Rudolf Hundt
Bonn, 1979

KPLOT

A Program for Plotting and Analysing Crystal Structures

Version 9

Technicum Scientific Publishing
Stuttgart, 2016

Preface

One of the major challenges in crystallography and solid state chemistry is the construction, analysis, handling, and depiction of the structures of the crystalline compounds that have been synthesized by the experimental chemist or proposed by the crystallographer, theoretical chemist or physicist, or materials scientist. If high-quality single-crystals are available, the structure can usually be deduced directly, but in many cases, the data are incomplete or too noisy. In such cases, the best we can hope for is a physically and chemically plausible model of the compound's structure. But even if the structure is available in all detail, it is still necessary to visualize and analyze the structure, in order to be able to understand it from a chemical and physical point of view.

This includes not only the determination of the local atom coordinations or the identification of elementary building groups and other structural elements, but also the comparison with known or proposed structures in other (chemically related) systems. Here, one particular aspect is the determination of structural relations between different modifications of the same compound, e.g. at standard and elevated pressures, in order to gain insight into the transformation process. Another task one frequently encounters is the need to complete a seemingly "solved" crystal structure: Often one is able to locate the heavy atoms in a structure from e.g. powder diffraction or single crystal data, but the signals from the light atoms (hydrogen, but also oxygen or fluorine) are too weak and thus overshadowed by those of the heavy atoms. As a consequence, one must add the positions of the light atoms by hand using empirical rules about bond lengths, bond angles, typical coordination behaviour etc., preferably in an easy and straightforward yet fully controllable fashion.

But this partial reconstruction of a structure on the computer is only the first step in another challenge of the crystallographer, i.e. the construction of a whole new structure model by hand guided by physical and chemical intuition or partial structural information. Of course, nowadays, new crystal structures are proposed by the millions using a large variety of structure prediction programs, which typically start from a set of randomly placed atoms in an arbitrary variable (periodically repeated) simulation cell. Usually, the search is completely unbiased, i.e. in particular no symmetry is assumed and the calculations are performed in P1. Thus the structure(s) generated by the search are presented as optimal arrangements of the atoms in some optimal simulation cell but without any information about the symmetry of this configuration or about the standard crystallographic unit cell of the structure. Finding all the symmetries, identifying the corresponding space group and transforming the structure to its appropriate standard setting is clearly an important and non-trivial task.

Over the past forty years, Dr. Hundt has developed a computer programm to address these kinds of questions. The KPLOT-program allows the user not only to visualize a structure, but also to analyze it in depth and to modify or construct crystal structures as desired in a detailed and systematic fashion. The code contains tools to e.g. determine symmetry properties, compare pairs of crystal structures and free clusters, or parts thereof, deduce symmetry relations between structures, perform crystallographic transformations, extract subsets of periodic structures, and allow for a detailed geometrical analysis of a given structure.

Of course, there exist more recent programs focussing on e.g. the esthetically pleasing depiction of a crystal structure. Similarly, there exist some other commerical or non-commercial programs that can perform one or two of the tasks listed above. But many of the routines above are only available in KPLOT, and only KPLOT contains the whole bundle in one package. It thus constitutes a great toolbox for those researchers who want and need to analyze and work with structures in detail, in the process employing the whole range of features available in KPLOT. This program is also particularly suitable for the automated analysis of a multitude of structures,

e.g. as generated during MD/MC simulations or global energy landscape explorations of chemical systems, since it can be operated via external scripts.

In many ways, KPLOT reflects the history of the field of computer-assisted analysis of crystal structures, from early needs to visualize and print structures over the completion and construction of crystal structures from incomplete measurement data to modern demands of automated crystallographical analysis of structure files in order to generate databases of existing and hypothetical crystal structures and subsequently to search and sort these databases with respect to geometrical and crystallographic features. Most of the features in KPLOT have been created in direct response to the needs of experimental and theoretical researchers facing difficult issues of chemical and crystallographical interpretation regarding their synthesized and simulated crystalline compounds. Thus, in the hands of Dr. Hundt, KPLOT has continued to evolve and grow with the times, from its roots as a simple drawing program to a highly valuable code for crystal structure analysis that remains a powerful tool for every researcher interested in studying crystal structures.

Prof. Dr. J. Christian Schön
Max Planck Institute for Solid State Research, Stuttgart, Germany
Dr. rer. nat. Dejan Zagorac
Vinča Institute of Nuclear Sciences,University of Belgrade, Belgrade, Serbia

July, 2016

KPLOT

A Program for Plotting and Analysing
Crystal Structures

Version 9

Rudolf Hundt

Bonn, 1979
(Revision date: 20.09.2016)

Contents

Introduction

The program KPLOT is used to draw and analyse crystal structures. The approach of the program to drawing structures is similar that of the program ORTEP. Meanwhile most tasks available with ORTEP have been integrated. Version 9 has the new feature that two structures can be handled simultaneously.

The pictures created with the program can be treated as simplified "ball and stick" models. The balls are drawn in projection as circles while the sticks which are supposed to symbolize bonds or indicate coordinations of atoms are represented as simple lines or double lines (tubes).

The geometry section of the program allows

- to select an arbitrary subset of the structure
- to define an arbitrary viewpoint
- to produce a stereoscopic pair of pictures in a simple manner
- to insert commands during the drawing stage
- to calculate coordinates which are given by geometric relations, providing a large number of commands for this purpose,
- to handle two structures simultaneously, and to compare them.

Due to the similarities to the program ORTEP it is possible to develop a drawing in dialogue mode and to produce with the results at once a final ORTEP plot.

In order to keep the program as flexible as possible it has been structured in such a way that during input no specific sequence has to be followed. In particular, in the case of erroneous commands, it is usually possible to remove the offending command without deleting previous commands.

The input for the program is entered format free in the form of commands. A single command consists of a keyword, 1-4 characters in length, that may be followed by a number of parameters. Note that one line may not exceed 80 characters. There are different data types which have to be taken into consideration:

- Strings: They consist of a sequence of characters. If they contain blanks or one of the following special characters ; ! = * , ( ) they have to be enclosed in quotes. Quotes within a quoted string have to be doubled. *Example:* `'Don''t do that!'`
- Integer numbers: Contiguous sequence of digits, which may be preceded by a + or - sign. *Examples:* `123 -17`
- Real (floating point) numbers: Contiguous sequence of digits, which may be preceded by a + or - sign, but they may contain a dot (decimal point). If the dot is omitted, it is assumed to be present after the last digit. Furthermore, the (capital) letter E may be added to real numbers, followed by an integer number. This is interpreted as an exponential representation of that number.
  Examples: `123 -17.8 1E3 (=1000) 1.3E-2`
- Fractions: They consist of two real numbers separated by a / (slash) without any blanks. These numbers are converted into real numbers. *Examples:* `1/2 -2/3 109.5/2`
- Codes: Basically, special type of integer numbers. They may, if greater than 100000, contain a character a,...,j or A,...,J in the last but one position (s. p. 4).
  Examples: `25 255501 37456J2`
- Hexadecimal numbers: There are a few occasions where the input has to be given using hexadecimal numbers. This is specially noted at these commands.

The special characters mentioned above have the following meaning:

Blank One (or more) blank(s) are separators for parameters.
Example: `TZ  0 1 0  0 0 1  1 0 0`

, Commas may also be used to separate data, but they are also used to indicate the omission of data. n commas are equivalent to $(n-1)$ asterisks (see below).
Examples: `Z 10,11,12` `UDST,,,-1`

; Terminates a command. It is used if more than one command is given in one line.
Example: `AE 2; SE 1`

! Indicates that a comment is following. The rest of the line is ignored.
Example: `Z 12.3 ! cubic cell`

= Indicates that the input stream is continued on the next line. The rest of the line is ignored. *Example:*
```
ACIM 2 3 4 =
5 6 7 8
```

***** Space holder that indicates that data are omitted and the program should use default values. *Example:*
`Z 11.4 16.8 6.3 * 112.3 *`

(and **)** enclose comments. All characters within backets are ignored, even line ends. Therefore one has to take care to ensure a valid structure (equal number of opening brackets and closing brackets in the correct sequence).

Often it is not necessary to input all parameters because the program contains a large number of default values, which are used during calculations if no explicit input is given for these parameters.

In order to create a drawing, five groups of commands are provided:

- Data defining the configuration: To this group belong the input of the lattice constants, the coordinates of the atoms of the asymmetric unit, the radii of the atoms, the type of the atoms, the symmetries, and the lattice type.
- Composition of the drawing. The atoms which are supposed to be drawn and connected have to be placed into the so-called "code list". Commands belonging to this group are necessary for this process.
- Optimisation of the view. In order to achive an optimal view of the structure, the coordinate system of the viewer can be placed in an arbitrary fashion with regard to the structure. The x-y-plane of this so-called "free" coordinate system is the plane of drawing: the coordinate system is connected to the output-unit, i.e. the paper (or the screen) onto which the structure is projected. This is done in such a way that z=0 is the plane of the paper, the x-axis goes to the right, the y-axis goes up, und the z-axis points towards the viewer (right-handed coordinate system).
- Plotting commands. Atoms and their connections, polyhedra etc., are drawn using these commands as long as the atoms defining the structure elements are in the code list.
- Auxiliary commands. During the design of the final drawing certain additional tasks may occur, e.g. the necessity to update tables, list intermediate results etc., for which a number of auxiliary commands have been provided.

In addition to creating drawings, the program can be used for a number of constructive tasks. In particular, we are dealing with points in space, which may be determined by geometric construction procedures. This is a wide field covering both simple calculations and sophisticated generation of structure models. Powerful tools are supplied for some tasks such as finding symmetries and idealisation of molecules.

