ERRATM	Writes an error message to say that the given name is not an atom name; there is a choice of subsequent action.
ERRCH2	Write an error message which involves a given WORD between 2 messages; there is a choice of subsequent action.
ERRCHK	(Possibly increases and) checks a value, giving if appropriate an error message; there is a choice of subsequent action.
ERRIN2	Writes an error message which involves a given integer INT between 2 messages; there is a choice of subsequent action.
ERRMES	Writes an error message, with choice of action on exit.
ERRRE2	Writes an error message which involves a given real X between 2 messages; there is a choice of subsequent action.
FETTLE	Decides the field width and fractional part of real number in order to print it.
FILNOM	Returns the name of the file on FORTRAN unit LUN.
FILPRO	Makes sense of general file names, mainly for the VAX under VMS.
FINDCD	Searches for a card starting with letter CH and with WORD in columns 3-6.
GENNAM	Finds all the starting letters of an atom name.
INCREM	Increments a file name.
INITIL	Initialises the CCSL system.
INPUTI	Gathers information from a user's interactive instruction I card.
INPUTN	Deals with the N card giving the Crystal Data File title.
INTCHR	Converts digits to characters, either left- or right-justified.
INTDIG	Unpacks an integer into its individual digits.
IPOPE	Machine specific routine to interpret error codes during file opening.
LENG	Determines the length of a text string, omitting trailing spaces.
LENGT	Determines the length of a character variable, omitting the final spaces.
LETTER	Determines whether a character is a letter.
MAKNAM	Makes an A4 name from the given character and the digits of the given number.
MESS	Writes a message on the given unit, preceded by N empty lines.
NCHINT	Converts an ASCII character into an integer
NDIGIT	Identifies a character as a digit or not.
NOPFIL	Opens a file on a FORTRAN unit for the first time in this job.
NSYMBL	Finds whether the character I is one of the symbols recognised by the system.
NUMA1	Prepares a number for writing, probably on a plotter.
NUMDEN	Converts a real number to the numerator and denominator of a fraction.
ONCARD	Finds a card which starts with the given letter and word, and reads one number from it.
OPNFIL	Opens file L according to requirements given in M; L may be preset.
PRILIS	Prints a list of real numbers held in an array, 5 per line.
PRIPLN	Given the normal to a plane face in A, prints the equation of the plane.
PUTPAR	Distributes parameters read by RDNUMS amongst individually named variables.

RDDATA Reads in free format h,k,l (possibly floating point) and a list of values, allowing for a possible title.
RDINTG Reads an integer in free format from a character string.
RDNUMS Reads all the numbers on a line in free format.
RDREAL Reads a real number in free format from a character string.
RDWORD Reads the next word from a character string.
RDWRDS Reads all the words on a line from column 3 onwards.
TESTP Puts a given heading at top of every printer page, counting lines.

14. GRAPHICAL OUTPUT

A Setting Up

GETSCL Chooses a sensible scale for a graph.
PINITL Initialises the system in order to make graphical output.
STPLOT Sets up the plotting of maps; fits elements of a picture together.

B Crystallographic

PICMOV If plotting to a screen (e.g. Tektronix), do nothing; if to a plotter, move to the next picture.
PLCONV Performs the transformation of coordinates between different plotter spaces.
PLOTCT Plots a single contour throughout a given array.
PLOTIT Plots the graph of given vector y against x, with esds.
PLOTO A multi-purpose graph-drawing routine.
PLTRIN Defines a new coordinate transformation for plotting.
PMTINV Specialist routine to invert a 2x3 matrix, such as those which transform plotting coordinates from one space to another.
PMTMUL Specialist routine to multiply together two 2x3 matrices, such as those which transform plotting coordinates from one space to another.
SPCSET Defines the space in which coordinates will be given for plotting.

C Utility

ARROW Draws an arrow centred at X,Y in the current space.
CIRCLE Draws a circle of given radius and centre, with various options.
DPLOT Does plotting in the current coordinate system.
FRAME Draws a rectangle in the plotting context.
KANGA1 Moves the plotter pen (or equivalent) to X,Y in current coordinates.
KANGA2 Writes on a plot a string of characters, or simulates this in order to measure the length of the string.
KANGA3 Plots a special symbol.
PIGLET A complete set of device-specific plotting commands.

15. LOGICAL OPERATIONS

C Utility

BITSET Sets or tests a single bit in a word.
GETSQ Works out where a contour crosses the sides of a square of function values.
ITPOS Sets ITPOS=a single bit (a one), in position IPOS, counting from the right.
LOCBIT A specialist routine for contour plotting, which finds the "next" bit in the bit-map, removes it, and indicates where on the picture it was.

LOGAND Performs logical **and** on 2 whole integers.

LOGOR Performs logical **or** on 2 whole integers.

MAKEBM A specialist contour plotting routine which makes a bit map to show where the contours are.

16. MISCELLANEOUS

C Utility

BATCH Signals batch running, not interactive.

DUMMY Does absolutely nothing; used as a default in routine calls.

FLIP Exchanges the integers I and J.

JFIX Rounds a real number to be an integer.

NPACK Deals with the packing and unpacking of up to 10 integers in/out of one integer.

NTICK Advances its argument by 1 and sets the function to that value also.

RESHUF Reorders a real array, given a parallel pointer array out of SORTX.

SORTN Sorts pointers to an integer array using Heapsort.

SORTX Sorts pointers to a real array using Heapsort.

17. MAGNETIC STRUCTURE FACTORS

A Setting Up

INPUTQ Reads individual **Q** cards.

LOGMAG Sets mnemonic logicals from the type of magnetic structure.

MAGCNC Finds magnetic constraints in a non-least squares calculation.

MAGCNL Does the fixing associated with constraints found by MAGCON and PSICON.

MAGCON Finds and reports the symmetry constraints on magnetic parameters.

PROPER Determines whether the satellites generated by the propagation vector PROP have integer indices, and generates its *star*.

PSICON Determines the constraints on the phase factors in helimagnets.

READRT Reads whatever follows on a **W** *atom-name* **ROTN** card.

SETFCM Calls the routines needed to set up magnetic structure factor calculations (or non-magnetic also).

SPHPOL Sets up spherical polar spin directions for magnetic structures.

B Crystallographic

CENTRO Executes the action of the centre of symmetry on a term in the magnetic structure factor for helimagnetic structures.

DOMAG Multiple entry subroutine to deal with parameters for magnetic structures on **Q** cards (but not their fixing/varying).

FMCALC Calculates magnetic interaction vectors and magnetic structure factors.

LMCALC Calculates a magnetic structure factor and its derivatives.

LMAGPR Gives the fix/vary information for one of the family 2 parameters for magnetic atoms.

MAGABS Tests for systematic absence of magnetic reflections.

MAGDIR Calculates various geometric corrections for various magnetic states. The ENTRY ENTMAG sets up the COMMON.

MAGDOM In Least Squares Refinement with magnetic scattering, forms the matrix needed for derivatives of Q with respect to a spin direction.

MAGSYM A routine with 4 named entry points, MAGSYM, MELIN, NELIN and ROTMAG, to deal generally with magnetic symmetry. MAGSYM sets magnetic symmetry, MELIN puts in an operator for a generator, NELIN puts in non-symmetric rotation and ROTMAG rotates using a magnetic operator.

MAGVAR Records the initial fixing, or subsequent varying of magnetic parameters.

MOLORB To read molecular orbital wave-functions from **W** *atom-name* **FUNC** card.

SATFND Finds the indices HS of the satellite equivalent to H, offset by the propagation vector AKSTAR(1:3,1) from a reciprocal lattice point.

SATGEN Generates a set of magnetic satellite reflections.

SPHELI Imposes perpendicularity on the two components of a helix.

MFCALC For a magnetic structure, calculates the structure factor and various related quantities.

C Utility

ASPHFF Calculates an aspherical form factor for a cubic space group.

GENMAG Generates the next set of magnetic h,k,l, scanning the asymmetric unit.

MTPROD Puts entries into the table of time inversion operators.

18. MULTIPOLES

A Setting Up

INPUTJ Reads individual **J** cards.

MPCON Finds the symmetry constraints on multipoles.

MPFORM Finds out which radial form factors to apply to which atom and L value.

ORTFUN Finds the best set of orthonormal functions compatible with symmetry, based on the users input, and hence defines the multipoles to refine.

PFSET Directs the reading of **J** and **W** cards for multipole calculations.

REAORB To change the orbital basis from Ylm and Yl-m to (Ylm +– Yl-m).

B Crystallographic

MF5ADD A specialist routine to deal with the refinement of multipoles. Converts between the two methods of addressing LSQ family 5 (multipoles).

PFCALC Calculates the COMPLEX nuclear structure factor for the reflection H, using a multipole expansion of the form factor.

PFORMF Calculates radial form factors for multipole refinement.

LMMPCA Calculates a magnetic structure factor and its derivatives using multipole description of the form factors.

C Utility

NAMPOL Creates the label for a given multipole.

Chapter 3

THE CRYSTAL DATA FILE

INTRODUCTION

Users' input to CCSL comes in two main types. One is a *crystallographic data file*, or Crystal Data for short, which is required by most CCSL jobs. Various other input files may contain, say, reflection data for the Fourier routines, or profile intensity data, in fact, any other set of data which the user wants to be read by CCSL.

In this chapter we first describe the Crystal Data. Then, because the file is interpreted by CCSL and not directly by FORTRAN, the routines which do this are introduced in some detail. Anyone wishing to write CCSL programs should find these routines useful. Details of exactly what is on the Crystal Data cards are given at the end of the chapter, in alphabetical order.

Other types of data input are described in Chapter 4.

INPUT OF THE CRYSTAL DATA FILE

The file holding the Crystal Data contains a sequence of records, each of which will initially be read in the FORMAT (A80), and which, for historical reasons, are referred to as *cards*. They should be held in a named file; PREFIN, the routine which reads them into the system, will ask for this name.

The first character of every card must be a letter; PREFIN will stop when it encounters a card not starting with a letter (or when it attempts to read beyond the last card in the file). It copies the Crystal Data to an internal unit, from which cards may be read repeatedly as necessary. Empty lines are ignored everywhere in the Crystal Data.

For the first letter on a card, upper and lower case letters are treated alike. In general, though, they are distinct. The character in position 2 on the card, immediately following the letter, is at present ignored, but it should be typed as a space.

The remaining 78 character positions depend on the initial letter. As far as possible the initial letters are mnemonic, like **C** for cell parameters, **A** for an atomic position, etc.

CCSL FREE FORMAT INPUT

In order to avoid FORTRAN fixed format READ statements, CCSL contains a number of *free format* routines. These routines are used to read information from the Crystal Data. Each card is read initially as a string of (up to) 80 characters. These are then interpreted by CCSL routines like:

RDINTG read next *integer*,

RDREAL read next *real* number,

RDWORD read next *word*.

DEFINITIONS FOR FREE FORMAT ITEMS

We define three types of item to read: integers, reals and words.

An *integer* is a (possibly signed) sequence of digits.

A *real* is a (possibly signed) sequence of digits, possibly containing one decimal point, or possibly containing one slash.

A *word* is a sequence of characters starting with a letter.

All three items are terminated by a space or the line end; integers and reals are also terminated by a comma. Surplus spaces are allowed.

EXAMPLES

Integers	`6 10 -4 99999,42, 3`
Reals	`12.34 92 5/12 -6.78904, -3,`
Words	`Fred Ca14 This-is-one-long-word`
Not integers	`3. 12/4 -4-3`
Not reals	`5./12 4.E2 67.89*`
Not words	`1Fred *JIM`

(Note that the "real" format including an exponent is not implemented).

SPECIAL WORDS

Many of the routines in CCSL recognise special *words* of up to four characters which are found in the Crystal Data. Some commonly used types are:

atom-label	A *word* used to identify a particular atomic site;
scattering-factor-label	A *word* used to identify a particular form factor;
CCSL-word	A *word* whose meaning is defined within CCSL.

GENERAL REMARKS ON CRYSTAL DATA CARDS

Cards which have the same initial letter *must* be placed sequentially, but cards within a group with the same initial letter may occur in any order. They are usually identified by a *CCSL-word* of up to 4 characters which follows the initial letter on the card (e.g. **CONT**, **PLOT** for Fourier maps or **FIX**, **VARY** for Least Squares).

In order to make input more compact, certain cards may (after the initial letter) contain sequences of *CCSL-word/number pairs*, as on the **M** card:

```
M DTYP 4 FTYP 2 SCAL 1000 DELT 0.3
```

The most common example of this is the **I** card described below. The facility extends to **M**, **D**, **G** and **L** cards, but *only* for those *CCSL-words* which expect a *single number* after them.

The first word on some types of card (e.g. **A** (atom position), **F** (form/scattering factor) and **J** (multipole) cards) is an atom-label or scattering-factor-label identifying a particular atom or form-factor. The atom-label or scattering-factor-label may itself be followed by a CCSL-word, as for instance on **Q** (magnetic structure) and **W** (wave-function) cards.

Almost all cards fit into the scheme in which the first letter and a following *CCSL-word* are used for identification, although none of the cards used in the simple example in Chapter 1 does.

The **N** (name of job) card uses all 79 characters as a title for output; the **S** (symmetry) cards simply give one symmetry operator each.

The **I** cards give instructions about the details of a particular run, and may change from run to run, whereas, say, an **S** card does not. An example is:

```
I MCOR 80 NCYC 5 CYC1 4
```

which, when read by a Least Squares Refinement program, says "for this particular run print only correlations above 80%, and do 5 cycles of refinement, labelling the first 4". The general format is a sequence of *CCSL-word/number pairs*. The words and numbers may be interpreted in any way the user wishes, so **I** cards may drive a variety of different main programs depending on how they are interpreted.

ROUTINES TO READ SPECIFIC CRYSTAL DATA CARDS

Most cards have a corresponding simple input routine INPUTx to read in one card starting **x**. Thus, INPUTC reads the **C** card and INPUTE reads the **E** card. There are often also more complicated input routines which then deal with the information on the card. So, e.g., SYMOP calls INPUTS for every **S** card given; INPUTS produces a rotation matrix and a translation vector, doing a little preliminary checking on the way, then SYMOP adds this new operator to its existing list and does further necessary generation of operators.

For some initial letters there is an obvious setting-up routine to read all cards starting with that letter. ATOPOS reads all **A** cards via INPUTA, checks them and does further setting up (e.g. detecting special positions) which the user would almost certainly want every time he inputs **A** cards.

Other cards are of more general application. For the **I** cards just described, INPUTI reads in all the *CCSL-word/number pairs* but does not interpret any of the words, so other setting-up routines (like SYMFRI to read **FRIE**, or IICD1 to read the variations from the defaults required in a Least Squares Refinement) are free to pick out the words they understand.

It is not essential to read a particular card using the existing CCSL routine. If the user wishes to put different numbers on, say, an **E** card for a new form of extinction correction, he may do this and write his own version of INPUTE to read them. He will probably do this by taking the existing INPUTE and modifying it. If he presents his new INPUTE to the FORTRAN compiler, following his main program and before it scans the Library, it will accept his and ignore the one in the Library. There is further discussion on writing one's own program in Chapter 7.