For the PC's there are several versions of KPLOT available. They are distinguished by the driver through which the pictures are exported to the printer port or written to files.

Program	Device
KPLOTH	HP Laser-Printer III P and HPGL code
KPLOT	HP Laser-Printer II P and .PCX files
KPLOTF	HP DeskJet 550C and .PCX files (color)

The drivers are activated by the special commands 'PXLG' or 'HPGL'.

In all versions except for IBM (CMS or MVS) files are handled as follows: When the program is started, only the files with the numbers *ntin* (5) and *ntout* (6) are opened. If other files are to be used during the execution of a command, the program will ask for the name of the particular file at the first attempt to read or write it. Alternatively, the files may be opened using the command 'OPEN' before working with them.

Example: A file with the name NACL.DAT contains the commands for creating the drawing of the NaCl structure. This file may be processed by:

```
EAE 1 NACL.DAT
```

or

```
OPEN 1 NACL.DAT
EAE 1
```

or

```
GET NACL.DAT
```

It is essential that the last line in such a file contains a command that switches the control back to the keyboard. If this command is 'EAE 5', the control is transferred but the file is still open. If 'CLSE' is used, the file will be closed. On some machines an attempt is made to open a file having the name KPLOT.STP in the current directory while starting KPLOT. If such a file is found, it is read in and processed as setup before any other command. The last line in this file *must* be `CLSE;DLG` .

Atom Designator Code

Atoms that are to be drawn, are coded by the program in the shape of so-called designator codes. Only such codes are stored in a code list. A code is a number with 6-9 digits with the following meaning:

	NR	TA	TB	TC	SYMNR
digit	9 8 7 6	5	4	3	2 1

NR is the position of an atom in the parameter list.

SYMNR is the position of a symmetry in the table of symmetries. Due to the fact that the symmetry list can store up to 192 symmetries, digit 2 may range from $0, \ldots, 9, \mathrm{A}, \ldots, \mathrm{J}$ $(= 19)$

TA, TB and **TC** are translations increased by 5 in the directions of the basis vectors of the unit cell, e.g. "5" means no translation, "6" a translation by +1 etc.

In order to obtain the coordinates of an atom given by a code, divide the code in the way given above into the numbers NR, TA, TB, TC, and SYMNR. Then take the parameters of the atom NR, apply the symmetry SYMNR, and add to x TA-5, to y TB-5, and to z TC-5, respectively.

Example:

The code 345706 denotes an atom whose coordinates are found at the 3rd position in the parameter list. To these parameters the symmetry is applied, which is found at the 6th position in the list of symmetries. The results are finally shifted by -**a** and 2**c**. Assume the coodinates (.1,.2,.3) on the 3rd position of the parameter list and the symmetry $-x, 1/2+y, 1/2-z$ at the 6th position in the symmetry list, then the code 345706 gives an atom with the coordinates (-1.1,0.7,2.2).

For certain commands of the program, e.g. when addressing atoms generated by the application of symmetries, the input of such designator codes is necessary. There is a simplification: If no symmetry and no translation is to be applied, i.e. the coordinates of an atom are to be taken directly from the parameter list, the specification of the number (position in the parameter list) will suffice. The program "completes" the number to the full (default) code by adding "55501". Therefore, it is mandatory that the identity is always placed in the first place of the symmetry list. Warning: there are many commands where only atom numbers (refering to the parameter list) are allowed as input.

A minus sign before a number where a code is expected has a special meaning: if not stated otherwise, this negative number $-n$ is replaced by the code at the position n of the code list. But be careful! There are commands that re-order the codelist. Usually this feature is used only to label atoms using 'PSN' or 'PN' immediately after drawing.

There are several commands where zeroes are allowed when codes are expected. If zeroes are entered, the codes are taken from a special list, which is used to store groups of codes (so-called "mouse list" or "group list of codes"). This list is often called "mouse list", because it may be filled using some pointing device like a mouse or a graphic cursor. For details c.f. the command 'LGC'.

Description of commands in detail

In the following description the semicolon represents the end of the command. If operating in dialogue mode (DLG), press the ENTER or RETURN key at this point. The semicolon may also be used as a command delimiter, if more than one command is to be written in one line.

Important options

Options do not perform any action by themselves, but affect actions of other commands. Some important options are mentioned here, but most options will be described in context of the commands they affect.

Number of structure

NS n ;

With this command, one can select a structure (1 or 2) to become the current structure. Default for n is the number of the structure, which is not the current one, i.e., entering `NS;` switches to the other structure.

Copy foreground structure

CFS ;

The background structure will be replaced by the foreground structure.

Dialogue mode on/off

DLG resp. NDLG ;

KPLOT is operating in dialogue mode when started, i.e. the option 'DLG' is activated. The behavior in this mode is as follows:

A '>' after the number of the foreground structure, e.g. `1>`, is written to the screen indicating that a command is expected. If a valid command is given without parameters, although parameters may be given, the program asks for them by listing the parameter abbreviations from the command description. If one wants to use the default values, one may enter a semicolon or comma. If 'NDLG' is in effect, no prompt is written nor are parameters asked for. This is usually the case when reading from a file.

Print commands (or not)

OKD resp. NOKD ;

If 'OKD' is in effect, all lines read are echoed. 'NOKD' suppresses echoing. The latter option is activated at startup.

Protocol

PROT n ;

Similar to 'OKD', lines read are echoed, but to unit no. n (e.g. a file) instead to screen. A value of zero (initial setting) for n stops the writing. See also 'LOG' p. 46.

Control amount of output

MORE n ;

The amount of output produced by some commands while being executed may be controlled by setting n to a higher value (default: $n = 0$). E.g. if $n = 1$ while transforming using 'TZ', the inverse matrix will be shown.

Title

Title

T '*title*' ;

Users are encouraged to give a title. If the structure is saved by 'PUT' or 'PUTC', the title is written to file. It has proved to be easier to recognize a structure, if some meaningful title had been chosen. The title may consist of up to 72 letters which have to be enclosed in quotes. Default is blank.

Print title

OT ;

The current title is echoed. See also 'ECHO'.

Cell constants

Cell constants

Z $a, b, c, \alpha, \beta, \gamma$;

Using this command a unit cell is defined, establishing a triclinic system in the process. From the data given the program determines the basis of the direct space lattice, and the basis of the reciprocal space lattice. Default values at start-up are: $a = b = c = 1$, $\alpha = \beta = \gamma = 90$. To simplify the input, the program automatically sets $b = a$ and $c = a$ if a zero is entered for b or c or no input is given. 'Z' may also be used to change the values of the current cell. Warning: The current orientation of the free coordinate system is lost when applying 'Z'.

Example: `Z 5.6` defines a cubic cell with $a = 5.6$

Lattice constants via basis vectors

ZBAS $b_{11}, b_{21}, \ldots, b_{33}$;

If the basis vectors of the direct lattice are supposed to be in an orientation different from the one that is implied by the command 'Z', they need to be entered using the command 'ZBAS'. This might occur after certain transformations have been applied. In that case, the lattice constants are computed from the three basis vectors, (b_{11}, b_{21}, b_{31}), (b_{12}, b_{22}, b_{32}), and (b_{13}, b_{23}, b_{33}).

Print cell constants

OZ ;

The current values of the cell parameters, the reciprocal cell parameters, and the volume are printed.

Interchange direct with reciprocal cell

XZ ;

Lattice parameters may also be entered via the reciprocal cell. Using 'XZ' the reciprocal lattice parameters and direct lattice parameters are interchanged and thus stored in the correct place.

Symmetries

The symmetries are numbered in the order in which they have been entered, and stored in the symmetry list. The last two digits of the atom designator codes refer to the position in this list. Up to 192 symmetries are allowed. A symmetry is coded internally by 12 numbers. These 12 numbers define a translation vector $T = (t_1, t_2, t_3)$ and a rotation matrix $\mathbf{S} = (s_{11}, \ldots, s_{33})$. The transformed coordinates x' of a point x are given as follows:

$$x' = \mathbf{S}x + T$$

Symmetry as matrix

S $t_1, s_{11}, s_{12}, s_{13}, t_2, s_{21}, s_{22}, s_{23}, t_3, s_{31}, s_{32}, s_{33}$;

A symmetry is defined explicitely by 12 numbers. E.g., if one wishes to code the symmetry $-x, 1/2+y, 3/4-z$, one has to enter the following numbers:

```
0,-1,0,0,   .5,0,1,0,   .75,0,0,-1
```

Symmetry (symbolically)

SY '*symmetry*' ;

The symmetries can also be introduced using a symbolic notation. Here the following rules have to be obeyed: The general form of a symmetry is

$$'a_1, a_2, a_3'$$

where the a_i have the following form:

$$[t][+|-][x][+|-][y][+|-][z] \ .$$

Here t is one of the 11 fractions: 1/12, 1/6, 1/4, 1/3, 5/12, 1/2, 7/12, 2/3, 3/4, 5/6, 11/12. The slash has to be included. The translation may also be given as decimal number, e.g. 0.5.

The brackets symbolize that the input may be omitted.

Examples:

```
'X,Y,Z'
'-X,1/2+Y,-Z'
'X-Y,2/3+Y,11/12-Z'
```

The following points have to be considered:

- The order of the translation X, Y and Z is arbitrary, e.g. the following symmetries are equivalent:
  X-Y,Y+1/2,Z = -Y+X, 1/2+Y, Z .
- The plus sign may be omitted, i.e. the following codings are equivalent:

  X+Y,Y,X+Z = XY,Y,XZ .

If an error is detected, an error message is issued, with an error code having the following meaning:

No	Meaning
1	X, Y or Z is missing
3	Translation coded wrongly
5	Illegal character
6	Translation duplicate
7	Sign illegal
8	Sign multiple
9	Symmetry incomplete
10	Coefficient not allowed

Lattice type

GTY *type* ;

Enter one of the following characters for *type*:

F	face centered lattice
I	body centered lattice
A	A centering
B	B centering
C	C centering
Z	center of symmetry in (0,0,0)
R	cyclic permutation
Q	rhombohedral obverse setting on hexagonal axes
S	rhombohedral reverse setting on hexagonal axes
L	apply last symmetry to all others

Note that 'GTY' automatically produces new symmetries. In order to have all symmetries for later calculations 'GTY' must always be supplied after (and not before) all other symmetries have been entered.

Centering

C x, y, z ;

The given centering x, y, z is put into the list of symmetries and the symmetry group is completed. The main purpose of this command is the introduction of unusual centerings. This may be necessary when using 'ZIDL'. *Example:* `C .5 .5 0` is the same as `GTY C`.

Space group by Hermann-Mauguin symbol

HMS '*symbol*' ;

The space group having the Hermann-Mauguin symbol *symbol* is loaded into the symmetry list. The symbols listed below refer to the 3rd edition of the International Tables ("old" tables, 1969). However, not all space groups are uniquely defined by the symbol alone. In such cases we proceed as follows: To the symbols of monoclinic space groups the unique axis is added; to the symbols of rhombohedral space groups the letter R, H, or Hr (hexagonal setting reverse) respectively is added. In all other cases, a Z is added, if the center of symmetry lies in (0,0,0), otherwise the letter S. These rules apply only to the non-unique space groups.

The following table contains all space groups provided in alphabetical order. Each symbol is followed by its number (see command 'RG'). The letter W indicates that the symbol is not unique.

A112/A	1503	A12/A1	1504	A2/AB	1504		
A2/AC	1503	ABA2	41	ABM2	39		
AMA2	40	AMM2	38	B112	501		
B112/B	1501	B112/M	1201	B11B	901		
B11M	801	B2	501	B2/B	1501		
B2/M	1201	BB	901	BM	801		
C12/C1	1502	C12/M1	1202	C121	502		
C1C1	902	C1M1	802	C2	502		
C2/C	1502	C2/M	1202	C222	21		
C2221	20	CC	902	CCC2	37		
CM	802	CMC21	36	CMCA	64		
CMCM	63	CMM2	35	CMMA	67		
CMMM	65	F222	22	F23	196		
F4-3C	219	F4-3M	216	F4132	210		
F432	209	FD3-	203	W	FD3-C	228	W
FD3-CS	22801	FD3-CZ	22802	FD3-M	227	W	
FD3-MS	22701	FD3-MZ	22702	FD3-S	20301		
FD3-Z	20302	FD3C	228	W	FD3CS	22801	
FD3CZ	22802	FD3M	227	W	FD3MS	22701	
FD3MZ	22702	FDD2	43	FDDD	70	W	
FDDDS	7001	FDDDZ	7002	FM3	202		
FM3-	202	FM3-C	226	FM3-M	225		
FM3C	226	FM3M	225	FMM2	42		
FMMM	69	I12/C1	1505	I2/C	1505		
I212121	24	I213	199	I222	23		
I23	197	I4	79	I4-	82		
I4-2D	122	I4-2M	121	I4-3D	220		
I4-3M	217	I4-C2	120	I4-M2	119		
I4/M	87	I4/MCM	140	I4/MMM	139		
I41	80	I41/A	88	W	I41/ACD	142	W
I41/ACDS	14201	I41/ACDZ	14202	I41/AMD	141	W	
I41/AMDS	14101	I41/AMDZ	14102	I41/AS	8801		
I41/AZ	8802	I4122	98	I4132	214		
I41CD	110	I41MD	109	I422	97		
I432	211	I4CM	108	I4MM	107		
IA3	206	IA3-	206	IA3-D	230		
IA3D	230	IBA2	45	IBAM	72		
IBCA	73	IM3	204	IM3-	204		
IM3-M	229	IM3M	229	IMA2	46		
IMM2	44	IMMA	74	IMMM	71		
P1	1	P1-	2	P112	301		
P112/B	1301	P112/M	1001	P1121	401		
P1121/B	1401	P1121/M	1101	P1121/N	1405		
P11B	701	P11M	601	P12/C1	1302		
P12/M1	1002	P121	302	P121/A1	1403		
P121/C1	1402	P121/M1	1102	P121/N1	1404		
P1211	402	P1C1	702	P1M1	602		

P2	3	W	P2/B	1301		P2/C	1302	
P2/M	10	W	P2/MB	1002		P2/MC	1001	
P21	4	W	P21/A	1403		P21/B	1401	
P21/C	1402		P21/M	11	W	P21/MB	1102	
P21/MC	1101		P21/NB	1404		P21/NC	1405	
P21212	18		P212121	19		P213	198	
P21B	402		P21C	401		P21NB	3303	
P21NM	3102		P222	16		P2221	17	
P23	195		P2B	302		P2C	301	
P3	143		P3-	147		P3-1C	163	
P3-1M	162		P3-C1	165		P3-M1	164	
P31	144		P3112	151		P312	149	
P3121	152		P31C	159		P31M	157	
P32	145		P321	150		P3212	153	
P3221	154		P3C1	158		P3M1	156	
P4	75		P4-	81		P4-21C	114	
P4-3M	215		P4-3N	218		P4-B2	117	
P4-C2	116		P4-M2	115		P4-N2	118	
P4/M	83		P4/MBM	127		P4/MCC	124	
P4/MMM	123		P4/MNC	128		P4/N	85	W
P4/NBM	125	W	P4/NBMS	12501		P4/NBMZ	12502	
P4/NCC	130	W	P4/NCCS	13001		P4/NCCZ	13002	
P4/NMM	129	W	P4/NMMS	12901		P4/NMMZ	12902	
P4/NNC	126	W	P4/NNCS	12601		P4/NNCZ	12602	
P4/NS	8501		P4/NZ	8502		P41	76	
P41212	92		P4122	91		P4132	213	
P42	77		P42/M	84		P42/MBC	135	
P42/MCM	132		P42/MMC	131		P42/MNM	136	
P42/N	86	W	P42/NBC	133	W	P42/NBCS	13301	
P42/NBCZ	13302		P42/NCM	138	W	P42/NCMS	13801	
P42/NCMZ	13802		P42/NMC	137	W	P42/NMCS	13701	
P42/NMCZ	13702		P42/NNM	134	W	P42/NNMS	13401	
P42/NNMZ	13402		P42/NS	8601		P42/NZ	8602	
P4212	90		P422	89		P42212	94	
P4222	93		P4232	208		P42BC	106	
P42CM	101		P42MC	105		P42NM	102	
P43	78		P432	207		P43212	96	
P4322	95		P4332	212		P4BM	100	
P4CC	103		P4MM	99		P4NC	104	
P6	168		P6-	174		P6-2C	190	
P6-2M	189		P6-C2	188		P6-M2	187	
P6/M	175		P6/MCC	192		P6/MMM	191	
P61	169		P6122	178		P62	171	
P622	177		P6222	180		P63	173	
P63/M	176		P63/MCM	193		P63/MMC	194	
P6322	182		P63CM	185		P63MC	186	
P64	172		P6422	181		P65	170	
P6522	179		P6CC	184		P6MM	183	
PA3	205		PA3-	205		PB	701	
PBA2	32		PBAM	55		PBANS	5001	
PBANZ	5002		PBCA	6101		PBCM	57	
PBCN	60		PBNM	6203		PC	702	
PCA21	29		PCAB	6102		PCC2	27	
PCCA	54		PCCM	49		PCCN	56	
PCMN	6205		PM	6	W	PM3	200	
PM3-	200		PM3-M	221		PM3-N	223	
PM3M	221		PM3N	223		PMA2	28	
PMB	602		PMC	601		PMC21	26	
PMCN	6206		PMM2	25		PMMA	51	

```
PMMM       47    PMMN       59 W  PMMNS    5901
PMMNZ    5902    PMN21    3101    PMNA       53
PMNB     6204    PN21A    3302    PN3-      201 W
PN3-M     224 W  PN3-MS  22401    PN3-MZ  22402
PN3-N     222 W  PN3-NS  22201    PN3-NZ  22202
PN3-S   20101    PN3-Z   20102    PN3M      224 W
PN3MS   22401    PN3MZ   22402    PN3N      222 W
PN3NS   22201    PN3NZ   22202    PNA21    3301
PNAM     6202    PNC2       30    PNMA     6201
PNN2       34    PNNA       52    PNNM       58
R3        146 W  R3-       148 W  R3-C      167 W
R3-CH   16702    R3-CHR  16703    R3-CR   16701
R3-H    14802    R3-HR   14803    R3-M      166 W
R3-MH   16602    R3-MHR  16603    R3-MR   16601
R3-R    14801    R32       155 W  R32H    15502
R32HR   15503    R32R    15501    R3C       161 W
R3CH    16102    R3CHR   16103    R3CR    16101
R3H     14602    R3HR    14603    R3M       160 W
R3MH    16002    R3MHR   16003    R3MR    16001
R3R     14601
```

Space group by number

RG n ;

This command works like the command 'HMS'. Here, however, the space group number n as given in the International Tables 3rd edition is specified. If more than one setting is possible, the first choice is selected by adding 01, the second by adding 02 etc. Note that the first and the second setting of the monoclinic space groups in the IT 4th edition has been interchanged compared to the 3rd edition.