CRYSTAL DATA CARD DESCRIPTIONS

The card descriptions which follow indicate the data required on each type of Crystal Data card, and define how it should be presented.

INITIAL LETTERS

Most initial letters have been assigned. **Y** and **Z** introduce comment cards. **I** is for general instructions, and **X** has been deliberately left for the user. The remaining assigned letters are used for cards containing specific items of crystallographic data.

SYSTEMS OF AXES

Three systems of axes are used in different places in CCSL. They are:

1. the direct space crystallographic axes with x, y, z parallel to the cell edges **a, b, c**. They are used for atomic positions, zone axes etc.
2. The reciprocal space axes with x, y, z parallel to the reciprocal cell edges **a*, b*, c***. They are used for reflection indices etc.
3. A standard set of orthogonal axes with:
   x parallel to **a*** (100)
   z parallel to **c** [001]
   y making up a right-handed set.
   These axes are used to describe the crystal form and for spin directions. They will be referred to as *CCSL orthogonal axes.*

§ § x x § §

A cards
Atomic positions etc.

ITEMS READ:

A, *atom-label* , x, y, z, isotropic temperature factor, *scattering-factor-label* , site occupation factor. The last two numbers are optional.

ASSUMPTIONS MADE:

The scattering-factor used will be that labelled with the initial letters (terminated by the first non-letter) of the *atom-label* , unless a specific label is given after the isotropic temperature factor.

The site occupation factor is = 1 **unless** a non-zero number is read after the (possible) scattering-factor-label.

DETAILS:

The *atom-label* is 1,2,3 or 4 characters, the first of which is a letter.

The *scattering-factor-label* is not needed at all if the scattering-factor for this atom is labelled (on an **F** card) with the first letters of the atom-label. (E.g. atoms Ca31 and Ca32 would both have scattering-factor Ca; atom P would have factor P, and so would atom P5, but atom Pb4 would expect factor Pb).

If the user wants the *scattering-factor-label* to be something other than the first letters of the atom-label, he gives the *scattering-factor-label* explicitly; this again has up to 4 characters, starting with a letter.

The x,y,z coordinates are fractions of a unit cell. If any of x,y,z or the isotropic temperature factor is absent, it is assumed to be 0.

The site occupation factor may be absent, in which case it is assumed to be 1. If the site occupation factor is wanted but the scattering-factor-label is not, the scattering-factor-label may be omitted.

NOTE:

Note also that site multiplicity is taken care of by the program, and need not be artificially put into the site occupation factor by the user.

EXAMPLES:

```
A Ca2 .1234 .2334 2/3  1.9
```

There is no site occupation factor on the card, and this atom would expect a scattering-factor to be given on an **F** card with label Ca.

```
A  Fe  0    1/2    1/2    0.6   Fe2   0.8
```

Here the atom-label is Fe, its scattering-factor is labelled Fe2, its position coordinates are $(0,\frac{1}{2},\frac{1}{2})$, its isotropic temperature factor is 0.6 and its site occupation factor is 0.8 .

ROUTINES WHICH READ THE CARDS:

Routine ATOPOS calls INPUTA to read each **A** card and interprets the data furnished. (Routine SETFC also calls ATOPOS).

§ § ∞ ∞ § §

B cards
Bond lengths and angles

A **B** card starts with **B** then a *CCSL-word* , which determines what else is on the card. There is no predetermined sequence for **B** cards. They drive the program BONDS.

DATA FOLLOWING ALLOWED CCSL WORDS:

SLAK

Data:

No further data.
The presence of a **B SLAK** card indicates to BONDS that a separate output file, the .SLK file, is required. This file can then be used as part of the Crystal Data input to Least Squares programs which allow geometric slack constraints. The presence of a **B SLAK** card alters the meanings of some of the other **B** cards.

BMAX

Data:

A real number, B_{max} which is the maximum value for a bond in Å.

Assumptions made:

The default for B_{max} is 2.

BMIN

Data:

A real number, B_{min} which is the minimum value for a bond in Å.

Assumptions made:

The default for B_{min} is 0.

Note:

BMAX and **BMIN** may also appear on an **I** card.

BOND

Data:

A list of *atom-labels* or *scattering-factor-labels* followed, if a **B SLAK** card is present, by four real numbers.

The use of scattering-factor names is not because **BONDS** has any connection with scattering-factors, but because it enables the user to specify all of: e.g. C1, C2, C3 $\cdots$ etc by giving simply C. The first atom-label defines a *source-atom*. A grid of 5 × 5 unit cells is searched for *destination-atoms*, symmetry equivalents of the remaining atoms in the list, whose distance from the source-atom is between B_{min} and B_{max}.

If a **B SLAK** card is present, the last four items on the **B BOND** card must be the real numbers b_1 σ_1 b_2 σ_2. The maximum and minimum bonds are taken to be:

$$B_{max} = b_1 + \sigma_1 \qquad B_{min} = b_1 - \sigma_1$$

When a bond is found it is written as an **L BOND** card to the .SLK file, with bond length = b_2 and allowed deviation = σ_2

Assumptions made:

If only one label is given as data, all atoms are used as destination atoms. If there are no labels, all atoms are used as both source and destination.

BOTH

Data:

Are as for **B BOND. A B BOTH** card will cause **both** bonds and interbond angles to be printed. It is not useful in the **B SLAK** context; to produce **L ANGL** cards, **B ANGL** cards should be used.

ANGL

Data:

Three *atom-labels* followed, if **B SLAK** is present, by four or six real numbers.

If there is no **B SLAK** card this simply asks for the calculation of one angle subtended at the second atom, by bonds from it to the other two.

If there is a **B SLAK** card the labels may be *atom-labels* or *scattering-factor-labels* and they must be followed by the four real numbers $\phi_1\ \sigma_1\ \phi_2\ \sigma_2$ and, optionally, B_ϕ and σ_{max}. If B_ϕ and σ_{max} are present:

$$B_{max} = B_\phi + \sigma_{max} \quad \text{and} \quad B_{min} = B_\phi - \sigma_{max}$$

Assumptions made:

If B_ϕ and σ_{max} are absent, values are sought for B_{max} and B_{min} on **B BMAX** and **B BMIN** cards (or **BMAX, BMIN** items on an **I** card). If they are not found they default to:

$$B_{max} = \frac{1}{4}a \quad \text{where } a \text{ is the real cell side} \qquad B_{min} = 0.$$

Note:

The B_{max} and B_{min} are applied to the three labelled atoms as though they came from **B BOND** cards, except that in this case **L ANGL** cards (with angle $= \phi_2$ and $\sigma = \sigma_2$) are written to the .SLK file when the calculated bond angle ϕ is in the range $\phi_1 - \sigma_1 < \phi < \phi_1 + \sigma_1$.

EXAMPLES:

```
B BOND
B BOND Ca3
B BOTH O1 O4 O8 C6 C7 C9
B ANGL Oxy2 Oxy1 Oxy3
B SLAK
Z
Z       This will look at all bonds starting from an atom whose
Z       label starts C
Z       (e.g.  C1 C14 C6A but not Ca3),
Z       and finishing at atoms whose label starts O.
B BOND C O 1.6 0.2 1.61 0.4
Z
Z       This considers only atoms O6, C4, O7 and those related
Z       to them by symmetry.
B ANGL O6 C4 O7 109 0.5 109 0.3
```

ROUTINES WHICH READ THE CARDS:

The main program BONDS reads and interprets **B** cards.

§ § x x § §

C cards
Cell parameters

ITEMS READ:

C a, b, c, α, β, γ

ITEMS DEMANDED:

C, a, and any others of b, c, α, β, γ which are **not** implied by the space group symmetry (which must have already been set up).

ASSUMPTIONS MADE:

As zero is not a plausible value for either a cell side or an angle, any of the 6 values which is read as zero is deduced from the space group symmetry, assuming that it is either fixed or related to one of the values read previously.

DETAILS:

a, b and c are cell sides in real space, normally in Å, but they need not be. However note that the units must be compatible with $\sin\theta/\lambda$ units on **F** cards, λ on **D** cards etc. α, β and γ are the interaxial angles in degrees.

NOTE:

Any change of units from Å may lead to implausible default values e.g. for bond lengths used in slack constraints.

EXAMPLES:

```
C 5.4560 5.4560 12.6700 90.0 90.0 120.0
C    5.4560    0   12.6700
C    5.4560,   0,  12.6700
C 5.4560,,12.6700
```

with hexagonal symmetry, are all equivalent. However,

```
C 5.456 12.67
```

would not be; this says a=5.4560 and b=12.6700

ROUTINES WHICH READ THE CARDS:

Routine RECIP calls INPUTC to read the **C** card and then deduces the real and reciprocal cells and transformations. (Routine SETFC also calls RECIP).

§ § ∞ ∞ § §

D cards
Diffraction information

A **D** card starts with **D** then a *CCSL-word* which determines what else is on the card. There is no predetermined sequence for **D** cards, although it is desirable to type all **D** cards starting with the same *CCSL-word* consecutively.

DATA FOLLOWING ALLOWED CCSL WORDS:

GEOM

Data:

An integer IGEOM defining the diffraction geometry.
Values of IGEOM allowed are:

1	Normal beam with Zone-axis vertical
2	Equatorial 3-circle with Zone-axis vertical
3	Equi-inclination
4	Precession
5	Anti-equi-inclination
6	4-circle bisecting geometry
7	4-circle general geometry, angles to be given
8	Normal beam general geometry
9	Powder data
10	Powder data, X-ray
11	4-circle diffractometer, high-chi geometry
12	SXD geometry

WVLN

Data:

The wavelength(s) in the same units as the lattice constants.

UMAT

Data:

The 9 elements of the UB-Matrix relating the crystallographic axes to the diffractometer axes thus:

	X	Y	Z	Diffractometer Axes
a*	u(1)	u(2)	u(3)	
b*	u(4)	u(5)	u(6)	
c*	u(7)	u(8)	u(9)	

X is parallel to the incident beam at omega=0,
Z is parallel to the omega axis,
Y makes up an orthogonal right handed set.
(Reference Busing and Levy Acta Cryst **22** p247, but note the different definition of the diffractometer axes and the transposition of the matrix).

TH2M

Data:

2θ for the monochromator in degrees (needed for X-ray polarisation corrections).

L/R

Data:

+ve if the incident beam is diffracted to the right.
−ve if the incident beam is diffracted to the left.
value 1 for angles measured clockwise about +ve diffractometer axes, or 2 for anticlockwise.

ROTA

Data:

The zone-axis which is parallel to the rotation axis for IGEOM = 1,2,3,4,5.

CHIA

Data:

Three integers and a real number. For IGEOM = 2 the real number is the angle in degrees between the normal to the Chi circle and the direction of the reciprocal space vector whose indices are the 3 integers.

ANGP

Data:

The precession angle in degrees for IGEOM = 4.

CARDS DEMANDED:

GEOM and either **UMAT** or **ROTA** with possibly **CHIA** or **ANGP**

TELESCOPING OF SIMPLE CARDS:

The **GEOM, TH2M, L/R** and **ANGP** information may occur together.

EXAMPLE:

```
D WVLN    0.9190
D GEOM 8
D L/R 1
D UMAT   0.20457  -0.00747  -0.01437
D UMAT   0.00894  0.20391   0.02125
D UMAT   0.01989  -0.03210   0.29975
```

or

```
D L/R -2  GEOM 11
```

ROUTINES WHICH READ THE CARDS:

Routine SETDC calls INPUTD to read the **D** cards and then sets up the diffraction geometry. SETABS, SETLP and SETPOL all call SETDC.

§ § ∞ ∞ § §

E cards
Extinction Correction Data

ITEMS READ:

E, followed by the type of extinction correction, the domain radius and the mosaic spread.

ASSUMPTIONS MADE:

That there is no extinction unless IEXTYP, the type of extinction correction, is either 1 or 2.

DETAILS:

IEXTYP=1 requests the Becker and Coppens Lorentzian model, and IEXTYP=2 requests the Becker and Coppens Gaussian model. For either model, the domain radius (DOMR) and the mosaic spread (MOSC) are required. DOMR is given in μmetres, and MOSC is given units of 10^{-4} inverse radians.

REFERENCE:

Becker and Coppens (1974) Acta Cryst **A30** p129.

EXAMPLE:

```
E    1   100.0      17.4
```

ROUTINES WHICH READ THE CARDS:

Routine EXTINC calls INPUTE to read the **E** card.

§ § ∞ ∞ § §

F cards
Form or scattering factors

ITEMS READ:

F, a *scattering-factor-label*, an integer FTYP giving the type of factor, followed by more numbers, depending on the type.

FTYP = 0	read 1 number to be a simple multiplicative factor.
FTYP = 1	read 1 number to be a neutron nuclear scattering factor (this is distinct from FTYP=0 because FTYP=0 may occur in addition to another type for the same atom, e.g. FTYP=2).
FTYP = 2	read 5, 7 or 9 numbers to be coefficients in an exponential approximation to the scattering factor.
FTYP = 3	read a table of pairs of numbers, S and f, being a table of values of the factor, f, tabulated versus $S = \sin\theta/\lambda$.
FTYP = 4	as FTYP=2 except that each term of the expansion is multiplied by $\sin\theta/\lambda$. This type is provided for radial integrals $< j_l >$ with $l \neq 0$.
FTYP = 5	Read radial wave functions from **W RADF** cards and use them to calculate the form factor.
FTYP =−1	read 2 numbers to be f' and f'' of an anomalous scattering factor.

DETAILS:

Every **F** card has the general format **F**, name, type, some number of real numbers. If there is not room for all the necessary numbers on the first card, other similar cards may follow. But **all F** cards must start **F** name, type.

For types 0,1 and −1, one card will suffice for one factor. For type 2 the 5, 7 or 9 coefficients in the expression:

$$a_1 exp(-a_2 S^2) + a_3 exp(-a_4 S^2) + etc + a_n$$

are given; it will usually be possible to fit them on to one card. (Reference: International Tables Vol. C Sections 4.4.5 and 6.1.1.4, I.U.Cr 1992).

For type 3 you will almost certainly need more than one card. The values of $S = \sin\theta/\lambda$ must be in ascending order, but need not be at equal intervals.

The label given on an **F** card will usually match one on an **A** card (q.v.). A warning is given if no **A** card matches at all, but this may be intentional.

EXAMPLES:

```
F N14   0   100.        (a multiplying factor in addition to a form factor)
F  Fe   1    0.951      (note that the 0.951 differs from the factor given on a type
                         0 card because it is allowed to be a refinable parameter)
F OXYG  2  3.048 13.277  2.287  5.701  1.546  0.324  0.867 32.909  .251
F  Cr4  3   0    1   .05 .9738  .1  .9  .15 .7917  .2 .6655  .25 .5372
F  Cr4  3   .35 .3156  .4 .2309  .45 .1638  .5 .112  .6 .0442
```

ROUTINES WHICH READ THE CARDS:

Routine SETFOR calls INPUTF to read an **F** card which it then links to the atoms to which it belongs. (Routine SETFC calls SETFOR).

G cards
Data for absorption and extinction corrections

(The **G** stands for "Gaussian integration")

A **G** card starts with **G** then a *CCSL-word* which determines what else is on the card. There is no predetermined sequence for **G** cards. It would be usual, but not essential, to type all **G** cards starting with the same *CCSL-word* consecutively.

DATA FOLLOWING ALLOWED CCSL WORDS:

FACE

Data:

The coefficents a, b, c, and d in the equation for one of the plane faces of the crystal:

$$ax + by + cz \geq d$$

where x, y, and z are CCSL orthogonal axes defined near the start of this chapter and d is positive.

Example:

```
G FACE   0.312  0.534  0    0.86
```

MU

Data:

The absorption or depolarisation coefficient, which should be in the same type of units as d in the **G FACE** cards above (e.g. in inverse mm if d is in mm).

MODE

Data:

the integer NINT indicating which kind of integral is required:

NINT = 1	Transmission factor integral: $exp(-\mu(\tau_{in} + \tau_{out}))$
NINT = 2	Depolarisation integral: $exp(-\mu\tau_{in})$
NINT = 3	Derivative integral for extinction: $(\tau_{in} + \tau_{out})exp(-\mu(\tau_{in} + \tau_{out}))$
NINT = 4	Means do 1 and 2
NINT = 5	Means do 1 and 3
NINT = 6	Means do 2 and 3
NINT = 7	Means do all three

PNTS

Data:

The number of points for integration in the x, y, and z directions respectively. The maximum for each is 10.

CARDS DEMANDED:

As many **FACE** cards as the crystal has faces

ASSUMPTIONS MADE:

For both **MODE** and **MU**, if no **G** card is given, the values of NINT and/or the absorption coefficient are expected to be set up in COMMON /ABSDAT by the calling program. If no **G PNTS** card is given, 5 points are set up as default in each direction.

TELESCOPING OF SIMPLE CARDS:

The **MU** and **MODE** information may occur on the same card.

ROUTINES WHICH READ THE CARDS:

Routine SETGAU calls INPUTG to read and interpret the **G** cards. Routines SETABS and SETPOL call SETGAU.

§ § ∞ ∞ § §

I cards
Instructions to programs

An **I** card starts with **I** followed by a series of *CCSL-word/number pairs*. The main program decides how this information is interpreted.

For programs like ARRNGE, ARRINC the vocabulary is:

DTYP	format for input data (mandatory)
PRIN	control of output format (default=1)
RNUM	control of record number output (default=0)
WGHT	weighting scheme required (default=0)
REJ	whether a list of rejections is to be supplied (default=0)

whose meanings are explained in the relevant program specifications. Routine IICD3 interprets the **I** card according to the meanings above.

For LSQ programs the vocabulary is:

NCYC	number of cycles (default 3)
CYC1	number to be assigned to first cycle (default 1)
PRIN	frequency at which output lists are printed (default 2) PRIN=0 no printing PRIN=1 first cycle PRIN=2 last cycle PRIN=3 first and last cycles PRIN=4 every cycle
MCOR	controls the output of correlations at end of job (default +70) MCOR=0 whole correlation matrix printed MCOR −ve no printing MCOR +ve correlations above MCOR printed
CONV	convergence criterion; LSQ cycling is terminated if the maximum shift/σ < CONV (default 0.01)

Routine IICD1 interprets the **I** card according to the meanings above.

ROUTINES WHICH READ THE CARDS:

Routine INPUTI reads and stores all the information from **I** cards without interpreting it. LOGICAL FUNCTION ONCARD('I',WORD,X) will detect the particular WORD if it has been given on an **I** card, and set X to the value of the number following it.

§ § x x § §

J cards

Data for structure factor calculations using multipole form factors

All **J** cards have the format: **J** *atom-label* *CCSL-word* data

DATA FOLLOWING ALLOWED CCSL WORDS:

MPOL

Data:

CCSL-word/number pairs giving the names of the multipoles and their amplitudes.

Multipole names have the form Y*lms* where:

l and m are digits in the range 0 to 6 with $l \geq m$
s is a sign, being one of $-$ $+$ or "space", and applies to m.

The amplitude is that of the real combination of the spherical harmonics Y_l^m and Y_l^{-m}. Thus the amplitude of Y*lm*+ multiplies the combination

$$\frac{1}{\sqrt{2}} i^m (Y_l^m + Y_l^{-m})$$

Several multipole amplitude pairs for the same atom may be given on one card.

FORM

Data:

Words which indicate which radial form factors apply to which multipoles. The words may be either:

a *scattering-factor-label* "f-label" followed by RADF, in which case the radial wavefunctions for this atom should be given on **W** "f-label" **RADF** cards.

or:

scattering-factor-label /*l-value* pairs, where the form factor on the corresponding **F** card applies to the multipoles with this value of l.

EXAMPLES:

```
J Mn2 MPOL Y20 0.25 Y22+ 0.1
J Mn2 FORM Mn RADF
```

These indicate that the quadropole functions Y_2^0 and $-(Y_2^2 + Y_2^{-2})$ are to be used to model the density around Mn2, and that the associated radial form factors are to be calculated from radial wave functions given on **W** cards.

```
J Mn FORM Mn2j 2 Mn4j 4
```

This indicates that the form factors labelled Mn2j and Mn4j read from **F** cards are to be used for the quadropoles and octopoles respectively.

ROUTINES WHICH READ THE CARDS:

Routines MPFORM and PFSET call INPUTJ to read the **J FORM** and **MPOL** cards

§ § ∞ ∞ § §

L cards
Least Squares Refinement information

L cards drive the various Least Squares Refinement programs of CCSL. For a general introduction to Least Squares the reader is referred to Chapter 5.

Each **L** card has its own format, not particularly connected with any other **L** card.

An **L** card starts with **L** then a *CCSL-word* , which determines what else is on the card. There is no predetermined sequence for **L** cards. It would be usual, but not essential, to type all **L** cards starting with the same *CCSL-word* consecutively.

DATA FOLLOWING ALLOWED CCSL WORDS:

The general *CCSL-words* **MODE, REFI, SCAL, TFAC** and **WGHT** are described first. Then follows the group **FIX, VARY, RELA** and **FUDG** concerned with Least Squares parameters, and finally the group **SLAK, ATOM, BOND, ANGL, EQUA, EQUB, LINE** and **TORS** concerned with geometric slack constraints.

MODE

Data:

An integer, MODER, specifying the mode in which reflection data will be given. This is interpreted by main programs SFLSQ, MAGLSQ etc in the range 0 to 8 as follows:

MODER=0	The user will supply a routine QLSQIN(K,NOMORE), which will read into COMMON /OBSCAL/ the necessary quantities. This is only necessary if the data are in some format unknown to the system.
MODER=1	h, k, l, Gobs, (Scale number) (Code)
MODER=2	h, k, l, Gobs, W, (Scale number) (Code)
MODER=3	h, k, l, Gobs, σ, (Scale number) (Code)
MODER=4	h, k, l, Gcalc, Gobs, σ, (as output by several main programs)
MODER=5	h, k, l, Gobs, σ, C1, C2, C3, C4, (Scale number) to be used when extinction corrections are applied. See Becker and Coppens, Acta Cryst **A30** p129: C1 = $\lambda^3\overline{\tau}/V^2 \sin 2\theta$ C2 = $\lambda/\sin 2\theta$ C3 = $A(\theta)$ C4 = $B(\theta)$
MODER=6	h, k, l, Acalc, Bcalc, Gobs, σ, (Scale number) (Code)
MODER=7	As mode 4 but h, k, l are floating point numbers
MODER=8	As mode 5 but h, k, l are floating point numbers

In these formats for reflection data, h, k, l are in format I5 (for MODEF=1-6) or in F8 (for MODEF=7,8). The numbers Gobs, σ its standard deviation, W the weight to be applied to this observation, and the four C values are all in format F10. The integers Scale number and Code are in I5, and have been given here in parentheses to indicate that they are optional. Gobs is the observed value of the structure amplitude for SFLSQ and MAGLSQ even for IREF=2, but for IREF=5 Gobs is the polarised neutron flipping ratio R.

The Scale number indicates which of several scale factors applies to this observation. If it is read as zero it is set to 1, so if there is only one scaling region this number may be omitted from the data. The Code is not at present interpreted, but the user may wish to use it in his own main program.

Assumptions made:

if no **L MODE** card is given, MODER=3.

REFI

Data:

An integer, IREF, indicating the type of refinement, that is, the calculated function which is to be compared with the observed data. IREF is consulted by SFLSQ, MAGLSQ etc. Currently recognised values are 1, 2, 4 and 5:

IREF=1 refine on the modulus of the structure factor Fc
IREF=2 refine on the square of Fc
IREF=4 refine on signed Fc (centrosymmetric)
IREF=5 refine on polarised neutron flipping ratio R (magnetic only)

Assumptions made:

If no **L REFI** card is given, IREF=1 is assumed.

SCAL

Data:

A scale factor for as many scale regions as are required. These will be indexed 1,2,3 etc as they are read. The scale factor is used to multiply the calculated function. If the numbers will not fit on to the 80-character line, further **L SCAL** cards may be given.

Assumptions made:

If no **L SCAL** cards are given, the program assumes that there is one factor, with value 1.0, to be refined.

TFAC

Data:

Overall temperature factor B for structure factors, used as $\exp(-B(\sin\theta/\lambda)^2)$.

Assumptions made:

If no **L TFAC** card is given, no overall temperature factor is used.

WGHT

Data:

An integer IWGHT, the number of the weighting scheme required. At present IWGHT must be 1 or 2:

IWGHT=1 use unit weights (no weighting)

IWGHT=2 use the weight as read with the reflections; for MODER=2 the weight is W, and for other modes where σ is read, the weight is $1/(\sigma^2)$

Assumptions made:

vary with the main program, as appropriate.

WORDS CONCERNED WITH LSQ PARAMETERS:

Please refer also to Chapter 5 for the terminology for naming parameters.

FIX

Data:

sets of *parameter specifications.*

In main programs for single crystal work like SFLSQ, the following names are set up:

Species in family 1, genus 1 : `TFAC DOMR MOSC A* B* C* D* E* F*` (`A*-F*` are for geometric slack constraints)

Genus name for family 1, genus 2: `SCAL`

Species names for family 2, (the structure parameters whose genus name is their atom name):

`X Y Z B11 B12 B13 B22 B23 B33 ITF SCAT SITE`, and also, for magnetic structures, `MU MU1 THET THE1 PHI PHI1 PSI1 PSI2 PSI3 PSI4`

A *parameter specification* is one of the following:

genus name, space(s), species name e.g. `Ca6 X`
`SCAL 4`

species name alone for family 1, genus 1 e.g. `TFAC`

the *CCSL-word* **ONLY** this means that all defaults are overridden, and only the parameters which are explicitly listed are to be fixed.

the *CCSL-word* **ALL** followed by some family, genus or species name, as built in to the particular main program reading the **L** cards.

e.g. `L FIX ALL SCAL ALL ITF ALL Nb2`

the *CCSL-word* **ALL** followed by another *CCSL-word* ; for SFLSQ etc the *CCSL-words* are:

XYZ (= `X` and `Y` and `Z`)
BIJ (= `B11, B12, B13, B22, B23` and `B33`)
XYZT (= `X` and `Y` and `Z` and `ITF`)
XYZS (= `X` and `Y` and `Z` and `SITE`)
XYZB (= `XYZ` and `BIJ`)
CELL (= `A*, B*, C*, D*, E*` and `F*`)
e.g. `L FIX ALL BIJ`

The parameters specified are *fixed* in the subsequent refinement.

Assumptions made:

If no **L FIX** cards are given, the main program's own defaults are taken.

VARY

Data:

an **L VARY** card has exactly the same specification as an **L FIX** card, except that it asks to *vary* the specified parameters rather than to fix them. If the *CCSL-word* **ONLY** occurs on any **L VARY** card, only the parameters which are explicitly listed are varied.

RELA

Data:

Each card contains one relation between parameters (a *constraint*) which the user wishes to impose in addition to those which the system will impose automatically.

The integer after **RELA** is LRELA, the type of relation; at present types 1 and 2 are available, as described in Chapter 5. The data following LRELA are:

LRELA=1 a_1, p_1, a_2, p_2, where the a's are real numbers and the p's are *parameter specifications*.
Constraint: $a_1\Delta p_1 = a_2\Delta p_2$

LRELA=2 as many of a_1 p_1, a_2 p_2, a_3 p_3 . . as necessary, all on one card.
Constraint: $a_1\Delta p_1 + a_2\Delta p2 + \cdots + a_n\Delta p_n = 0$
(Note that the a_2 of LRELA=2 is minus that of LRELA=1)

Note:

The *CCSL-words* like **ONLY, ALL** etc. cannot be used here.

Example:

```
L RELA 1    1 Co SITE  2  Mn SITE
L RELA 2    1.4 SCAL 1  2.8 SCAL 3  -1.9 SCAL 5
```

Assumptions made:

If there are no **L RELA** cards, only the constraints which are inherent in the symmetry of the problem are applied.

FUDG

Data:

Sets of (*parameter specification*, Factor), where the *parameter specification* is as on **FIX** and **VARY** cards except that the *CCSL-word* **ONLY** is not used here. The Factor is either a real number or a *range indicator*. If it is a real number it is a simple multiplicative factor for the shift on the specified parameter.

A range indicator has the form **GE** *limit* or **LE** *limit* where **GE** means "greater than or equal to", **LE** means "less than or equal to" and *limit* is a real number giving the limiting value. If the information will not fit on to an 80-character line several **L FUDG** cards may be given.

Note:

The calculated shifts for the specified parameters are adjusted by the fudge factors before they are applied. In the case of range indicators if, after applying the shift, any of the specified parameters are outside the range, it is set equal to *limit*. The specified parameters may be individuals or groups.

Example:

```
L FUDG    ALL SITE 0.6      Na4 ITF 0.8
L FUDG    Mn SITE LE 1.0
```

Assumptions made:

If no **L FUDG** cards are read, no fudge factors are applied.

WORDS CONCERNED WITH SLACK CONSTRAINTS:

The remaining CCSL-words control the imposition of geometric slack constraints.

SLAK

Data:

The presence of an **L SLAK** card asks for geometric slack constraints to be used. The card gives an integer, STYP and a real number, SWGHT.

The units digit of STYP=1 means "use only slack constraints, and none of the conventional LSQ observations."

The units digit of STYP=2 means "use both LSQ observations and slack constraints."

The tens digit of STYP gives the weighting scheme required for slack constraints; 1=unit weights, 2=use weight as read, and 3=read σ, use $1/\sigma^2$.

Each slack constraint is weighted by its own individual weight, usually read from the card which specifies the constraint. If the units digit of STYP=2 it will be weighted, in addition, by multiplication by SWGHT.