Extra Hermann Mauguin symbol

EHMS *symbol* ;

There are many space groups where several unconventional settings are in use. These are tabulated in the International Tables (Table 4.3.1, p. 56ff). The symbols given there are available when using the command "EHMS". In that case the symmetries of the corresponding space group are generated, and the transformation needed to obtain the given non-standard setting is applied. Sometimes there are two choices of origin available. One adds the letter 'z' to the space group symbol if the center of inversion lies at the origin, and the letter 's' otherwise, respectively.

Example: `EHMS Am2m` gives

```
Space group:  38 Action:  TRSY 1.  0.  0.  0.  0.  -1.  0.  1.  0.
```

HMS-Option

HMSO n ;

As mentioned above (see 'HMS') the short symbols of some space groups are not unique (Pm, C2 etc). Furthermore there are space groups given in two settings. If one wants to connect a certain setting to a short symbol anyway, a number n (usually 2) may be given to specify this setting. *Example:* If `HMSO 2` was entered, typing `HMS Pm` loads space group P1m1 (=602, Int. Tab., 3rd edition, 1969).

Print symmetries

OS ;

The symmetries currently in the symmetry list are printed in matrix form.

Print symmetries symbolically

OSY n_1, n_2, opt ;

The symmetries in the range from n_1 [1] to n_2 [last symmetry on the list] are printed in the format associated with the command 'SY'. If one enters a number unequal zero for *opt* [0], following each line containing the symmetry a line is printed where the symmetry is given in the notation used in 'ST'.

Further commands for symmetries

Symmetry by type

ST $type, n, ia_1, ia_2, ia_3, p_1, p_2, p_3, g_1, g_2, g_3$;

A symmetry also may be given by its type. *type* is one of the following numbers:

-6	sixfold roto-inversion axis
-4	fourfold roto-inversion axis
-3	threefold roto-inversion axis
-2	mirror plane
-1	center of symmetry
1	identity
2	twofold rotation axis
3	threefold rotation axis
4	fourfold rotation axis
6	sixfold rotation axis

n is a number, specifying how many times the rotation operation is to be executed (default: 1); ia_1, ia_2, ia_3: three integer numbers which describe the direction of the axis, or, in case of a mirror plane, the normal to the plane (default: 0,0,0)
p_1, p_2, p_3: three real numbers describing a point in triclinic coordinates lying on the symmetry element (default: 0,0,0)
$g1, g2, g3$: three real numbers describing a glide vector in triclinic coordinates (default: 0,0,0)

Examples:

```
ST -1             (Centre of symmetry in 0,0,0)
ST 3,,0 0 1       (Threefold axis with direction z. Caution!
                   The angle gamma must be 120 degrees.)
ST 3,,1 1 1       (Threefold axis with direction 1 1 1)
```

The result is stored and can be referred to by other commands, in particular by 'TSYI'. If the option 'STA' is active, the result is also stored in the symmetry list (initially not active).

Add ST-symmetry

STA n ;

The symmetry given by the command 'ST' is stored in the symmetry table, if n has a value unequal zero (initial setting 0). This value can be changed by the command 'STA'.

Group test

GRTS *opt* ;

It is checked, whether the introduced symmetries form a group. In addition, one checks whether the identity (i.e. the symmetry x, y, z) is in the first place of the symmetry list (this is required at several places in the program), and whether each symmetry appears only once. If necessary, error messages will be printed. Symmetries which are loaded by the commands 'HMS' or 'RG' are checked beforehand and do not need to be tested again. If *opt* [0] is set unequal zero, the symmetries present are treated as generators, and the full group is generated.

Remove double symmetries

EDS ;

After transformations leading to a smaller unit cell, it may happen that certain symmetries occur more than once in the symmetry list. 'EDS' will remove them.

Set end of the list of symmetries

SE m ;

One can set the number m of symmetries arbitrarily. The purpose is to make corrections in the list. How to perform such a task is described in detail at 'AE'.

Remove symmetries

ES n_1, n_2 ;

All symmetries stored in the symmetry list from n_1 (no default) to n_2 [n_1] are removed.

Select symmetries

SEL $s_1, \ldots, s_n$;

As a complement to the command 'ES', one can use the command 'SEL' to select those symmetries (up to 48) in the symmetry list, which are to be kept. All non-trivial symmetries except the ones in the symmetry list at the positions s_1 to s_n (no defaults) are eliminated. This command is useful in the analysis of group-subgroup relations.

Symmetries to file for RGS

SRGS *file* ;

The symmetries which are stored in the symmetry list will be written to the *file* [Default: `FOR012.DAT`]. The command is useful when studying supergroup / subgroup transformations.

Atom parameters

Atom parameters

ATOM s, e, x, y, z, r, p ;

With this command the atom parameters for an atom can be defined. The symbol s usually is an element name, but may be an arbitrary string of up to four characters. To distinguish atoms having the same name, a second string (extension) e is specified which may also have up to four characters. x, y, z refer to the unit cell (in the triclinic coordinate system), r is the radius assigned to the atom given in Å. p is a color pointer, i.e. the number of a control block where details of the graphical representation of the atom are stored (see 'ATF').

Atoms are numbered in the order they are entered and stored in the parameter list. The first digits of a code point to this position. The default values for x, y, z, r, p are the parameters entered last. The current version of the program can store up to 9999 atoms.

Every point (location in space) which one wants to address has to be entered as an "atom". It is not forbidden that several such generalized atoms have the same coordinates. For many purposes it is helpful to define two "pseudo atoms" (ORGN,NULL) having the coordinates (0.5,0.5,0.5) and (0,0,0), the center and the corner(s) of the unit cell, respectively. The radii of these "atoms" are meaningless of course. These two atoms are already "prepared" and can be placed in the parameter list by entering '`AE 2`'. Thus, the first "real" atom is usually number 3 in the parameter list.

Atom parameters with Cartesian coordinates

AKRT s, u, x, y, z, r, f ;

This command corresponds exactly to the command 'ATOM', except that (x, y, z) are Cartesian coordinates which will be transformed to triclinic coordinates when read in. See also 'PUKC' p. 46.

Print atom parameters

OA n_1, n_2 ;

The parameters as stored in the parameter list are printed. The output may be restricted to a certain range $n_1, \ldots, n_2$. If only n_1 is given, only one atom is shown. Default values are $n_1 = 3$, n_2 = last atom in the list.

Print atom parameters including the Wyckoff letter

OAWY n_1, n_2, opt ;

The parameters as stored in the parameter list are printed to screen. The output may be restricted to a certain range $n_1, \ldots, n_2$ in the parameter list. Default values are $n_1 = 3$, n_2 = last atom in the list. In addition to the data printed with the command 'OA', the multiplicity and the Wyckoff letter are printed, followed by the symbolic coordinates of that site. Only the first entry of the Int. Tables for the Wyckoff letter will be given, e.g. `x,2x,1/4`. Thus, these coordinates may *not* agree with the actual ones of the atom in the list (although they are related by a symmetry operation, of course). Note that for the space group Pmmm (no. 47), the letter α will be given as `A`. If $opt = 1$ (Default: 0) is entered, the coordinate parameters of the atoms are replaced by those parameters corresponding to the Wyckoff coordinates. If $opt = -1$ is specified (or $n_1 = -1$), all Wyckoff sites of the current space group are listed. For each, the following information is provided: (i) the multiplicity, (ii) the Wyckoff letter, (iii) the type: 1: `x,y,z`; 2: `x,x,z`; 3: `x,-x,z`; 4: `x,x,x`; 5: `x,y,y`; 6: `x,y,-y`; 7: `x,2x,z`; (iv) a number which is to be interpreted as a binary and defines the

fixed coordinates, and (v) coordinates or constants to be added.
Example: `24 d 1 3 0 0 .25` translates to 24 d $x, 0, 1/4$. Here, type 1 indicates that the coordinates x, y and z are independent. Next, the number 3 has to be read in binary as 011, indicating that the second and third coordinate are kept fixed. Finally, 0 0 .25 means that the first two coordinates have no offset (x+0=x and 0+0=0), while the third coordinate is shifted by 1/4 (0+0.25=0.25).

Print atoms overview

U ;

Often one only wants to know the positions of the atoms in the parameter list, but not the coordinates etc. The atoms stored are listed as follows:

```
    Symbol from -to (from -to) ...
...
```

Each of the symbols appearing in the parameter list is printed from where to where it is placed in the parameter list. This command helps to construct 'VB' and 'PK' commands.

Set starting point for parameters

SPST n ;

Many commands are using default values where it is assumed that the first two atoms of the parameter list are ORGN and NULL, respectively, i.e. the default value for the starting point for "real" atoms is 3. With this command this value may be changed to n.

Atom color control block

ATF $nr, f_r, s_r, f_m, n_m, f_s, s_s, n_s, n_l, f_i, s_i, n_1, n_2, n_3, n_4, a_0, a_1, z$;

For each atom in the parameter list a parameter p is included which represents the number of a control block containing parameters which define how atoms (and bonds) are to be drawn. Up to 20 control blocks can be defined. nr is the number of the control block. If an invalid number is entered in the atom parameters, the first ATF block ($nr = 1$) is used. The remaining parameters are: f_r: color of the outline (see table below) - initial setting: 15; s_r: thickness of the line (in pixels $1, 3, \ldots, 9$) - initial setting: 3; f_m: color of filling pattern - initial setting 15; n_m: number of filling pattern (see also 'FLM') - initial setting: 0; if one enters -1 for n_m, atoms will be transparent; f_s: color of label - initial setting: 15; s_s: thickness of line for labeling - initial setting 3; n_s and n_l have to be set, if a circle is drawn by dashed lines: n_s: number of points which are to be connected by a line - initial setting 1000; n_l: number of points not to be connected - initial setting: 0. f_r and f_m must be different, if this option is used.

Starting with version 7 the ATF blocks have been increased by parameters which allow formatting ORTEP ellipsoids and bonds. f_i and s_i are color and thickness of the lines inside the ellipsoids (initial setting: 15, 3). The numbers that follow control the type of ellipsoid representation. They are explained at the command 'EPAR'. The numbers for the colors are different for different machines:

No.	Atari-TT	PC
0	white	black
1	black	blue
2	red	green
3	green	cyan
4	blue	red
5	cyan	magenta
6	yellow	brown
7	magenta	grey
8	oliv-green	darkgrey
9	darkbrown	lightblue
10	brown	lightgreen
11	green	lightcyan
12	blue	lightred
13	lightblue	lightmagenta
14	yellow	yellow
15	violett	white
16	see text below	

On the color printer HP Desk Jet 550C only the colors are implemented. The brightness can only be influenced by the filling pattern. If one chooses the color number 7 (grey) on a PC, the desk jet uses the cartridge black instead of mixing fundamental colors (color number 0). A special meaning has the number 16 in those versions which create black/white pixel graphics: All colors (0-15) are given in black, 16 is given in white. Thus it is possible to label black spheres with white letters. Furthermore one may give a bond a white border and a black filling pattern.

A special meaning has the parameter z. It gives the angle (in degrees) of the first vertex of the polygon. KPLOT uses the following formula to produce circles: $x = r\cos(t+z)$, $y = r\sin(t+z)$. If e.g. with 'PK' four vertices are given, the square would stand on the corner if $z = 0$ (default). If the square should stand on one edge one has to set $z = 45$.

Print atom color control blocks

OATF n_1, n_2 ;

Atom color control blocks are printed in the given range n_1 to n_2. Default values are: $n_1 = 1$ and $n_2 = 20$.

Color atom

CA *sybl*, n, bnr, n_1, n_2 ;

To the atoms having the symbol *sybl* the color number n is assigned als follows: In the ATF blocks with the number *bnr* the first positions are redefined: $|n|$, 3, $|n|$, 1, 3, 15 if $n > 0$, and 0 otherwise. To the atoms with the symbol *sybl* in the range from n_1 to n_2 in the parameter list the number *bnr* is given as color pointer. If no valid (1-20) number is given for *bnr* the program seaches for an ATF block not used. Defaults: *sybl* none; n none; *bnr* 0; n_1 3; n_2 last atom. *Example:* `CA C 4` (makes C atoms red (on pc's)).

Color pointer for numbers

FRBN p, n_1, n_2 ;

This command may be used to set the parameter p (color pointer) of the atoms in the parameter list even at a later stage. The color pointers of the atoms having numbers $n_1, \ldots, n_2$ are set to p. Default settings: $p = 1$, $n_1 = 3$, n_2 =last atom.

Color pointer for symbols

FRBS p, s, n_1, n_2 ;

This command may be used to set the parameter p (color pointer) of the atom parameters even at a later stage. To all atoms having the symbol s, the number p (no default) is assigned if the atoms have numbers $n_1, \ldots, n_2$. Here the defaults are: $n_1 = 3$ and $n_2 =$ last atom.

Temperature factor

TF $s, e,\ t_1, \ldots, t_6, type$;

In order to pass temperature factors to the program ORTEP or to work with them in ORTEP mode, the command 'TF' allows to define the temperature factors. The parameters $t_1, \ldots, t_6, type$ have the same meaning as the parameters on the subsequent lines following the coordinates of the program ORTEP:

	Type 0-3	Type 4-5,8-9	Type 6	Type 7
t_1	b_{11}	U_{11}	*b b*	*r r*
t_2	b_{22}	U_{22}	0 0	0 0
t_3	b_{33}	U_{33}	0) vdc-	0) vdc-
t_4	b_{12}	U_{12}	0) (1)	0) (1)
t_5	b_{13}	U_{13}	0) vdc-	0) vdc-
t_6	b_{23}	U_{23}	0) (2)	0) (2)
type	0,1,2,3	4,5,8,9	6	7

The coefficients b_{ij} $(i, j = 1, 2, 3)$ of the anisotropic temperature factor (typ 0-3) are defined as follows:

$$Base^{(-(b_{11}h^2+b_{22}k^2+b_{33}l^2+cb_{12}hk+cb_{13}hl+cb_{23}kl))},$$

where *Base* and c have the following values:

Typ 0: Base = e, c = 2,
Typ 1: Base = e, c = 1,
Typ 2: Base = 2, c = 2,
Typ 3: Base = 2, c = 1.

(e = 2.71828..., Euler number)

The coefficients U_{ij} of the anisotropic temperature factors of type 4-5 and 8-9 are given by:

$$\exp(-d(\mathbf{a}_1^{*2}U_{11}h^2 + \mathbf{a}_2^{*2}U_{22}k^2 + \mathbf{a}_3^{*2}U_{33}l^2 + c\mathbf{a}_1^*\mathbf{a}_2^*U_{12}hk + c\mathbf{a}_1^*\mathbf{a}_3^*U_{13}hl + c\mathbf{a}_2^*\mathbf{a}_3^*U_{23}kl))$$

Here $\mathbf{a}_1^*$, $\mathbf{a}_2^*$ and $\mathbf{a}_3^*$ denote the reciprocal cell constants. For types 4 and 8, $c = 2$ holds, for types 5 and 9, $d = 1/4$, and for types 8 and 9, $d = 2\pi^2$.

Type 6 allows the input of the Debye-Waller factor B (isotropic temperature factor), defined as follows:

$$\exp(-B\sin^2(\theta)/\lambda^2),$$

where λ is the wavelength and θ the Bragg angle. The parameter B is related to the mean-square displacement $\overline{\mu^2}$ of the atom from its average position by the relation

$$B = 8\pi^2\overline{\mu^2}.$$

When the isotropic temperature factor is used, the atom is represented as an isotropic ellipsoid (sphere) with equal principal axes of length $\overline{\mu}$. If no input is given for $t_3, \ldots, t_6$, the directions of the principal axes are along the Cartesian system axes. However, one can re-orient these arbitrary orthogonal vectors by using the two vector designator codes (vdc) u (= vdc(1)) and v (= vdc(2)); then the new principal-axes vectors will be u , $u \times v$, and $u \times (u \times v)$. This is strictly an esthetic feature of no physical significance.

Type 7 allows the input of arbitrary spheres of radius r in Å. The vector triplet orientation is specified as with type 6.

The input for the name s and the extension e refer to an atom stored in the parameter list, e.g. an atom must be present having the same name. If more than one atom is present having this name and extension, all atoms are assigned the same temperature factor. Note, that the cell has to be given before temperature factors can be assigned.

Temperature factor for last atom

TL $t_1, \ldots, t_6, type$;

The command 'TL' is nearly identical with 'TF'. The difference is that the parameters refer to the last atom in the parameter list. This allows to assign individual temperature factors to atoms having the same name, an operation impossible when using 'TF'. Such atoms may be generated when using 'MTRI'.

Mode of output of temperature factors

TFL n ;

This option controls how to write temperature factors to file by 'PUT' or 'PUTC': If $n = 0$ (default) is specified, all temperature factors are written as a separate block in form of 'TF' commands after the parameters. A value unequal zero switches to 'TL' form, e.g. each atom is followed by its temperature factors. This option must be used, if the atoms are not distinguishable by their names alone.

Delete temperature factor(s)

LTF s, e ;

The temperature factor of a single atom, with the name s and extension e, or temperature factors of a group of atoms may be deleted by 'LTF', if used without extension. E.g. the command `LTF H` removes the temperature factors of all H-atoms. A special meaning has the "symbol" s =ALLE: all temperature factors will be deleted.

Print temperature factors

OTF n_1, n_2 ;

The anisotropic temperature factors of the atoms in the parameter list between n_1 (default: 3) and n_2 (default: last atom) will be listed. Note that there may be gaps within the sequence, if not all atoms have been assigned such factors.

Additional commands for atoms

Edit atom parameters

ATED $nr,\ s, e, x, y, z, r, p$;

In order to alter certain parameters of an atom, 'ATED' may be used. nr is the position of the atom in the parameter list. The input that follows is the same as with the command 'ATOM', but here all values stored at nr are used as defaults. Example: The y-value of the atom stored at the 5th position in the parameter list should be set to 0.5.
Input: `ATED 5 *** 0.5`

Normalize parameters according to cell

NPZ n_1, n_2 ;

The coordinates stored in the parameter list in the given interval $n_1, \ldots, n_2$ are normalized so that $0 \leq x, y, z < 1$. Defaults: $n_1 = 3$, $n_2 =$ last atom.

Parameter check

PCK resp. NPCK ;

If option 'PCK' is active (initial setting) during the input, it is checked whether the absolute values of the triclinic coordinates exceed the value 3 and the parameters are rejected if so. 'NPCK' turns off this checking. This may be necessary, if Cartesian coordinates are used.

Atom parameters

AT x, y, z, r, s, e, p ;

This command works like 'ATOM'. Here the parameters are entered in a different order.

Define a grid

GT na, nb, nc ;

In the literature the coordinates of atoms are sometimes not given in the usual way. E.g., instead of 0.0123 as 123 with the instruction to divide each number by 10000. In such a case the definition of an appropriate grid will help. na, nb, nc [100,100,100] give the appropriate divisor in the a, b, and c direction, respectively. 'GT' is used in connection with 'OCG' and 'ATG' to print or to input atoms.