ATOM

Data:

A new atom name and an *Atom Specification.*

An **L ATOM** card is used to assign a name to an atomic position which is equivalent to, but not identical with, one on an **A** card. Cards like **L BOND, L ANGL** etc can then refer to this new atom by name. The Atom Specification can be given in one of two formats. In either case it starts with the atom name.

In the first format the new atomic position is specified by 5 integers, s, l, c_x, c_y, c_z. s is the serial number of the symmetry operator producing the required position from that on the **A** card; it is given negatively if the centre of symmetry is also involved, and can be found by running a CCSL job which calls SYMOP and OPSYM(1). l is similarly the number of the lattice translation used. c_x, c_y, c_z are cell translations in the x, y and z directions enabling the position generated by s and l to be put into any of the neighbouring unit cells; their values would usually be 0, +1 or −1.

In the other format the atom name is simply followed by the actual x y and z fractional coordinates to be used, and the CCSL works out the values of s, l etc.

The two types of format are distinguished by the presence or absence of a decimal point in the x coordinate.

Example:

```
L ATOM Na1A  Na1 2 0 0 0 1
L ATOM Mn99  Mn3   .1234 .2345 .5
```

BOND

Data:

A *bond name* followed by two atom names, and, optionally, a bond length b_{opt} and its allowed deviation σ. The atom names must occur on either an **A** card or a **L ATOM** card. If no bond length is given, this card simply defines the bond name for subsequent use by **L ANGL, L EQUB** etc cards.

If b_{opt} and σ are given, this is a request to apply a *slack constraint* to the calculated length of the bond b_{cal} of the form:

$$b_{cal} = b_{opt} \pm \sigma$$

Example:

```
L BOND   B04   C1  C2
L BOND BMn9  Mn99  Mn9  2.3   0.05
```

ANGL

Data:

An *angle name* followed by the bond names of two bonds to a common atom and, optionally, an angle ϕ_{opt} in degrees and its σ. The bond names must occur on **L BOND** cards. If ϕ_{opt} is not given, this card simply defines the angle name for subsequent use by **L EQUA, L TORS** cards.

If ϕ_{opt} and σ are given, this is a request to apply a *slack constraint* to the calculated bond angle of the form:

$$\phi_{cal} = \phi_{opt} \pm \sigma$$

Example:

```
L ANGL PHI6   Bnd1  Bnd2  109.5  0.3
```

EQUA

Data:

Two angle names and a σ. The angle names must occur on **L ANGL** cards. This card defines a *slack constraint* on the values ϕ_1, ϕ_2 of the two angles of the form:

$$\phi_1 = \phi_2 \pm \sigma$$

Example:

```
L EQUA   Phi4  Phi5  0.4
```

EQUB

Data:

Two bond names and a σ. The bond names must occur on **L BOND** cards. This card defines a *slack constraint* on the values b_1, b_2 of the two bonds of the form:

$$b_1 = b_2 \pm \sigma$$

Example:

```
L EQUB  Bon1  Bon2 0.001
```

LINE

Data:

Two bond names and a σ. The bond names must occur on **L BOND** cards and must have one atom in common. If b_1 b_2 are the lengths of the two bonds and b_3 the length of the third bond in the triangle, this card defines a ***slack constraint*** of the form:

$$b_1 + b_2 = b_3 \pm \sigma$$

Example:

```
L LINE  Bon1  Bon2  0.001
```

TORS

Data:

A ***torsion angle name***, 3 bond names, a torsion angle in degrees θ_{opt} and its allowed deviation σ. The bond names must occur on **L BOND** cards. The torsion angle is defined to be the angle between the plane of bonds 1 and 2 and the plane of bonds 2 and 3. This card defines a ***slack constraint*** on the calculated torsion angle θ_{cal} of the form:

$$\theta_{cal} = \theta_{opt} \pm \sigma$$

Example:

```
L TORS  B1   B47  B23  99.9  .1
```

TELESCOPING OF SIMPLE CARDS:

The **REFI**, **MODE** and **WGHT** information may all occur on the same **L** card.

Example:

```
L WGHT 2  MODE 1  REFI 4
```

ROUTINES WHICH READ THE CARDS:

Routines whose names start SFLS in general set up LSQ programs, which includes the reading of **L** cards. Routines with names starting INPL read a subset of **L** cards, omitting the **FIX**, **VARY**, **RELA** and **FUDG** cards and all the cards for geometric constraints.

RDFV	reads **L FIX** and **L VARY** cards.
RDRELA	reads **L RELA** cards.
FUDGIN	reads **L FUDG** cards.
GEOMIN	reads **L SLAK**, **L ATOM**, **L BOND**, **L ANGL**, **L EQUA**, **L EQUB**, **L LINE** and **L TORS** cards.

Called from within routine INPLSF there are:

LLTFAC	to read **L TFAC** cards.
LLSCAL	to read **L SCAL** cards.

NOTE:

See also **I** cards in their application to driving LSQ programs.

Example **I** card for LSQ:

```
I  NCYC 3   CYC1 10   MCOR 50   CONV 0.05
```

§ § ∞ ∞ § §

M cards
Map-drawing information

An M card starts with M then a *CCSL-word*, which determines what else is on the card. There is no predetermined sequence for M cards. It would be usual, but not essential, to type all M cards starting with the same *CCSL-word* consecutively.

DATA FOLLOWING ALLOWED CCSL WORDS:

AXES

Data:

9 elements of an orientation matrix U to rotate the Fourier map during its calculation.

Assumptions made:

If no M **AXES** card is present, the unit matrix is used. If the numbers read include decimal points, it will be assumed that the general plane Fourier routine is wanted.

Note:

There is a main program GPCARD to generate M AXES for sections passing through three designated atoms.

Details:

The 3 sets of 3 numbers may also be viewed as the zone axis symbols of the x, y and z axes of the computed map.

Example:

```
M AXES 0,0,1  1,0,0  0,1,0
```

CM/A

Data:

Number of centimetres per Angstrom if graphical output on a plotter (not a screen, e.g. Tektronix) is required.

Assumptions made:

If no M **CM/A** card is read, the scale=2.5

CONT

Data:

Real numbers on one or more cards which give the contour levels to be plotted. The *CCSL-word* **STEP** allows a range of equally spaced contour levels to be specified; it is followed by 3 numbers, which are the first value, the last value and the step.

Details:

The values will be sorted into ascending order. Each contour value is sought and plotted in turn.

Example:

```
M CONT  10,25,40, STEP 50,210,20 235
M CONT  0.5  0.8  1.4
```

DELT

Data:

Δ, the resolution length for Fourier calculation.

Assumptions made:

No **M DELT** card implies no resolution function.

Details:

The density is averaged over a cube of side 2Δ.

DTYP

Data:

(Mandatory if Fourier) an integer (MODED) specifying the type of data input.

MODED=0	the user will provide his own data input routine QFOUIN (K,F,ALPHA) to read h, k, l into array K (dimension 3), the modulus of the Fourier coefficient to F and its phase in radians to ALPHA. QFOUIN may use the free format input routines.
MODED=1	h, k, l, F(calc), F(obs)
MODED=2	h, k, l, |(F(calc)|, phase, F(obs)
MODED=3	h, k, l, A(calc), B(calc), F(obs)
MODED=4	h, k, l, any F

For modes 1-4 the reflection data are read in fixed format 3I5, several F10.

FTYP

Data:

(Mandatory if Fourier) an integer MODEF specifying the type of Fourier to be calculated.

MODEF=1	coefficients are F(calc), complex if non-centrosymmetric.
MODEF=2	coefficients are F(obs) and are given with their signs; this can only be used with a centrosymmetric structure and MODED=1 or 4.
MODEF=3	coefficients are |(F(obs)| with the phase (or sign) of the F(calc); this cannot be used with MODED=4.
MODEF=4	coefficients are F(obs)–F(calc), both as read, and so can only be used with a centrosymmetric structure and MODED=1.
MODEF=5	coefficients are (|F(obs)|-|F(calc)|)×phase (or sign) of F(calc); this cannot be used with MODED=4.
MODEF=6	coefficients are $F(obs)^2$ (Patterson function). If used with MODED=4 this will use whatever numbers are typed as coefficients, so these could be F(calc) if the user wishes.
MODEF=7	calculate the standard deviation of the electron density (using the routine ERRMAP). All MODED values are allowed, but the coefficient used in the calculation will be the number given **after** all those listed above under MODED (e.g. MODED=1 means h, k, l, F(calc), F(obs), ΔF where ΔF is the coefficient for ERRMAP.)

GET

Data:

Some number of z values of maps.
M GET cards have the same syntax as **M PLOT** and **M PRIN** cards (below). In this case the z values refer to maps which have already been computed in a previous run, and saved on a file using an **M SAVE** card.

Note:

The name of the saved file must be provided in answer to a request from a Fourier program (e.g. FOURPL).

Saving a map means that if the same map is wanted again it need not be recomputed; the most obvious application for this is the contour plotting of the map using a different scale, or new contours, etc.

There are a limited number of things which may be changed if a map is retrieved with **GET** after having been **SAVE**d. It would not make sense to change the symmetry, or the **MESH**, the **AXES**, the **FTYP** etc. The only cards it makes sense to change are at present **M CM/A** and **M CONT**

When routine PREFIN meets an **M GET** card, it calls routine MAJUST which replaces almost all the current Crystal Data by the previously dumped Crystal Data which matches the saved maps. It accepts new **M PLOT**, **M PRIN**, **M GET** and **M SAVE** cards and ignores any old ones; and it replaces any old **M CM/A** or **M CONT** cards by new ones if given. It would also accept a new **N** card.

Example:

```
M GET   0.4  STEP 0.9 0.93 0.01
```
used with:
```
M CM/A  2.54
M PLOT  STEP 0 0.4 0.1  STEP 0.9 0.93 0.01
```
would first undump the Crystal Data from the named file, and add to it the above 3 cards (losing its own **M CM/A** card). It would compute the maps for z=0, 0.1,0.2 and 0.3 (because they are not dumped), then use dumped maps for 0.4, 0.9, 0.91,0.92,0.93; it would contour plot all of them, at a scale of 2.54 cm/Å, using the contour values it found in the undumped Crystal Data.

MESH

Data:

(Mandatory if Fourier): 6 numbers defining the points at which the map will be calculated. They are the initial value, the final value and the step in fractional coördinates, first for x then for y. In this context x, y, and z are the axes defined by the **M AXES** card.

Note:

SETFOU checks that the step is strictly positive, and that the final value is greater than the initial value.

There are also checks on the allowed storage space in FOUR1Z; if

n_x = no. of x points
n_y = no. of y points
$n_h = 2(h_{max} + 1)$ and
$n_k = k_{max} + 1$

then none of: $n_x n_y$, $n_h n_k$, or $n_x n_k$ must exceed a maximum which has been set in the program, and is at present 10201 (= 101 × 101)

Using the variable array dimensions facility of CCSL.FOR, it is possible to alter this number and recompile the Library.

Example:

```
M MESH   0  1  0.2,   0.5  1  0.2
```

NDIM

Data:

Number of dimensions for Fourier, 2, 3 or 4

NDIM=2	produces a projection, using 2-D data.
NDIM=3	produces sections of a 3-D Fourier map.
NDIM=4	produces "bounded" sections i.e. the density between two given z values projected on the xy plane. (This is useful when data of limited resolution in one direction only are available.)

Assumptions made:

If no **M NDIM** card is present, NDIM=3 is assumed.

PLOT

Data:

Some number of map z values at which contoured plotted maps are required. If NDIM=2, no numbers are given; for NDIM=3 a set of map z values is given, extending to more than one **M PLOT** card if necessary. For NDIM=4 alternate positive and negative z values are given to indicate the limits of the bounded sections.

Assumptions made:

If no **M PLOT** card is given, no plotting is wanted.

Details:

The z values are given as a list, and the use of **STEP** (see **M CONT**) is allowed. They are sorted into ascending order.

Example:

```
M PLOT 0.5  0.8  STEP 0.9 0.95 0.01
```

PRIN

Data:

These cards are exactly similar to **M PLOT**, but cause the values of the Fourier map to be printed, rather than to be plotted. Printing is via routine PRNTMP, which writes 21 integers each of width 5 to a printer line, but could easily be changed to suit individual requirements.

Note:

The map z values involved need not be the same as, say, on an **M PLOT** card. The use of **STEP** (see **M CONT**) is allowed.

Example:

```
M PRIN 0  0.1  0.3564
```

READ

Data:

Numbers with the same syntax as those on **M PLOT**, **M PRIN** and **M GET** cards giving map z values. The relevant maps are assumed to have been previously dumped on to a file, whose name will be requested interactively (if a Fourier program like FOURPL is being used.)

Unlike **M GET** maps, no more information is expected on the dumped file; the user must match his current Crystal Data to the dumped maps. These maps are then used as though they had just been computed.

The routine READMP reads back such a map into array DENS in COMMON /MAPDA/. The reading is unformatted, a line at a time. Essentially, the map must be read back in the same chunks as that in which it was written, so if the existing reading is not adequate, READMP should be adjusted.

Example:

```
M READ 0.1234
```

SAVE

Data:

Numbers with the same syntax as those on **M PLOT**, **M PRIN** and **M READ** cards, giving map z values. The relevant maps will be dumped on to a file whose name will be requested interactively (if a Fourier program like FOURPL is being used), together with enough information to retrieve them (using an **M GET** card) and then use them as though they had just been computed.

M SAVE may be used in the same run as **M PLOT**, **M PRIN** etc., and its z values may occur on those cards also, or they may be peculiar to the **M SAVE** card(s).

M SAVE and **M GET** may also both occur in the same run, as they write to and read from different units.

Example:

```
M SAVE STEP 0.025  0.325  0.025
```

SCAL

Data:

A real number which is the Scale factor by which the Fourier coefficients will be multiplied.

Assumptions made:

If no **M SCAL** card is given, the scale factor is unity

SMAX

Data:

(Mandatory for Fourier maps): Maximum value of $\sin\theta/\lambda$ for data to be included. Input data for which $\sin\theta/\lambda$ exceeds this value are not used in the Fourier calculation.

TELESCOPING OF SIMPLE CARDS:

CM/A, DELT, DTYP, FTYP, NDIM, SCAL and **SMAX** may be put together on the same **M** card.

ROUTINES WHICH READ THE CARDS:

INPUTM reads and interprets all the **M** cards given. SETFOU then deals with default values, and sets the system up to perform Fourier calculations.

EXAMPLE:

```
M DTYP 1   FTYP 1   DELT 0.25   CM/A 1.5   SCAL 1000
M SMAX 0.75
M AXES 0 0 1,  1 1 0, -1 1 0
M MESH -.5  .5  .02   0  1  .01
M PLOT 0
M CONT STEP -9 17 2
```

§ § ∞ ∞ § §

N cards
Title for job

ITEMS READ:

N, any number (up to 79) of characters to form a title.

ASSUMPTIONS MADE:

If no **N** card is given, and a title is called for (e.g. over a plotted contoured map) the word "UNTITLED" will be used.