Atom parameters in grid units

ATG nx, ny, nz, r, s, e, p ;

This command works in the same way as 'AT', but the coordinates nx, ny, nz (floating point numbers are allowed) refer to a grid which is defined by the command 'GT'. Defaults: $e =$ blank, $p = 0$.

Radius

R $r, n_1[, n_2[, f]]$;

The radii of the atoms may be redefined or changed to the value r by this command (no default). n_1, n_2 specify the range in the parameter list. If only n_1 (no default) is given, only one radius is changed. If $f > 0$ (default: 0), the new radii are calculated according the formula $r_{new} = fr_{old} + r$.

Radii for atoms by symbol

RS $r, s[, n_1[, n_2[, f]]]$;

The radii of the atoms having the name s (no default) may be redefined or changed to the value r by this command (no default). $n_1, \ldots, n_2$ is the range in the parameter list (defaults: 1, last atom in the parameter list). If $f > 0$, the new radii are calculated according the formula $r_{new} = fr_{old} + r$.

Radius by color pointer

RFRB $r, f_1[, f_2[, n_1[, n_2]]]$;

The radii of the atoms having a color pointer in the range from f_1 (default: 0) to f_2 (default: f_1) may be redefined or changed by this command to the value r (default: 0.3). $n_1, \ldots, n_2$ specify the range in the parameter list (defaults: 3, last atom in the list).

Symbol for atoms with color pointer

SFRB $s, e, f_1[, f_2[, n_1[, n_2]]]$;

A renaming of the atoms in the parameter list takes place. The name s (default: blank) and an *integer number* e (default: 1) which is incremented every time by one, are assigned to the atoms having the color pointers f_1 (default: 0) up to f_2 (default: f_1). This renaming may be restricted to parts of the parameter list, if n_1 and n_2 are specified (defaults: 3, last atom).

Print atoms separately

OAS $n_1, \ldots, n_m$;

Atoms stored in the parameter list may be printed separately according to the list specified. A sequence can be defined by preceding the second number by a minus sign.
Example: `OAS 3 -5 8 17` will print atoms 3, 4, 5, 8 and 17.

Always U resp. not always U

IU resp. NIU ;

If the option 'IU' is in effect, after executing 'EPU' or 'STPU' the command 'U' is executed automatically. 'NIU' (initial setting) turns off this feature.

Add codes to parameter list

ACAL n_1, n_2 ;

Codes which are stored in the code list from position n_1 [1] to position n_2 [last code] in the code list are added as atoms in the parameter list. This command may be useful when generating molecules from structure fragments scattered throughout the unit cell.

Add foreign codes to parameter list

ACAF n_1, n_2 ;

Codes which are stored in the code list *of the background structure* from position n_1 [1] to position n_2 [last code] in the code list are added as atoms in the parameter list. This command may be useful if bonds are to be shown between both structures.

Interchange parameters or atoms

XP resp. XAT n_1, n_2 ;

Using the command XAT, the atoms at the positions n_1 and n_2 in the parameter list are interchanged. When using 'XP', only the parameters and r are interchanged.

Move parameter

MVP $n_1[, n_2], n_3$;

The atoms in the parameter list from n_1 to n_2 are moved in such a way, that this group is placed before the atom at position n_3 in the list. If n_2 is omitted, only atom no. n_1 is moved. Note that the size of the parameter list is not changed.

Add multiple codes

AMC $c_1, c_2, \ldots, c_n$;

The triclinic coordinates of the atoms with the codes $c_1, \ldots, c_n$ are calculated and added to the parameter (!) list. The remaining parameters (name etc.) are inherited from the original atoms. Up to 72 atoms can be added in one step to the parameter list.

Name of an atom

N s, e, n ;

An atom located at position n in the parameter list may be assignd a (new) name s and a (new) extension e to distinguish atoms having the same name. If the command is entered a second time and no new value for n is specified, then the previously entered value of n is incremented by one. The default for s is the last name given; the default string for e is blank.

Set end of parameter list

AE n ;

With this command one can set the variable containing the number of atoms stored in the parameter list. One purpose is to correct possibly invalid parameters. *Example:* If seven atoms are stored in the parameter list and the fourth atom has to be corrected, the following sequence will do this job:

`AE 3 ; AT` x, y, z, r `; AE 7`

Generate temperature factor

GTF $s, e,\ d_1, d_2, d_3$;

The directions of the axes of the free coordinate system with half diameters in the principal directions of the ellipsoids d_1, d_2, and d_3 are used to define the principal axes of an ellipsoid and assign it to the atom(s) with the the name (s, e) given.

Test for reflection

TR h, k, l ;

For the Miller indices h, k, l all symmetry-equivalent reflections (excluding Friedel reflections) are calculated and printed, together with the face multiplicity.

Remove canceled reflections

EAR *n, type* ;

A very special application of KPLOT is the study of reflection patterns, especially in case of twinning, i.e. we have to deal with the reciprocal space. Atoms are used to represent reflections, and no symmetries may be present containing translation parts, i.e. the symmetries are valid for the reciprocal space. The reflections are generated from one atom having the coordinates (0,0,0), and are stored in the code list. If centering is present, certain reflections are systematically absent. These canceled reflections may be removed using 'EAR'. n is the number of the atom in the parameter list used to generate the reflections, and *type* one of the following letters: P, I, R, F, A, B, or C, indicating the type of centering in the structure.

Tables for distances and angles

Distance and angle table

ATAB resp. WTAB $c_1, c_2, n_1, n_2, dmax[, dmin[, w_1[, w_2]]]$;

These commands will generate tables containing distances or distances and angles, respectively. All atoms having codes between c_1 [no default] and c_2 [c_1] are taken as centers of search spheres. (Usually, one enters the codes in abbreviated form by using the parameter number only.) Within such a sphere with the radius $dmax$ [3.0], one searches for target atoms (codes) having numbers in the parameter list between n_1 and n_2 [no defaults]. Atoms having a distance less than $dmin$ [0.0001] are excluded. If c_2 is given as atom number the rest of the code of c_1 is added. To decrease the amount of angles being printed an additional condition may be given: Only angles between w_1 [0] and w_2 [180] will be printed. A special meaning has a value unequal zero for w_1 if used with 'ATAB': Distances to atoms having codes in the code list are excluded from the calculation. This is useful for the calculation of intermolecular distances.

Distances using symbols

AS $s_1, s_2, [[, dmin], dmax]$;

'AS' is an easy to use but less sophisticated command to calculate distances. Source atoms are taken from the parameter list if they have the symbol(s) s_1, which may also be a list enclosed in quotes. Target atoms are those having the symbol(s) s_2. If $dmin$ is omitted, the value 0.0001 is used. If $dmax$ is omitted too, a value is taken from the table which is also used with 'FM'. *Examples:* `AS Na Cl 3; AS Ta 'Br O' 2.7` . There may be up to 20 symbols present.

Mean distance in molecule

MDM resp. MDMR $u_1, u_2, z_1, z_2, dmax[, dmin]$;

All pair-wise distances are calculated between atoms having numbers between u_1 and u_2 [no defaults] and atoms having numbers between z_1 and z_2 [no defaults] in the given range $dmax, dmin$ [Defaults: none, 0] for all those atoms that are found in the code list. The mean and the standard deviation are calculated and printed. Using the command 'MDMR', each distance is, in addition, automatically reduced by the radius of the atom at the endpoint.

Composition of the drawing

The composition of a drawing is a compilation of those atoms which are to be plotted and connected. These atoms are stored in the code list which can store up to 9999 codes.

Multiple occupation allowed

AGLP ;

Usually it is impossible that in crystal structures a position, or two very close points, is occupied by more than one atom. On the other hand there exist cases where the occupation of one point by more than one atom would be forced by symmetry. These positions are called "special positions". In the program this situation is recognized and the occupation by only one atom of a certain kind (i.e. atom number) allowed. If 'AGLP' is given this check is turned off. In that case, the options 'GLP' and 'NGLP' are no longer in effect.

Multiple occupation excluded

GLP resp. NGLP *tol* ;

If the option 'GLP' is in effect (default), the occupation of a point, or two very close points, by more than one atom with the same atom number is prohibited, i.e. special positions are occupied only once. Two points are treated as being too close, if their distance in each (triclinc) coordinate is less than *tol* (default: 0.01). *tol* may be redefined with the commands 'GLP' and 'NGLP'.

Option 'NGLP' is more restrictive than 'GLP'. If 'NGLP' is active, it is not allowed to occupy a position by more than one atom at all. 'GLP' and 'NGLP' do not affect 'ACI' or 'ACIM'.

Output while adding to code list

AO resp. NAO ;

If the option 'AO' is set, the program will print all additions to the code list while executing one of the commands 'ATB', 'AKS', ... (except 'ACI'). It will print: (1) the position in the list, (2) the code itself, (3) the triclinic coordinates, and (4) the distance from the origin, if the command was 'AKS', 'AU', or 'AUW'. When using the other commands, this last number is meaningless. The option 'NAO' (initial setting) suppresses the output.

Clear codes

CC ;

The code list will be cleared (equivalent to CE 0).

Add codes immediately

ACI $c_1[, c_2]$;

All codes will be generated and added to the code list which lie beween c_1 and c_2. If only one code is specified, only this code will be added. *Example:* `ACI 2 266601` will generate 8 codes: 255501, 255601, 256501, ..., 266601 describing the corners of the cell.

Add codes for cell outlines

ACIZ c_1, c_2 ;

All codes will be generated and *a subset* added to the code list which lie between c_1 and c_2. If only one code is specified, only this code will be added. The subset which will be stored meets the following conditions: If the sum of the digits of the translational part is odd (i.e. 555) the atom number of the first code is selected; otherwise the atom number of the second code. This command may be used to store a special subset in the code list (i.e. of a hexagonal cell) in order to avoid diagonal bonds. *Example:* The atoms having numbers 2 and 5 may both have the coordinates $(0, 0, 0)$. The command `ACIZ 2 566601` plus a 'VB' command joining atoms having number 2 with atoms having number 5 will do this job.

Add codes immediately multiply

ACIM $c_1, c_2, \ldots, c_n$;

This command works in principle like 'ACI', but is more sophisticated. Up to 72 codes may be entered. Runs (intervals of codes) are recognized by a minus sign of the second code. Single codes and runs may be intermixed. The example above using 'ACIM' would be:

```
ACIM 2 -266601 .
```

Add codes by translation

ACT n_1, n_2, t_a, t_b, t_c ;

All codes that are in the code list and have numbers between n_1 and n_2 are shifted by the vector (t_a, t_b, t_c) in units of lattice constants (the t_i are integers!) and added to the code list (if not already present). In this way layers can be easily generated. No defaults are supplied, but the values entered are kept and used as defaults next time.

Subtract codes immediately

SCI $c_1[, c_2]$;

From the code list, all codes from c_1 [no default] to c_2 [c_1] are removed. If only one code is specified only this code will be removed.

Delete code(s) according to number(s)

DCN $n_1[, n_2]$;

Codes having numbers that point to atoms in the parameter list from n_1 [no default] to n_2 [n_1] will be removed from the list. In this way one may easily remove certain types of atoms.

Delete code(s) with symbol

DCS *symb* $[, n_1[, n_2]]$;

Codes of atoms having the name *symb* will be removed from the code list. In addition the selection may be restricted to a certain part of the parameter list: only atoms from n_1 [1] to n_2 [last atom] are those which may be removed.

Remove a range from the code list

EN $n_1[, n_2]$;

The codes from n_1 [no default] to n_2 [n_1] will be removed from the code list. But be careful! A lot of commands reorder this list. Thus it is advisable to first check the current list with 'OC'.

Add triclinic box

ATB $or, t_1, t_2, d_x, d_y, d_z$;

Around the point given via the code *or* (center of the box, default 155501), a box (a parallelepiped) is defined with the side vectors parallel to the lattice axes. The box has semidimensions d_x, d_y, and d_z referring to the lattice constants (default 0.5,0.5,0.5). For all atoms having numbers between t_1 [3] and t_2 [last atom] the codes that lie in the box will be added to the code list if not already present. In this way atoms on special positions will be added only once.

Example:
The atom no. 1 may have the coordinates (0.5,0.5,0.5). It is supposed that atoms with the numbers 7, 8, 9, and 10 of one unit cell are added to the code list. The command doing this would be:

```
ATB 1,7,10,.5,.5,.5 ;
```

Note that atoms on cell bounderies may occur several times. Defaults are: $or = 1$, $t_1 = 3$, $t_2 =$ last atom, $d_x = d_y = d_z = 0.5$. So, if one wants to fill a cell, just enter ATB;. A special meaning has the input of zero for *or* (c.f. 'AKS').

Add Cartesian box

AKB t_1, t_2, d_x, d_y, d_z ;

Around the origin of the free coordinate system (center of the box) a Cartesian box is defined with semidimensions d_x, d_y, and d_z (no defaults) in Å. All codes of atoms having numbers between t_1 and t_2 in the parameter list (no defaults), which lie in the box, will be added to the code list if not already present.

Add sphere

AKS $or, t_1, t_2, dmax[, dmin]$;

Around the point given via the code *or* (center of the sphere, no default), a sphere is defined with the radius *dmax* (no default). For all atoms having numbers between t_1 and t_2 (no defaults) the codes that lie in the sphere will be added to the code list if not already present. In this way atoms on special positions will be added only once. In addition the sphere with radius *dmin* can be excluded (default: 0).

A special meaning has $or = 0$. In this case all codes from the group list of codes (mouse list, see 'LGC') are taken as origins. In this way atoms may be added to the code list selectively.

Add surroundings

AU $o_1, o_2, t_1, t_2, dmax, ccentr, dcentr, dmin$;

This command works similar to 'AKS'. But here atoms that are already present in the code list which have numbers in the parameter list between o_1 and o_2 (no defaults) are taken as the centres of the spheres. For the meaning of *ccentr*, *dcentr*, and *dmin*, c.f. 'AUW'.

Add surroundings to foreign atoms

AUF $ur_1, ur_2, zl_1, zl_2, d$;

Defaults: $ur_1 = zl_1 = 3$, ur_2 and zl_2 = number of last atom in both structures, respectively, $d = 1$. This command identifies those atoms in the current structure that have atom numbers between zl_1 and zl_2 and which are located within a distance $\leq d$ around those atoms in the "'foreign"' (i.e. non-current) structure that have atom numbers between ur_1 and ur_2. The atoms found are added to the current code list, unless they are already in the list.

Add surroundings by symbols

AUS $s_{or}, s_{tr}, dmax[, nghbr[, n_1[, n_2]]]$;

This command corresponds to 'AU'. Here, those atoms which have the name s_{or} serve as origin of the spheres. Note that again only atoms are considered, which are already in the code list. The target atoms have the name s_{tr}. *dmax* [0] is the radius of the sphere. The default value 0 has a special meaning: if given (or an entry omitted) the value is taken from the table which is also used with 'FM'. A number for *nghbr* [1] may be specified defining the minimal number of neighbours which the new atom must have among the atoms in the original code list. In addition the search for target atoms may be restricted to the range from n_1 to n_2 of the parameter list (defaults: 3, last atom).

Subtract codes in certain surrounding

SU $o_1, o_2, t_1, t_2[, r[, r_2]]$;

The spheres of r Angstroms around atoms in the code list having numbers in the parameter list from o_1 to o_2 (origin atoms) are searched for target atoms with numbers in the parameter list from t_1 to t_2 (no defaults). If found, they will be removed from the code list with the exception that no atom removes itself. The command allows e.g. in the case of disordered structures to remove those atoms from the drawing which make no sense chemically. Furthermore, atoms may be removed that occupy the same positions as other ones. Default values are: $r = 0.01$, $r_2 = 0.0$. r_2 may be used to define an interval, but $r > r_2$, always.

Add surroundings with repetition

AUW $o_1, o_2, t_1, t_2, dmax, ccentr, dcentr, dmin$;

'AUW' works like 'AU'. Recall that when using 'AU', only those codes are used as origin points, which are in the code list before the command is executed. 'AUW' uses also the new codes found and stops if no new atoms are found (or the code list overflows). The command is useful for creating molecular structures: Only one atom is stored in the code list (e.g. using 'ACI'), and for o_1, o_2 and t_1, t_2 all possible atom numbers of the molecule (in the parameter list) are specified. 'AUW' will then generate the complete molecule, if the distance parameters are chosen appropriately.

If a code for *ccentr* is given, only those atoms will be added to the code list which have a distance from *ccentr* less than *dcentr* (default: largest lattice constant). This way the search process is stopped at some point, even if one deals with an infinite polymeric structure. Default for *ccentr* is zero, meaning that this additional test is not performed.

If a value for *dmin* [0] is specified, atoms are excluded if found within spheres of this radius around $o_1, \ldots, o_2$

Add surroundings in cycles

AUZ $o_1, o_2, t_1, t_2, dmax, ncycl$;

'AUZ' works like 'AU'. In addition a value for *ncycl* can be specified (default: 2). After each cycle o_1, o_2 is exchanged with t_1, t_2 and the search is repeated. Using this technique layers (binary networks) can easily be generated.

Remove group of codes from code list

DGCC ;

The code list is searched for codes which are on the list of group of codes (mouse list, see 'LGC'). If they

are found they will be removed from the code list. In this way certain unwanted atoms can be selectively removed from the drawing: 'M' - click on atoms - 'DGCC'.

Load code list from group list

LGCC ;

The code list is cleared and the codes from the group list (mouse list, see 'LGC') are added.

Generate cell outlines

GNZL n_1, n_2 ;

The codes and plot commands which are necessary to produce the outline of the cell are created automatically. The numbers n_1 and n_2 are the positions of atoms in the parameter list having the coordinates (0,0,0). Default for n_1 and n_2 is 2. If the diagonal across one face or the body diagonal is as long as a lattice constant (e.g. in the hexagonal system), a different strategy is chosen: instead of 'VB' a 'VD' command is created with atom n_1 (connecting the 8 codes n155501 -n166601).

Print codes

OC $[n_1[, n_2]][s_1, s_2, \ldots]$;

All codes stored in the code list will be printed with both their triclinic and their Cartesian coordinates (the latter with respect to the free coordinate system). The range may be restricted by specifying n_1 (default: 1) and n_2 (default: last code). Another restriction may be given by a list of symbols. Only codes that match one of the symbols will be printed, otherwise all symbols s_i are affected.

Automatic generation of a drawing

Find (organic) molecule

FM resp. FMA res, opt, n_1, n_2 ;

To obtain a quick overview when dealing with molecular structures, these commands provide a highly automated way to generate a drawing. All atoms having numbers in the parameter list from n_1 (default: 3) to n_2 (default: last atom) are included in a process as follows:

The code list is cleared when 'FM' is entered. This step is skipped when 'FMA' is used. Next one code is added to the code list. Then the surroundings of all atoms in the code list are used as origins and spheres are searched for neighbouring atoms. If no new atoms are found, the first residue is complete. Then it is checked whether all atoms from the given range appear at least once. If so, the process stops. If not, the next atom not yet appearing in the code list is added, and the search process is resumed. The newly constructed fragment is assigned the next residue number. If *res* was specified (default 0, i.e. all residues) only the residue with that number is kept.