DETAILS:

The first time a call of INPUTN is made, the title is read and stored from an **N** card (or "UNTITLED" is held). INPUTN also has an input parameter giving the number of an output unit, and the title is output to that unit. Subsequent calls of INPUTN would merely output a copy of the title.

When a title is plotted above a map, it is taken as A1 characters from its COMMON /TITLE/.

ROUTINES WHICH READ THE CARDS:

INPUTN

EXAMPLE:

```
N Mn2CoSn at room temperature - trial 1
```

§ § x x § §

P cards
Polarisation Data

ITEMS READ:

P, followed by:

the polarisation,
the standard deviation of the polarisation,
the flipper efficiency, and
the standard deviation of flipper efficiency.

DETAILS:

When using a cryoflipper the first number is the "up" polarisation and the third is the ratio of the "down"/"up" polarisations.

EXAMPLE:

```
P  0.986  0.005  1  0.005
```

ROUTINES WHICH READ THE CARDS:

SETPOL

§ § x x § §

Q cards
Data for magnetic structure factor calculations

Q cards have the format: **Q** *CCSL-word* data
or
Q *label* data
where *label* is an *atom-label* or a *scattering-factor-label* .

DATA FOLLOWING ALLOWED CCSL WORDS:

In the first case the *CCSL-word* may be one of **STYP PROP MSYM** or **NSYM**, and in the second one of **FORM MU SDIR** or **PSI**.

STYP

Data:

A *CCSL-word* describing the magnetic structure type. The types defined are:

ANTI	ordinary comensurate antiferromagnetic structure
AMOD	incommensurate amplitude modulated antiferromagnetic structure
FERO	unmagnetised ferromagnet with equal population of all domains
FERA	aligned ferromagnet with spins in the direction given by the **SDIR** card.
HELI	helical spin structure

PROP

Data:

The three components of the propagation vector in reciprocal space coordinates.

MSYM

Data:

Pairs of integers defining the magnetic space-group. The first number of a pair is the identifying number of one of the generating elements of the group, and the second is -1 if the symmetry operation is combined with time-reversal, and 1 if not. The special operator number -1 is used to refer to a centre of symmetry.

There must be one such pair of integers for each of the generating elements of the magnetic group.

Note:

In CCSL the space group symmetry operators are assigned identifying numbers which can be printed out using OPSYM. It is these numbers which are used to refer to individual operators when defining their magnetic symmetry.

NSYM

Data:

Operators for symmetry elements not in the magnetic space group.

The magnetic space group **M** may or may not coincide with the nuclear group **N** but **M** must be a subgroup of **N**.

In the case that $\mathbf{M} \neq \mathbf{N}$ then some **NSYM** cards are needed to describe how each symmetry element which is **not** in the magnetic group acts on the directions of the magnetic moments. There is one **NSYM** card for each element of the factor group **F** where $\mathbf{M} \times \mathbf{F} = \mathbf{N}$.

The data given on the **NSYM** cards are the integer label which has been assigned to the operator, followed by nine real numbers. These specify the matrix describing the rotation with respect to the representative atom, of the moment on the atom generated from the representative atom, by the operator. The matrix is expressed in the CCSL orthogonal axes.

FORM

Data:

A *scattering-factor-label* appearing on an **F** card.

Not all atoms appearing on **A** cards are necessarily magnetic. They are defined to be magnetic if their names appear to the right of **FORM** on a **Q** card.

The form factor will be used as the *magnetic* form factor applying to all the atoms whose *atom-labels* appear to the right of **FORM** on the card.

MU

Data:

An *atom-label* corresponding to one on an **A** card and, unless the structure type is HELI, a single number giving the magnetic moment in Bohr magnetons.

The *CCSL-words* **MU** and **SDIR** refer to individual magnetic atoms. There must be one of each of these cards for each of the **A** cards which refer to magnetic atoms.

If the structure type is HELI two numbers are required which are the major and minor axes of the elliptical envelope of the helix in Bohr magnetons.

Note:

The moment values are referred to as `MU` and `MU1` in the Least Squares programs.

SDIR

Data:

An *atom-label* corresponding to one on an **A** card followed by the spherical polar angles θ and ϕ of the moment direction of the atom with respect to CCSL orthogonal axes.

If the structure type is HELI a second pair of angles is required; the first pair give the orientation of the major axis of the elliptical envelope and the second pair the direction of the minor axis (the two directions must be perpendicular).

Details:

The angles are given in degrees.

The vector describing a moment direction or one of the axes of the elliptical envelope thus has components:

$\mu \sin\theta \cos\phi$, $\mu \sin\theta \sin\phi$ and $\mu \cos\theta$ on the orthogonal CCSL axes

Note:

These angles are referred to by the names `THET`, `PHI`, `THE1`, `PHI1` respectively in the Least Squares Programs.

PSI

Data:

An *atom-label* corresponding to one on an **A** card followed by up to four pairs of numbers. The first number of each pair is the integer label of an operator and the second is the phase shift in degrees to be applied to the sublattice generated by that operator.

Note:

One or more **PSI** cards will be required for each magnetic atom when the structure type is AMOD or HELI and the magnetic symmetry is less than the nuclear symmetry, i.e. there are one or more **NSYM** cards.

Phase shifts must be defined for operators not in the magnetic group which relate different atomic positions of an equivalent set. At present there is space for only four phase shifts per atom which means that the programs can not deal with structures in which magnetic atoms occur on sites whose multiplicity due to symmetry operators not in the magnetic group is greater than four.

EXAMPLES:

The following is an example of part of the Crystal Data describing the magnetic structure of Mn_3Sn:

```
A Mn      0.8415    0.68291    0.25000    0.00000
S x-y, x, 1/2+z
S x, y, 1/2-z
S y, x, 1/2+z
F Mn       1   -0.37300
F MnM  2  0.4220 17.6840 0.5948 6.0050 0.0043 -0.6090 -0.0219
Q PROP 0 0 0
Q STYP ANTI
Q MnM FORM Mn
Q MSYM -1 1 4 -1 8 -1
Q NSYM 2 -.5 .866 0 -.866 -.5 0 0 0 1
Q NSYM 3 -.5 -.866 0 .866 -.5 0 0 0 1
Q Mn MU      3.0
Q Mn SDIR    90.0   60.0
```

The first two **Q** cards indicate an antiferromagnetic structure with zero propagation vector (nuclear and magnetic cells are the same). The **S** cards define space group $P6_3/mmc$ and the output from OPSYM(1) is:

```
          General equivalent positions are:
        0                  0                  0        +-
 1      x                  y                  z
 2      x-y                x                 1/2+z
 3     -y                  x-y                z
 4     -x                 -y                 1/2+z
 5     -x+y               -x                  z
 6      y                 -x+y               1/2+z
 7      y                  x                 1/2+z
 8     -x+y                y                  z
 9      x                  x-y                z
10     -x                 -x+y               1/2+z
11      x-y               -y                 1/2+z
12     -y                 -x                  z
```

This information is needed to understand the **Q MSYM** and **Q NSYM** cards. The interpretation of the card `Q MSYM -1 1 4 -1 8 -1` is as follows:

Operator -1: (the centre of symmetry) is not combined with time reversal so that the spins on atoms related by the centre of symmetry are parallel.

Operator 4: $-x, -y, 1/2+z$ (a screw diad parallel to z) is combined with time-reversal and atoms related by this operator have parallel x and y components and anti-parallel z components.

Operator 8: $-x+y, y, z$ (a mirror plane bisecting the angle between x and y) is also time-reversing and, since spins are axial vectors, the components parallel to the plane, of spins which it relates, are parallel to one-another whereas those perpendicular to the plane are anti-parallel.

The **NSYM** cards

```
Q NSYM 2 -.5 .866 0 -.866 -.5 0 0 0 1
Q NSYM 3 -.5 -.866 0 .866 -.5 0 0 0 1
```

indicate that the operators 2 and 3 which describe the screw hexad (6_3) and triad operations respectively are not in the magnetic space group. The matrix on the **NSYM** 2 card implies that atoms related by the screw hexad have spin directions rotated with respect to one-another by 60° about z but with the rotation direction opposite to that of the symmetry axis. The matrix acompanying element 3 implies that the spin rotation associated with the triad axis is also opposite to that of its symmetry operation.

The **MU**, **SDIR** and **FORM** cards indicate that the Mn atom is magnetic with a spin of 3.0 μ_B. The representative Mn atom (that whose position is given on the **A** card) has its spin direction in the $x-y$ plane ($\theta = 90°$) at 60° to orthogonal x ($10\bar{1}0$). The magnetic form factor for Mn is MnM.

A second example desribes the magnetic structure of the helical phase of monoclinic CuO:

```
S 1/2+x, 1/2+y, z
S 1/2+x, 1/2-y, 1/2+z
S -x, -y, -z
A Cu      0.25000    0.25000    0.00000    0.00000
Q CuM FORM Cu
Q PROP .507 0 -.482
Q STYP HELI
Q Cu MU      0.5013     0.4860
Q Cu SDIR     90.0000    90.0000    28.2215     0.0000
Q NSYM 2 1 0 0 0 1 0 0 0 1
Q NSYM -1 1 0 0 0 1 0 0 0 1
Q Cu PSI      2    4.5
```

Note the irrational values on the **Q PROP** card for this incommensurate structure. In this example neither the twofold axis (operator 2) nor the centre of symmetry is in the magnetic group, but in both cases the atoms related by these operators have parallel spins. The major and minor axes of the elliptical envelope of the spin helix are 0.5013 and 0.4860 μ_B, oriented parallel to **b** and at 28.2° to **c** in the (010) plane, respectively. The phase of the spiral based on the Cu atom at $(\frac{3}{4}\frac{3}{4}\frac{1}{2})$ leads that on the representative Cu atom $(\frac{1}{4}\frac{1}{4}0)$ by 4.5°.

§ § x x § §

S cards
Space group symmetry

ITEMS READ:

S followed by 3 items giving general equivalent positions which make crystallographic sense, producing a 3x3 rotation matrix and a 1x3 translation vector.

ASSUMPTIONS MADE:

If no **S** cards are present at all, the space group is assumed to be P1. The x,y,z operator is always assumed to be present.

DETAILS:

For each item one or two letters (x,y,z,X,Y or Z) will be present.There may also be a fraction, typed as n/m where n may be 1, 2, 3 or 5 and m may be 2, 3, 4 or 6. Letters and fractions may be preceded by a sign, + or −, and an initial + is optional.

There need only be enough **S** cards to give the generators of the group, but redundant cards are not faulted. When SYMOP reads the **S** cards and has formed the whole group, it will identify and print out a set of generators.

SYMOP also checks that the rotation matrix has no zero row or column. The new operator is added to an existing list, and the resulting set of operators is augmented (if necessary) to form a group. If more than 48 (primitive) operators are found in this way an error is reported.

NOTE:

It is now also possible to specify the space group on an **S GRUP** card, using either its short symbol or an integer in the range 1-230, all of which refer to the listings in the International Tables for X-Ray Crstallography, Vol I, 1952. Spaces must be inserted in the symbol in the obvious places, e.g. P21/c, Ab m 2, P-3 1 m. The user is advised to check that such **S** cards have produced the operators he expects.

WARNING:

The maximum permitted number of operators stored is usually 24 not 48. A centre of symmetry found other than at the origin will cause the structure to be treated as non-centrosymmetric. If this means that there are finally 48 operators a specially compiled version of CCSL with space for these must be used.

EXAMPLES:

```
S -x, -y, -z
S 1/2+X,   1/2+Y,   1/2+Z
S   y,   y-x,   z + 5/6
S GRUP P21 21 2
S GRUP 42
```

ROUTINES WHICH READ THE CARDS:

INPUTS reads one **S** card, and SYMOP calls INPUTS repeatedly to read all the **S** cards and form the group. SETFC also calls SYMOP.

The complete set of operators generated by SYMOP may be printed out by calling OPSYM(IS) where IS is 1 for output of equivalent positions and 2 for output of equivalent reflections with their relative phases.

§ § ∞ ∞ § §

T cards
Anisotropic temperature factors

ITEMS READ:

T , an *atom-label*, an integer TTYP indicating the the units in which the constants are given, and the 6 coefficients, B_{11}, B_{22}, B_{33}, B_{23}, B_{13} and B_{12}, **in that order.**

ASSUMPTIONS MADE:

If there is no **T** card for a particular atomic position, there is no anisotropic temperature factor for that atom.

DETAILS:

The atom label must match one on an **A** card. TTYP is an integer with the following interpretation:

TTYP=0 make an anisotropic temperature factor from the existing isotropic factor. The new factor will be type 2.

TTYP=1 is not used

TTYP=2 uses the expression:
$\exp(-\frac{1}{4}(B_{11}h^2a^{*2} + .. + 2B_{23}kb^*lc^* + ..etc))$ (note the "2")

TTYP=3 uses the expression:
$\exp(-2\pi^2(B_{11}h^2a^{*2} + .. + 2B_{23}kb^*lc^* + ..etc))$ (note the "2")

TTYP=4 uses the expression:
$\exp(-(B_{11}h^2 + .. + B_{23}kl + ..etc))$

TTYP=5 uses the expression:
$\exp(-(B_{11}h^2 + .. + 2B_{23}kl + ..etc))$ (note the "2")
which is also the form used internally in the program.

EXAMPLES:

```
T CA2  0
T Cu11 2  1.43  1.58  1.89  0.65  0.50  0.37
```

ROUTINES WHICH READ THE CARDS:

INPUTT reads one **T** card. SETANI reads all the **T** cards which the user gives. SETFC also calls SETANI.

§ § x x § §

U cards
Asymmetric unit definition

NOTE: It is not usually necessary to give a **U** card.

ITEMS READ:

U and 3 indices h, k, l

ASSUMPTIONS MADE:

If no **U** card is provided, the typical reflection indices are taken to be 13,11,10. These have the property of being all positive, with $h > k > l$.

DETAILS:

The indices h, k, l are used by routine SYMUNI to select which of the possible asymmetric units it will choose. SYMUNI finds one unit, then transforms it using the space group symmetry until it contains the given h, k, l.

This means that h, k, l must be properly inside a unit, not on a face or edge. Routine TRYUNI will give an error if this is not so.

Note that, although in general the provision of a **U** card will force the asymmetric unit which you intended, in some cases, notably for tetragonal groups, there are (at least) two ways of chopping reciprocal space into asymmetric units, and the user will not be able to impose one if the program has chosen the other.

EXAMPLE:

```
U  -1,4,5
```

ROUTINES WHICH READ THE CARDS:

INPUTU reads one **U** card, and is called by SYMUNI

§ § ∞ ∞ § §

V cards
Space group representations

Details of V cards are not yet finalised.

§ § ∞ ∞ § §

W cards
Information about atomic wave-functions

A **W** card has the form **W**, *label* , *CCSL-word* , data.

The *label* is either an *atom-label* appearing on an **A** card or a *scattering-factor-label* appearing on an **F** card. Which of the two it is depends on the *CCSL-word* which follows.

The *CCSL-word* may be one of **AMP FUNC PROD RADF ROTN.**

DATA FOLLOWING ALLOWED CCSL WORDS:

AMP

Data:

A *product-name* (see **PROD** below) followed by two real numbers giving the complex amplitude of the given many-electron configuration labelled *product-name.*

FUNC

Data:

The description of a molecular orbital wave function for atom *label.* The data are:

A *name* (up to 4 characters, with which to label the function).

Two integers, l and m, for the spherical harmonic term in the function.

Either two ($m = 0$) or four ($m > 0$) real numbers, being the complex amplitudes of Y_l^m and Y_l^{-m} repectively.

PROD

Data:

A *product-name* of up to 4 characters followed by two real numbers giving a complex amplitude, then n coded labels defining the one-electron functions in one Slater determinant for an n-electron or n-hole orbital. At present this can only be used for d-electrons(l=2). Each label consists of a signed integer m in the range $-l \leq m \leq l$ indicating the z component of the angular momentum of the electron on the quantum axes followed without an intervening space by either + or − to indicate whether the spin state is parallel or antiparallel to quantum z.

There may be several **W PROD** cards with the same *product-name* where they are the different one-electron products making up a single configuration of the n electron wave-function.

Note:

The routines which read these cards are experimental and are not yet in the main Library.

RADF

Data:

In this case the *label* is a *scattering-factor-label* .

Then on each **W RADF** card there are two integers ITYP and NVAL, followed by the coefficients A and ξ in the expansion of the radial wave function.

ITYP indicates the type of wave function being used:

ITYP=1 is not normalised (e.g. E Clementi & C Roetti, Atomic Data & Nuclear Data Tables, **14** 183)

ITYP=2 is normalised (e.g. R E Watson, Report Solid-State Molecular Theory Group M.I.T., No. 12)

NVAL is the power n in the Slater expansion of the form:

$$U(r) = \sum_i r^{n_i} A_i e^{-\xi_i r}$$

Notes:

The units of length for A and ξ should be atomic units.
As many **W RADF** cards are needed as there are terms in the expansion.

ROTN

Data:

The elements of a matrix relating the quantum axes for atom with the given *atom-label* to the CCSL orthogonal axes. The data are an axis label (**X Y** or **Z**) followed by the direction cosines of the corresponding quantum axis with respect to the crystallographic axes. All three cards are needed for each relevant atom.

EXAMPLES:

```
W Mn1 FUNC x2y2 2 2 .70711 0.0 0.70711 0.0
W Mn RADF 1 2 .27716 3.65559
W Mn RADF 1 2 .03372 10.72370
W Mn RADF 1 2 .29360 5.53874
W Mn RADF 1 2 .45627 2.37383
W Mn RADF 1 2 .08058 1.48214
W Mn1 ROTN    X    0.0000    0.0000    1.0000
W Mn1 ROTN    Y    .7071    -0.7071    0.0000
W Mn1 ROTN    Z    .7071     0.7071    0.0000
```

These cards define a one electron wave-function to be applied to the atom whose label is Mn1 and whose form-factor label is Mn. The angular part of the wave-function is labelled x2y2 and defined as: $\frac{1}{\sqrt{2}}(Y_2^2 + Y_2^{-2})$ with the x, y and z axes of the spherical harmonic functions having the direction cosines given by the **ROTN X,Y**, and **Z** cards with respect to the CCSL orthogonal axes. The radial wave-function to be used is defined by the **RADF** cards as a sum of 5 terms of the unnormalised (Clementi Roetti) type.

```
W   Fe PROD PS12    0.8150    0.0000   2+   1+  -1+  -2+
W   Fe PROD PS12    0.4101    0.0000   2+   1+   0+  -1+
W   Fe PROD PS12    0.4101    0.0000   1+   0+  -1+  -2+
W   Fe PROD PS32    0.7071    0.0000   2+   1+   0+  -1+
W   Fe PROD PS32   -0.7071    0.0000   1+   0+  -1+  -2+
W   Fe AMP  PS12    0.9577
W   Fe AMP  PS32    0.2729
```

In this second example the cards describe a 4 electron wave-function for Fe. It is made up of two terms named PS12 and PS32. **W Fe ROTN** and **W Fe RADF** cards would also be needed to define the function completely.

ROUTINES WHICH READ THE CARDS:

The input of **W** cards is directed by various setting up routines such as WAVSET, MSETUP (not in main Library) and PFSET or by main programs such as FORFAC. These call INPUTW to read the **W** cards and subsequently special routines to interpret the data on each type of card: MOLORB for FUNC, RADFUN for RADF and READRT for ROTN.

NOTE:

W ROTN and **W RADF** cards are also used to define orientations and form-factors used in multipole calculations.

§ § x x § §

X cards
Left free for the user

No routine in the Library assumes that it will find anything on an **X** card, so the user is free to put any information he wishes there. To read and interpret information from a set of **X** cards, see the specifications of:

SUBROUTINE CARDIN	read requested card, A80, to COMMON /SCRACH/
SUBROUTINE RDINTG	read integer
SUBROUTINE RDREAL	read real number
SUBROUTINE RDWORD	read word
SUBROUTINE RDWRDS	read all remaining words on a card
SUBROUTINE RDNUMS	read all remaining numbers on a card.

In the COMMON /CARDRC/ INREAD(24) holds the position in the temporary file (unit IO10) at which **X** cards start, if there are any, and ICDNO(24) holds the number of **X** cards read by PREFIN.

So, having declared COMMON /CARDRC/ and COMMON /SCRACH/ at the start, the sequence:

```
      N=ICDNO(24)
      IF (N .GT. 0) THEN
      ID=INREAD(24)
      DO 1 I=1,N
      CALL CARDIN(ID)
      ID=ID+NYZ
      ........
    a piece of code to extract your information off the card
      ........
1     CONTINUE
      ENDIF
    etc
```

will read and interpret a set of **X** cards.

If the user wants to use **X** cards for more elaborate input, in the same way, for example, as **L** or **M** cards, he could use FUNCTION ONCARD which looks for an **X** card with a given *CCSL-word* , or SUBROUTINE FINDCD which will read the next **X** card with a given *CCSL-word* .

NOTE:

The instruction `ID=ID+NYZ` rather than `ID=ID+1` is needed to skip over any **Y** or **Z** cards which may be interleaved with the **X** cards.

§ § ∞ ∞ § §

Y and Z cards
Comments

A card starting **Y** is copied to the printer output as soon as it is read by PREFIN, then ignored. It can be used to put comments, which the user wishes to appear on his output, into the Crystal Data.

EXAMPLE:

```
Y  RUN WITH LOW FUDGE FACTORS ON ALL TEMPERATURE FACTORS:
```

A card starting **Z** is ignored on input, and so can be used to intersperse comments in the Crystal Data.

EXAMPLE:

```
Z Note this atomic position not as in literature
```

NOTE:

Most Crystal Data cards must appear in blocks, with all cards of the same letter together, but **Y** and **Z** cards may appear anywhere.

§ § *x x* § §

Chapter 4

INPUT AND OUTPUT OF OTHER DATA

INTRODUCTION

Some uses of CCSL require as input only the Crystal Data. More often, a file of what the crystallographer usually thinks of as his *data* is needed as well. This will typically be a list of items, all similar in format, e.g. indexed reflection or intensity data.

At present such files are usually read by FORTRAN fixed format READ statements. CCSL free format read routines are gradually being introduced instead, in cases where the fixed format causes problems. An example is in GRLSQ, the Least Squares Refinement of a structure for which it is required to group together several reflections on input, to compare against one calculated value. Such data are read by SUBROUTINE INOBGR, which is required to read an unknown number of integers followed by a real number (with decimal point). This is not simple using FORTRAN READ.

Another application of free format is the routine RDDATA, which reads h, k, l and a set of other numbers, with the inclusion of a *comment* facility, so that such a list can be headed by some identifying title.

INPUT OF LEAST SQUARES DATA

Wherever possible, the point at which these other data (which we will call *reflection data*) are read is accessible to the user, so that if he has data in some strange format he can still read them. An example of this is in the various Least Squares Refinement main programs like SFLSQ, GRLSQ, MAGLSQ, which actually contain the READ and FORMAT statements for the reflection data. Several alternative formats are already provided; there is also a data input type 0, indicating that the data will be provided, a line at a time, by successive calls to SUBROUTINE QLSQIN, which the user must provide.

In general, any SUBROUTINE QxxxIN is a user-provided input routine for reflection data. (The user need not concern himself with special formats or a routine QLSQIN if he has data in some standard format known to the system; there is a dummy routine QLSQIN in the Library, which will be superseded by any routine of the same name if the user provides one.)

INPUT OF FOURIER DATA

Input of reflection data for Fourier calculations is less accessible, being buried in the various Fourier routines like FOUR1Z, FOUR1D as a call to FOUINP. Again an input type 0 input routine QFOUIN is available. The user would need to write a new QFOUIN if, say, he had non-centrosymmetric data and was not assuming Friedel's Law. He would have to decide what to do with Friedel pairs before presenting them to the standard Fourier calculations, which do assume Friedel.

FILE HANDLING AND THE USE OF INPUT/OUTPUT UNITS

The assignment of FORTRAN units is handled by CCSL. Interactive terminal input and output units, the line-printer and the plotter are assigned to the units named ITI, ITO, LPT, and IPLO respectively, and are given values appropriate to the system by INITIL. They are held in the COMMON /IOUNIT/.

Assignment of files to FORTRAN units is done using FUNCTION NOPFIL to open a file and SUBROUTINE CLOFIL to close it.

Fifteen units are required by CCSL and managed by it; if not forbidden by the particular operating system they are units 20-34. The user of standard programs should find that his input and output are taken care of by calls of NOPFIL already in the system. A user writing or modifying a main program will need to know more details, in order to deal with his own files.

OPENING A FILE TO BE READ

To open a file to be read by a program in the simplest possible way the code:

```
LUN=NOPFIL(1)
```

should be obeyed. The system will respond with the message:

```
Give name of input file
```

to which the user should respond with the file name. Under VMS on a VAX, this can include the full path name and an extension if required but it may be simply the letters (up to 6) of the file name, in which case the extension .DAT is added. LUN will be set to the value of the Fortran unit allocated by the system. The value of LUN should not normally concern the user; he reads the file by code such as:

```
READ (LUN,42)  FRED
```

but if he needs for some reason to open a specifically numbered file he may use SUBROUTINE OPNFIL, thus:

```
LUNIT=14
CALL OPNFIL(LUNIT,1)
```

OPENING AN OUTPUT FILE

To open for writing a file which is not destined for a line printer, the code:

```
LUN=NOPFIL(2)
```

should be obeyed. The file is then written using, for example,

```
WRITE (LUN,43) JOAN
```

To open for writing a file which is destined for a line printer, a VMS user may write:

```
LUN=NOPFIL(2002)
```

When the file is opened this will use:

```
CARRIAGECONTROL='FORTRAN'
```

OPTIONS AVAILABLE WHEN OPENING FILES

Other options can be obtained by obeying:

```
LUN=NOPFIL(MODE)
```

where MODE indicates the type of file to be opened and how to obtain the file-name.

MODE may have up to 5 digits.The least significant, MODE1, indicates the file-type; the second digit, MODE2, shows how to obtain the file name; the third, MODE3, deals with default extensions; the fourth, MODE4, selects formatted or unformatted files and the fifth, MODE5, selects sequential or direct access files.

Character data such as a message or the file name are transmitted to NOPFIL via the COMMON /SCRACH/ which can contain 100 characters, and within NOPFIL is divided into MESSAG of 40 characters and NAMFIL of 60 characters.

The significance of MODE is as follows:

MODE1 = 1 for a read file
= 2 for a write file status new
= 3 for a write file status undefined (UNKNOWN)
= 4 for a write file to be modified (APPEND for sequential files)
= 5 for a scratch file.

MODE2 = 0 give the standard messages for read or write files; machine specific information like the path name and file extension may be included in the user's response.
= 1 use the message in MESSAG; otherwise as MODE2=0
= 2 use the file-name in MESSAG. Report that the file is opened.
= 3 as 2 but do not give file-opened message.
= 4 as 0 but do not give file-opened message.

MODE3 = 0 use the system default for path and default extension .DAT
= 1 use defaults for extension, disc and path-name found in characters 1-4, 5-10, and 11-30 respectively of NAMFIL (up to 40 characters, in common SCRACH after item MESSAG). If the disc or path-name is absent, default as system.
= 2 use the file-name exactly as given in response to the request.

MODE4 = 0 for a formatted file, not expecting FORTRAN carriage control characters (and therefore not a line-printer file)
= 1 for an unformatted file
= 2 for a formatted file with FORTRAN carriage control characters, such as would be sent to a line printer.

MODE5 = 0 for a sequential file
= 1 for a direct access file
= 2 for RECORDTYPE='FIXED' (needed for "GENIE" files).

EXIT FROM NOPFIL

After a successful exit from the call:

```
LUN=NOPFIL(MODE)
```

LUN is set to a small positive number, being the unit number allocated to the opened file by NOPFIL. LUN is zero if no response is obtained (i.e. the RETURN key is pressed) when a file name is reqested. LUN is negative if the file could not be opened, although NOPFIL allows the user to try again after typing errors or other blunders from which it has a sensible way to recover.

The short names (10 characters) are stored in COMMON FILNAM and may be obtained by declaring the COMMON and obeying:

```
CHARACTER*10 NAME,FILNOM
NAME=FILNOM(LUN)
```

CLOSING FILES

When input or output operations are terminated on a particular unit it should be returned to the system by obeying:

```
CALL CLOFIL(LUN)
```

EXAMPLES OF THE USE OF NOPFIL

The main program for a Fourier calculation is given below:

```
C MINIMAL FOURIER PROGRAM.
      CALL PREFIN
      LUNI= NOPFIL(1)
      CALL RECIP
      CALL OPSYM(2)
      CALL SETFOU
   . . . map-producing code - see various main program examples
      STOP
      END
```

It contains one explicit call to NOPFIL; however the call to PREFIN provokes several other calls to NOPFIL, one of which sends the message:

```
Give name of Crystal Data File :
```

to which the user must reply by typing the appropriate file name. Another call to NOPFIL does not result in a message as it just opens an internal scratch file. (If the Libary has been generated for RAL rather than ILL, an early call of NOPFIL asks for the name of the printer output file.)

The call to NOPFIL in the main program sends the message:

```
Give name of input file :
```

and the user should type the name of the file containing his reflection data.

A less trivial example is given in the fragments from the main program ABSMSF reproduced below. ABSMSF is a program which reads a file containing reflection intensities in the special format produced by the sorting programs ARRNGE and ARRINC. The file name has the extension .ARR. It then makes an absorption correction if required and calculates the average of the corrected intensity over all equivalent reflections. Finally it writes an output file containing the mean structure factors and their standard deviations.

```
      PROGRAM ABSMSF
            .
            .
      DIMENSION LUN(2)
      CHARACTER*56 HEDING
      LOGICAL INC
            .
            .
      COMMON /IOUNIT/LPT,ITI,ITO,IPLO,LUNI,IOUT
      COMMON /REFS/K(3,2),LL(48,2),R(500,2),SCALE(2),INC,II,FF(3,2)
      COMMON/SCRACH/MESSAG,NAMFIL
      CHARACTER *40 MESSAG
      CHARACTER *60 NAMFIL
      DATA HEADNG/'(5X,''h'',4X,''k'',4X,''l'',10X,''Fobs'',7X,
     1''DFobs''/)'/
            .
            .
      IF (INC.EQ.0) GO TO 2
C ALTER THE FORMAT FOR FLOATING POINT OUTPUT:
      HEADNG(2:2)='6'
      HEADNG(9:9)='7'
      HEADNG(16:16)='7'
      HEADNG(23:24)='12'
            .
            .
      NAMFIL='.ARR'
      LUN(1)=NOPFIL(101)
      NAMFIL='.SF '
      LOUT=NOPFIL(102)
            .
            .
      STOP
      END
```

The first call to NOPFIL asks the user for the name of the input file and uses the default .ARR if no "." is found in the reply. The second call creates a new file for output with the same name as the input file but with extension .SF (structure factors).

PRINTED OUTPUT FILES: USE OF SUBROUTINE TESTP

There is a SUBROUTINE TESTP which has been written to facilitate the production of long lists of similar items with column headings at the top of each page. Calls to TESTP have the form:

```
      CALL TESTP(LUN,LCOUNT,NTEST,HEADNG,NLINES)
```

where LUN is the FORTRAN unit on which the list is being written, LCOUNT is the number of lines printed since the last page throw, NTEST is used to provoke a page throw if there are less than NTEST lines left on the page.

HEADNG is a character variable containing the page heading (see the example above and also that in the BLOCK DATA sub-program below). NLINES is the number of lines in the heading. HEADNG need not be in COMMON if data are only written to the list from the main program. In ABSMSF it can be seen that HEADNG is modified by the program if INC is .TRUE. (non-integer indices) to accomodate the longer FORMAT required to print the indices in this case.

```
      BLOCK DATA PAGEHD
      COMMON /HEDING/ HEADNG
      CHARACTER*132 HEADNG
      DATA HEADNG/' (''0Seq No       h    k    l   Omega  2Theta
     1  Nu       R        dR      Peak cps  Bkgd cps       DVM
     2  Date    Time''/)'/
      END
```

Chapter 5

LEAST SQUARES REFINEMENT USING CCSL

WHAT WE MEAN BY LSQ

The term LSQ is used here to mean *Least Squares Refinement*, which uses observations of a function to improve the values of some set of parameters on which that function depends.

Refinement involving a crystal structure could be either *standard* single crystal LSQ refinement, in which each observation depends on a single structure factor, or *profile refinement* (PR) for which several structure factors contribute to one observation.

CCSL as generally distributed, and as described in this Manual, does not include the routines which have been written specifically for PR; it includes standard LSQ in the main program SFLSQ, and has a few related LSQ programs like GRLSQ (refining structure parameters using groups of calculated values for each single observed value) and MAGLSQ (single crystal LSQ for magnetic structures).

The PR routines are written to use many of the same CCSL routines as do standard LSQ main programs. The PR programs and Libraries are available separately, with a separate Manual.

CCSL also deals with simpler LSQ problems. For example, the main program FWLSQ fits 5,7 or 9 parameters in the exponential approximation to a scattering curve. The simplicity in this case is that no crystal structure is involved. Another simple case is the refinement of (up to) 6 cell parameters, given an observed list of d-spacings (actually d* squared values, in the main program DSLSQ). This must deal with the constraints which are necessarily imposed on the cell parameters by the space group symmetry.

The essentials of a LSQ problem are:

a) a set of *observations* of something, (the *observed function*) usually with their *estimated standard deviations*, σ's, measured at different values of some argument *ARG*; *ARG* may be h, k, l (for standard LSQ), $\sin\theta/\lambda$, θ (for Rietveld PR) or λ, or energy, or time of flight; it is a quantity which takes different values, and will identify the observation.

   For crystallographic applications the observations are often all of the same physical thing, but this is not necessary. For example, geometric constraints may be introduced by giving bond lengths and/or angles as additional types of observation, with σ's.

b) some *calculated function* involving *ARG*, which defines a mathematical model to be compared with an observation. This calculated function depends on *parameters*, things which may possibly vary in order to improve the fit of the function to its related observations.

   By *fit* we mean minimisation of the weighted sum of squares of differences between observed and calculated function values (where the summing is over the different values of *ARG*). It is desirable to use statistical weights ($1/\sigma^2$) for the differences.

   The theory may be viewed as using the beginning of a Taylor series expansion, and therefore requires that the parameters be close to their *correct* values, making the required shifts so small that their squares may be neglected.

PARAMETERS AND VARIABLES

In LSQ a *parameter* is something which it is sensible to try to adjust. It represents some physical thing like "the x coordinate of the 3rd atom", "an overall temperature factor" or "the scale factor for all observations coded number 2".

A *parameter* is also something with respect to which the function may be differentiated. Differentiation may be analytic, if it is possible (and sensible) to write down such derivatives algebraically. It may also be numerical in awkward cases, using a simple approximation to the derivative involving function values.

For any particular run of a LSQ refinement program, it is unlikely that the user will want to vary every parameter. The subset of parameters which are actually to be varied (i.e. those for which shifts are required) we call *variables.*

Parameters are thus either *fixed* or *varied.* For a fixed parameter, there is no need for a derivative; but for all parameters to be varied, derivatives will be needed.

BASIC VARIABLES

It is often the case that parameters are related, either inherently by the symmetry of the problem, or by a strict constraint applied by the user. If two variables are related, they must not both be refined by LSQ; the process expects the shifts to be independent. Every constraint or relation reduces by 1 the number of variables to be refined.

The subset of variables for which shifts are actually calculated we call *basic variables*, or basics for short.

Those variables which are not basic we call *redundant.* Thus if a problem has n_p parameters, n_f of which are fixed and n_v varied,

$$n_p = n_f + n_v$$

and if, of those n_v, n_b are basic and n_r are redundant,

$$n_v = n_b + n_r$$

STRICT CONSTRAINTS

The existence of n_r redundant variables implies that there are n_r strict constraints on the problem. Derivatives are calculated (and shifts are required) for all variables, not just the basics.

Those constraints which must be imposed because of the crystallographic symmetry are generated automatically by CCSL, and need not be provided by the user. For example, the special position (x, x, x) will give rise to one basic variable, x, with two redundant variables y and z related to it by two constraints, but the user will not need to tell the system this.

The user may impose additional strict constraints. Two constraint types are at present available. The simpler is type 1:

$$a_1 \Delta p_1 = a_2 \Delta p_2$$

that is, constant a_1 times the shift in parameter 1 = constant a_2 times the shift in parameter 2.

Type 2 constraints involve a linear combination of parameters, with constant coefficients, thus:

$$a_1 \Delta p_1 + a_2 \Delta p_2 + a_3 \Delta p3 + \ldots \text{ etc } = 0$$

Note that the way these expressions are written involves differentiating the original constraint. For example, if a type 2 constraint is needed to fix the sum of three parameters to be 3, the constraint

$$p_1 + p_2 + p_3 = 3 \quad \text{becomes} \quad \Delta p_1 + \Delta p_2 + \Delta p_3 = 0$$

Other more complicated types of constraint could be added if needed.

PARAMETER NAMES

The need to fix, vary or relate parameters implies the need to call these parameters by name. The CCSL LSQ facilities allow each problem to be set up individually. Names for the parameters of a problem are given in its main program; the user needs to know what these names are in order to refer to the parameters in his Crystal Data.

Examples of names used in various standard main programs are:

```
Na2 X      Ca B11      SCAL 2      P23 ITF      A*
```

Most names have two parts, which we call the genus and species names; the genus names above are `Na2, Ca, SCAL, P23`, with corresponding species names `X, B11, 2` and `ITF`. The name `A*` is simply a species name.

In LSQ programs which involve structure factors, those structure parameters which belong to a particular named atom have the atom name as genus name, and one of:

`X Y Z B11 B12 B13 B22 B23 B33 ITF SITE SCAT`

as species name. (`SCAT` means the scattering factor, when this is represented by one number and it is sensible to refine it).

MAGLSQ has the additional species names:

`MU MU1 THET THE1 PHI PHI1 PSI1 PSI2 PSI3 PSI4.`

When LSQ programs refer to parameters in output for shifts, correlations etc, they use these parameter names.

More details about parameter names are to be found in Chapter 3 under **L FIX** cards. Note the availability of words like **XYZ** to mean "all three x,y,z coordinates", **CELL** to mean "all six cell parameters", etc. One may also use the word **ALL** to mean "all the members of this genus" (as in `ALL Na3` for "all the parameters of atom Na3") or "all the genera with this species" (as in `ALL X`, or `ALL SITE`, or even `ALL XYZ`).

FAMILIES OF PARAMETERS

Structure parameters such as those introduced in the previous section, being a commonly occurring group in LSQ, are said in CCSL to belong to *family 2* of the parameters. *Family 1* contains all other parameters in the simple LSQ applications we are considering here. Other families are used in PR, and in other LSQ programs.

Family 1, genus 1 is treated by CCSL as special. It contains the parameters for which the species name by itself is enough, like `A*` above; another example is an overall temperature factor for a structure, known as `TFAC`.

Genera 2, 3 etc of family 1 may in general be given any genus name. Within such a genus the species name could be simply an integer, or species could have individual names. This choice is for the writer of the main program. In SFLSQ there are two genera in family 1, the second being named `SCAL` with species numbered 1,2 etc.

Unless the user is writing or modifying a main program, he need not be concerned with the detailed mechanism for numbering parameters. He only needs to know what names he is allowed to use on his Crystal Data for the programs he runs.

EXAMPLES

We illustrate these general ideas by, first, a simple example of LSQ which does not involve structure parameters, and, second, notes on the use of the standard single crystal structure refinement program, SFLSQ.

There are two fundamental questions to be asked of a main LSQ program: on which calculated function is it refining, and which parameters may be refined.

DSLSQ

DSLSQ is a main program which determines the values of up to 6 cell parameters by LSQ fitting, using values of d spacings observed at different values of h,k,l.
The calculated function used is the expression for the square of d*, in reciprocal space:

$$d^*(h,k,l)^2 = (h^2a^{*2} + k^2b^{*2} + l^2c^{*2} + 2kb^*lc^*\cos\alpha^* + 2lc^*ha^*\cos\beta^* + 2ha^*kb^*\cos\gamma^*)$$

The parameters to be varied are the reciprocal cell quadratic products A*, B*, C*, D*, E* and F*, where A*=a^{*2}, D*=$b^*c^*\cos\alpha^*$, etc.

The author of the main program had the choice of names for these parameters, and decided to make them members of family 1, genus 1, having species names A*, B* etc. Alternatively, she could have made them members of family 1, genus 2, with genus name CELL, and species names the integers 1 to 6. There are no other parameters in this simple example.

The calculated function $G_{calc}(h,k,l)$ has been programmed into a routine CALCDS, which is given the values of h,k,l, and returns the value of G_{calc} as defined above, together with all derivatives of G_{calc} with respect to the relevant cell parameters. If a user happened to have data which were observations of, say, d rather than d* squared, CALCDS could be simply rewritten to calculate instead $G_{calc}(h,k,l) = d$ and its derivatives.

Other routines which have been written for this specific application are:
APSHDS to apply shifts to the parameters,
NWINDS to output new Crystal Data containing the shifted parameters,
PARSDS to decide which are the parameters of the problem, and
VARSDS to set up which parameters are variables.

The routines APSHDS, CALCDS, NWINDS, PARSDS and VARSDS may be inspected in a listing of CCSL if further detail is sought.

STRUCTURE FACTOR LSQ

SFLSQ (standard LSQ refinement with possible geometric constraints), GRLSQ (taking groups of input reflections together) and MAGLSQ (for magnetic structures) are all examples of CCSL single-crystal LSQ programs.

Crystallographic LSQ involving structure factors follows much the same pattern as the DSLSQ example above. The calculated function involves a structure factor computation by a routine such as LFCALC, during which derivatives are made with respect to all structure parameters which are variables.

For details of the **L** cards which drive such LSQ programs the user should consult the specification in Chapter 3. An example of the Crystal Data for SFLSQ is as follows:

```
I NCYC 5   PRIN 3   CONV 0.02
L RELA 2   2 Co SITE  1 Mn SITE  1 Sn SITE
L VARY   ALL SITE
L MODE 5     WGHT 1
L SCAL     1    1    1
L FIX  DOMR
N Co2Mnsn at room temp
S     -X,    -Y,     -Z
S  1/2+X, 1/2+Y,      Z
S      Y,     Z,      X
S     -Y,     X,     -Z
A Co   1/4   1/4   1/4    0.3115
A Mn     0     0     0    0.2842
A Sn   1/2   1/2   1/2    0.3386
F Co    1    0.2500
F Mn    1   -0.3730
F Sn    1    0.6228
C 6
E    1    100.    0.05
```

The **C**, **S**, **A**, **F**, and **E** cards are the same as for other programs, giving 3 atoms in special positions in a cubic space group, with neutron nuclear scattering factors, and type 1 extinction corrections on the structure factors.

The **I** card requests 5 cycles of refinement, with printing of the observed and calculated values of the function on the first and last cycles, and terminating before the 5th cycle if the largest shift/σ is smaller than 0.02 (rather than the default of 0.01).

L cards are needed where defaults need to be changed, so **L WGHT** asks for unit weights rather than statistical weights, **L MODE** asks for the special *extinction correction* input mode rather than the standard format for observed input files, and so on.

Each LSQ main program has its own defaults built in as to whether a given parameter should be fixed or varied. In general most structure parameters are by default varied, but site occupation factors are by default fixed, so the `L VARY ALL SITE` card is needed here.

Similarly, the extinction parameters `DOMR` (domain radius) and `MOSC` (mosaic spread) are varied by default if an **E** card is given, so to fix `DOMR` the `L FIX DOMR` card is given.

It is required to relate all three site occupation factors by the linear relation expressed on the **L RELA** card.

INTERPRETATION

The main program SFLSQ interprets the above Crystal Data and deduces that all three atomic positions are so special that all their coordinates are fixed. It makes 9 basic variables, being:

`MOSC`, `SCAL 1`, `SCAL 2`, `SCAL 3` (having deduced that there are 3 scale zones from the **L SCAL** card) `Co ITF`, `Mn ITF`, `Sn ITF` (the isotropic temperature factors, varied by default) and `Mn SITE` and `Sn SITE`.

It makes 10 variables, being all the above plus `Co SITE` (a redundant variable), and records the constraint:

```
2* Co SITE + Mn SITE + Sn SITE = constant
```

By default, as there is no **L REFI** card, the program refines on the modulus of the structure factor. Another input dataset is needed, in the format indicated by the card saying `L MODE 5`; the name of the file containing this dataset is requested interactively.

MAKING OTHER MAIN PROGRAMS

The existing main programs can be adapted to refine different parameters, or to use a different calculated function in the refinement. Examples which already exist are:

MPLSQ, multipole LSQ, in which the scattering density is expressed as a sum of spherical harmonic (multipole) functions, and the coefficients of these functions may be refined. A new family 5 has been defined for them, as there may be different numbers of multipole coefficients for different atoms.

PALSQ, polarization analysis LSQ, in which the observations to be fitted are the components of the scattered neutron polarization for a given input polarization.

If small changes are required, it should be possible to adapt an existing program and its own routines. For major changes, particularly involving new parameters of which there is one per atom like those in family 2, expert help should be sought.

Chapter 6

MAGNETIC STRUCTURES AND STRUCTURE FACTORS

DESCRIPTION OF A MAGNETIC STRUCTURE

In CCSL magnetic structures are described using a propagation vector to define the periodicity, and a magnetic space group to define the relative orientations of spins on different sub-lattices within the non-magnetic unit cell. This approach has the advantage that only the minimum information need be given and multiple unit cells are not required. The propagation vector κ is defined so that the ordered moment **S** on a magnetic sublattice is related to that $\mathbf{S}_l$ in another unit cell at vector distance l (a lattice vector) from it by

$$\mathbf{S}_l = \mathbf{S} fn(\kappa \cdot l)$$

where $fn(x)$ is a periodic function of x such that $fn(x) = fn(n + x)$ for all integer n.

The magnetic space group must be congruent with the crystallographic space group or one of its sub-groups. Each of the elements of the magnetic group acts on the magnetic moment with the rotation and translation appropriate to the corresponding element in the crystallographic group. This may be followed by the operation of time inversion in which case the element is *primed*; otherwise it is *unprimed*.

If $\tilde{R}_s$ and $\mathbf{t}_s$ are the rotation and translation operators associated with one of the elements in the magnetic group and $\tilde{T}_s$ the corresponding time reversal operator (1 or -1 depending on whether time-reversal is invoked) then magnetic moment **S** at vector distance r from the origin of the unit cell implies magnetic moment

$$\mathbf{S}_s = \tilde{T}_s \tilde{R}_s \mathbf{S} \quad \text{at} \quad \tilde{R}_s \mathbf{r} + \mathbf{t}_s$$

There is one such relationship for each of the elements in the magnetic group. One must remember that magnetic moment is an axial vector so that *improper* rotations introduce an additional inversion.

The information needed to describe a magnetic structure is given on **Q** cards which are fully described in chapter 3.

TYPES OF MAGNETIC ORDER

Five types of magnetic order are recognised. They are defined on **Q STYP** cards by the words:

FERO Ferromagnetic and unmagnetised so all possible domains are equally populated.

FERA Ferromagnetic and magnetised in the direction of the z diffractometer axis.

ANTI A simple commensurate antiferromagnetic structure.

AMOD A possibly incommensurate amplitude modulated structure; this differs from **ANTI** in that the relative phases of the modulation at the different symmetrically related sub-lattices may need to be defined.

HELI A possibly incommensurate helical structure. It differs from **AMOD** because the magnitude and direction of both spin components which define the helical envelope must be defined. They are necessarily perpendicular to one another.

MAGNETIC STRUCTURE FACTORS

The magnetic structure factor $\mathbf{M(k)}$ can be defined as the Fourier transform of the magnetisation distribution.

The intensity of neutron scattering is directly related to the component of $\mathbf{M(k)}$ perpendicular to the scattering vector. This is the *magnetic interaction vector* $\mathbf{Q(k)}$. Thus

$$\mathbf{Q(k)} = \mathbf{k} \times \mathbf{M(k)} \times \mathbf{k}$$

Using the description of a magnetic structure just given the magnetic structure factor is written

$$\mathbf{M(k)} = \sum_{l}^{crystal} \sum_{n=-\infty}^{\infty} p(n) \exp i(\mathbf{k} + n\boldsymbol{\kappa}) \cdot \boldsymbol{l}$$

$$\times \sum_{s}^{m} \sum_{p}^{g/m} \sum_{i}^{cell} \tilde{T}_s \tilde{R}_s \mathbf{S}_{ip} f_i(\mathbf{k}) \exp i[\mathbf{k} \cdot \tilde{R}_s(\tilde{R}_p \mathbf{r}_i + \mathbf{t}_p) + \mathbf{t}_s]$$

$f_i(\mathbf{k})$ is the magnetic form factor and $p(n)$ the Fourier transform of the modulating function $fn(x)$. The sum over the lattice vectors is zero unless $\mathbf{k} = g + n\boldsymbol{\kappa}$ where g is a reciprocal lattice vector. At present CCSL only recognises sinusiodal modulations so that $p(n) = 0$ unless $n = 1$. The rest of the equation defines the unit cell magnetic structure factor; its projection on the plane perpendicular to the scattering vector is the quantity calculated by the magnetic structure factor routines FMCALC and LMCALC.

MAGNETIC DOMAINS

Whenever the magnetic symmetry of a structure is less than the symmetry of the nuclear structure magnetic domains can occur. Different parts of the crystal conform to one or other of the possible domains. The different types that occur are: configuration domains, 180°domains, orientation domains and chirality domains.

Configuration domains occur when the propagation vector $\boldsymbol{\kappa}$ is not invariant under one or more of the symmetry operations of the space group. Magnetic structure factors are calculated for the configuration domain whose propagation vector is given on the **Q PROP** card.

180°domains occur for all structures for which $\boldsymbol{\kappa} = 0$; the magnetic structure factors for pairs of 180°domains are reversed in direction.

Orientation domains occur whenever the magnetic group has symmetry lower than the configurational symmetry i.e. when there are one or more **Q NSYM** cards.

The subroutines FMCALC and LMCALC calculate magnetic interaction vectors for all the orientation domains and store them in COMMON/QCAL/. They also calculate the mean squared interaction vector for all orientation domains assuming equal populations. This is the quantity used in the MAGLSQ magnetic structure refinement and printed out as F_m^2 by the magnetic structure factor program GETMSF.

In cases where chirality domains occur their interaction vectors are also calculated and are included in the averages and in /QCAL/.

Chapter 7

USING THE SYSTEM

We assume first that the user has access to a computing installation on which CCSL has already been compiled and made into a Library.

Any discussion of the details of running jobs must naturally refer to specific items such as file names, which differ on different computers. Most current users of CCSL in fact use VAX-VMS computers and the examples in general refer to them. Users of other operating systems such as UNIX or DOS should take them only as a rough guide.

RUNNING "BLACK BOX" JOBS

A standard main program such as ARRNGE (which groups together the results of single crystal diffraction measurements) will have been precompiled and linked using CCSL as a search Library. An executable file such as ARRNGE.EXE should therefore exist. The user wishing to run this as it stands, as a ***black box***, first puts his Crystal Data into some file, named, for example, TESTAR.CRY.

He would also have files containing data to arrange, called, say, REFLN1.C5, REFLN2.C5 and REFLN3.C5, and a list of the numbers of reflections to reject in, say, REFLN2.REJ.

The following sequence would run the program from an on-line terminal. The dialogue in italics is to be typed by the user.

RUN ARRNGE

```
Give name of Crystal data file  TESTAR
Give name of file containing rejection list   REFLN2
Give name of reflection file   REFLN1.C5
Give name of next reflection file (or c/r to end)  REFLN2.C5
Give name of next reflection file (or c/r to end)  REFLN3.C5
Give name of next reflection file (or c/r to end)

Sort started   . . .   Sorted 1485 records

Give name for output file of sorted reflections   ARROUT
FORTRAN STOP
```

In general, the user supplies file names on request. The file extension for Crystal Data files defaults to .CRY or .CCL . General data files have default extension .DAT although some default extensions are associated with particular data types. (For example, for ARRNGE, .FLI is the default for DTYP=1, meaning polarised neutron flipping ratios). The extension .C5 is not one of these, hence the .C5 file extensions above were typed explicitly. Exactly which defaults have been used is under the control of the writer of the main program, but routine NOPFIL should in general allow the user to try again if he types a file name it cannot handle.

THE "LINE PRINTER" OUTPUT FILE

Every job will open a file which is intended for the line printer. On the VAX cluster at ILL this file is given the name of the main program and extension .LIS. On the VAX at RAL, the user will be asked to name it, and if he responds by typing RETURN the file-name will default as above. So at RAL, there will be an extra line at the start of the dialogue above saying, for example:

```
Give name for Output  ARTEST
```

The opening of file LPT is done by a call of OPNFIL, from SUBROUTINE INITIL, usually from the call of PREFIN which occurs near the start of most CCSL programs. If the user wishes to open a specific unit number, say 8, he must set `LPT=8` before the `CALL PREFIN`.

If he does not do this OPNFIL (via NOPFIL) will choose its own unit number.

GRAPHICAL OUTPUT

The form of graphical output is very dependent on the hardware used to produce it and the local software used to drive that hardware. CCSL addresses this problem by chanelling all calls to the plotter hardware through a single SUBROUTINE PIGLET.

There is a separate section of the CCSL Master File which contains various PIGLETs which have been written to drive different devices, such as Tektronix 4010 terminals and BENSON plotters. Special PIGLETs for GKS, PostScript and PGPLOT interfaces are also included. A user who wishes to compile a program which uses CCSL graphics should extract the most suitable SUBROUTINE PIGLET from the Master File, modify it if necessary, and include it in his link statement together with any graphics libraries called by this particular PIGLET.

In many cases the output will drive a graphics terminal directly and local system facilities will be used to obtain hard copies. Output files destined for centralised plotters will usually be created and named by the local system software.

For cases in which the graphical output is sent directly to a file by the CCSL graphics interface, (for instance when using the PostScript PIGLET), there is a CCSL output unit IPLO reserved for this purpose. In this case the output file will be named by CCSL, usually with the name of the program and an extension approriate to the type of output (e.g. .PS for a PostScript file).

RUNNING NEW OR CHANGED PROGRAMS

If a new main program is written, or if new versions of CCSL routines are made, the progam must be recompiled and relinked. For VAX-VMS the commands required are:

```
$FOR PROG,SUB1,SUB2,.....,SUBN
$LINK PROG,SUB1,SUB2,.....,SUBN,'CCL
```

where CCL is a symbol defined to be the name of the CCSL Object Library (.OLB) file. At ILL an appropriate setting of CCL would normally have been made during login. At RAL it would be something like:

```
$ CCL=="LIB$DISK:[CCSL]LIBMK4/LIB"
```

where LIBMK4.OLB is the CCSL object library. Graphics Libraries may also be needed.

After that the job may be run by:

```
RUN PROG
```

USE OF SCRATCH COMMON

Two areas of labelled scratch COMMON are used in CCSL routines, and can with certain restrictions be made use of by user programs; they are called /SCRAT/ and /SCRACH/.

COMMON /SCRAT/ has a maximum length within CCSL of 10201 items (integers or real numbers).

COMMON /SCRACH/ is a character buffer to contain at least 80 characters, and usually 100.

It is used extensively by the card reading routines. The CHARACTER *80 item ICARD in COMMMON /SCRACH/ is used as the buffer in which the routine CARDIN places a character string from the Crystal Data, and where all the CCSL free format subroutines expect to find it. It is this character string which is interpreted by the various INPUTx routines, and which could be interpreted by user programs.

In general it must not be assumed that either of the scratch common areas will be unchanged by CCSL routines; the user should in general avoid them.

MAIN PROGRAMS IN CURRENT USE

A list follows of some main programs written for CCSL and available in the Master File. Appendix D will give more details.

ABSMSF	Calculates mean structure factors from a file of reflection intensities which has been written by ARRNGE or ARRINC, with correction for absorption.
ARRNGE	Arranges data measured for Bragg reflections so that remeasurements and symmetrically equivalent measurements are grouped together.
ARRINC	A similar program to ARRNGE for incomensurate structures.
BESTPL	Calculates the best plane through a group of atoms.
BONDS	Calculates bond lengths and angles; also prepares **L SLAK** cards for LSQ slack constraints.
CALMSF	Calculates magnetic structure factors for a given set of reflection indices.
CALQSF	As above but prints out the magnetic interaction vectors for each domain separately.
COMPAR	Compares two sets of data output by ARRNGE or ARRINC and computes a scale factor between them.
CRYPAD	Performs data processing for CRYOPAD at ILL.
DEPOS	Prints structure factor material in a form suitable for deposition at the time of publication.
DSLSQ	Refines the lattice constants from a list of values of d^* squared.
D3OP90	Data processing program for the D3 polarised neutron diffractometer.
EXTCAL	Calculates the 4 coefficients needed for extinction corrections and writes a file suitable for SFLSQ input.
FORFAC	Calculates tables of form-factors and radial densities from Slater type wave-functions. Produces **F** cards for input to other CCSL programs.
FOULAS	Minimal Fourier program using GKS to draw maps on a screen or laser printer.
FOURPL	Minimal Fourier program to draw maps on a Benson type plotter.
FOURPS	Minimal Fourier program to draw maps using PostScript.
FWLSQ	Least squares determination of Forsyth and Wells form factor coefficients.
GENDAT	Generates a complete set of *hkl* data from a set in the asymmetic unit.
GETMPL	As GETSFZ (see below) but uses multipole form factors.
GETMSF	As GETSFZ (see below) but calculates both magnetic and nuclear structure factors.
GETSFZ	Produces a list of calculated structure factors sorted in ascending order of $\sin\theta/\lambda$, or in layers about some given zone-axis.
GPCARD	Generates **M** cards for a general Fourier section through three chosen atoms.
GRLSQ	Least Squares Refinement on groups of structure factors.
LINFOU	Calculates the Fourier density along a given line.
MADUBM	Writes **D** cards giving the CCSL UB-Matrix from data read from a "NUMOR" on the ILL data-base.
MAG3D	Makes perspective drawings of magnetic structures.
MAGLSQ	Least Squares Refinement of magnetic structures.
MPLSQ	Least Squares Refinement with multipole expansion of the scattering density.
NONCRY	Inverts 3x3 matrices; demonstrates the non-crystallographic use of CCSL.
PALSQ	Least Squares Refinement of magnetic structures from polarization analysis data.

POLSQ	Least Squares Refinement of magnetic structures from polarised neutron flipping ratios.
QUINTO	Preliminary processing and correction of CRYOPAD data.
SFLSQ	Single crystal structure factor LSQ.
SORGAM	Calculates gamma from the flipping ratios on files output by ARRNGE.

More main programs are being added all the time. Although some of these are for applications specific to their authors, they may well help the new user tackle some related problem.

The user should almost certainly find that the totality of CCSL code (Library routines and main programs) will be available on file in FORTRAN form. He may find it useful to work with printouts of these files, but should be cautious of printing out the whole Library, which is long. He may also find that there are several versions of the files in existence, if some section is actively being updated by one of the authors. If he has at his computing establishment a resident CCSL expert, it would be as well to consult her or him before embarking on any major new project.

J.C. Matthewman
P.J. Brown
August 1992