The hydrogen atoms play a special role. They are never used as origin atoms in the search and completely excluded if *opt* is set unequal the letter 'H'. Default: blank. Plot commands will be generated and an attempt is made to optimize the view, if the search was successful. Thus, after 'FM', in most cases only the command 'EPU' or 'STPU' has to be entered.

In the case of a polymeric structure the building process stops if an atom is found which is already in the code list, with a different translational component. Usually one wants to clear the code list and the list for plot commands at the beginning, using 'FM'. If 'FMA' is used, these resettings will not be done.

Find cell content

FZ $d_x, d_y, d_z, or, t_1, t_2$;

A set of atoms is generated and stored in the code list analogously to 'ATB'. In addition, plot commands are generated and an optimisation of the view is performed. Default values are: $d_x = d_y = d_z = 0.5$, $or = 155501$, $t_1 = 3$, and $t_2 =$ last atom.

Set limit for codes (for FM)

FMMC n ;

Since the commands FM, FMA, and FZ use much computer power there exists a limit (initial setting: 150) for the number of the codes that are generated using these commands. If while searching for new atoms this limit is exceeded, the process is stopped. Nevertheless the search is treated, as if it had been successfully completed, and the atoms found so far may be plotted. This limit may be changed to the value n by 'FMMC'.

Distance pair (for FM)

FMDP at_1, at_2, dis ;

During the search for molecules a predefined internal table is used containing interatomic distances. Using 'FMDP', these values may be overwritten and new pairs added. If a certain pair of atoms is not found in the table, a default value (2.7) is used.
Example: `FMDP Na Cl 3`

Reset distance pair table

FMDR ;

In the table containing distances for the molecule search the newly introduced pairs will be removed, while the values changed are not affected.

Default distance for 'FM'

FMDD dis ;

During the search for molecules a predefined internal table is used containing interatomic distances. If a pair does not appear in the table, a default distance (2.7 Å) is used. This value may be changed to dis with this command.

Print distance table

OFMD ;

The current values for interatomic distances as used by 'FM' are printed.

FM options

FMOP $opt_1, opt_2, \ldots$;

Sometimes it is advisable to control the actions that are being performed when generating a drawing automatically by 'FM', 'FMA', and 'FZ'. The possible options are entered as a number n [31=16+8+4+2+1] or a sequence of letters from which this number is calculated.

16 Option Z:	Using 'FZ' the outlines of the unit cell will be generated (see 'GNZL'). 'NZ' cancels this option. Initial setting: Z.
8 Option S:	Before 'FM' etc. is executed, the parameter list may be reordered. This is advisable in order to avoid creating an excessive number of plot commands. The option NS cancels option S. Initial setting: S.
4 Option P:	Before the process is started, the table containing plot commands is cleared and new plot commands are generated. The option NP cancels option P, i.e. plot commands are not touched. Initial setting: P.
2 Option A:	View optimisation. After compiling the atoms in the code list, an attempt is made to optimize the view. This is done by calculating a "best plane" and some rotations to avoid overlap of atoms. The option NA cancels option A. Initial setting: A
1 Option F:	The scaling factor is calculated. The option NF cancels option F. Initial setting: F

Example: `FMOP NP` This will prohibit changing the plot commands. All other options currently active will not be changed.

Additional commands affecting codes

Print codes in grid coordinates

OCG n_1, n_2 ;

The command works in the same way as 'OC'. Instead of printing Cartesian coordinates of the codes in the code list from n_1 to n_2 (defaults: 1, last code), here grid coordinates are printed which refer to the grid defined with 'GT'. This command may be helpful when interpreting e.g. a Fourier map.

Print "Overview" over content of code list

UC ;

The atoms currently in the code list are counted, sorted according to their symbols, and printed. (The order of the entries in the code list itself is not changed in the process.)

Add content of a unit cell

AZIN n_1, n_2 ;

First, the code list is cleared (!) and then filled with the codes belonging to those atoms within the unit cell which have numbers in the parameter list from n_1 to n_2, i.e. for the triclinic coordinates of all atoms $0 \leq x, y, z < 1$. At the same time atoms having the same name are counted and the result appears on the screen. Default values are: $n_1 = 3$ and $n_2 =$ last atom of the parameter list.

Which triclinic box?

WTB n_1, n_2 ;

The smallest triclinic box is found in which the atoms can be included which are in the code list and have numbers between n_1 [1] and n_2 [last atom]. The semidimensions and the center will be printed.

Cut triclinic box

CUTT $or, t_1, t_2, d_x, d_y, d_z$;

In some sense this command is the opposite to the command 'ATB'. Around the point with the code *or* [155501] a parallelepiped is defined with side vectors parallel to the lattice axes. The box has semidimensions d_x, d_y, and d_z referring to the lattice constants (defaults: 3×1000). For all atoms having numbers between t_1 [3] and t_2 [last atom], the codes that lie outside the box will be removed.

Set number of codes

CE n ;

With this command one can set the variable containing the number of codes stored in the code list. This command is most often used to clear the code list by setting n to zero. The default value is set to the number of the last code, i.e. using the default does not change anything.

Orientation

Define a coordinate system

K or, x_1, x_2, y_1, y_2 ;

This command defines a coordinate system, the so-called free coordinate system. All axes are 1 Å long and pairwise orthogonal (orthonormal system). In addition a right-handed orientation is enforced. The Cartesian coordinates which refer to this coordinate system are sometimes called free coordinates. *or* is a code specifying the origin. $x_1 \rightarrow x_2$ is a vector u defining the direction of the x-axis. $y_1 \rightarrow y_2$ is a vector v; the y-axis is defined in such a way that it is perpendicular to the x-axis and lies in the same plane as u and v. The z-axis is then defined by the requirement of the coordinate system being right-handed and orthonormal. If x_1,...,y_2 are omitted, only the origin is changed. If y_1 and y_2 are omitted, the program determines a usable vector automatically. (One of the lattice vectors a, b, or c is chosen having the minimal absolute value of the dot product with u.) Defaults: none.

During the plotting stage, the atoms are projected onto the x-y plane of this coordinate system, while the view point (position of the observer) has the coordinates $(0, 0, z)$. On the screen (or paper) the x-axis points to the right and the y-axis to the top.

The definition of the coordinate system may be visualized as follows: A sheet of paper is placed inside the structure, representing the drawing plane. The point *or* defines the center of the paper. The vector $x_1 \rightarrow x_2$ points from the left to the right and the vector $y_1 \rightarrow y_2$ from bottom to top.

Rotate coordinate system

DK a, w ;

The free coordinate system will be rotated around the axis a. The axis has to be entered as 1, 2, or 3, instead of x, y, or z. The rotation is counter clockwise when viewed along the axis selected, and the value for angle is given as a positive number in degrees. As the x-y-plane of the free coordinate system is the drawing plane, the viewer gets the impression that the model is rotated clockwise. The default value for the angle is the last result from 'W' or 'AW', which is initially set to zero.

Rotate coordinate system around a vector

DKV p_1, p_2, w ;

The free coordinate system will be rotated around the vector $p_1 \rightarrow p_2$, specified by two codes p_1 and p_2. If p_1

and/or p_2 are entered as zeroes, the codes from the mouse list will be used (see 'LGC'). No default values are provided, but values once entered are kept and used as defaults next time. The rotation is counter-clockwise when viewed along the specified vector. The default value for the angle is the last result from 'W' or 'AW'. Recall that in 'W' the initial setting is zero.

Special rotation of the coordinate system

WK ;

The axes of the free coordinate system will be interchanged cyclically as follows:
- the x axis becomes the z axis,
- the y axis becomes the x axis,
- the z axis becomes the y axis.

Print Orientation

O ;

Three angles are calculated and printed, which are to be used to obtain the current orientation. The sequence is (1) x axis, (2) y axis, and (3) z axis.

Reset coordinate system to original setting

KS ;

The vectors of the basis are re-set to the values, to which they were set when the cell was defined:
- x-axis in direction of the a axis,
- y-axis in the a-b plane,
- z-axis perpendicular to x and y to form a right-handed system.

The origin is set to (0,0,0).

Origin of coordinate system from point register

KUP ;

The origin of the free coordinate system is defined by the point currently in the point register. The orientation is not changed.

Origin of coordinate system from highest point above the center of gravity

KUSP n_1, n_2 ;

The origin of the free coordinate system is determined as follows: The weighted center of gravity of all codes in the code list having numbers in the parameter list from n_1 to n_2 is computed; the radii are used as weights. The z-coordinate of this point is replaced by the z-coordinate of the point nearest to the viewer. This command is useful when generating stereo plots in a frame (see 'PR'), because when following the DIN rule the model appears to lie behind the frame.

Coordinate system via code

KOC c ;

The origin of the free coordinate system is determined as follows: The x- and y-coordinate are taken from the given code c (no default). The z-coordinate is derived from the atom nearest to the viewer.

The purpose is to generate stereoscopic drawings which follow the DIN rule according to which the model appears to lie behind the frame (c.f. 'PR' and 'KUSP'). On the other hand such a choice is unsuitable while optimizing the view. In this stage a better choice is a point in the middle of the structure. E.g. `K 1`, optimizes the view. Then before producing the final drawing one places the origin above this point, e.g. `KOC 1;BF;STPU`.

Save, restore, and exchange free coordinate system

SK resp. RK resp. XK ;

The basis vectors of the free coordinate system may be copied to memory in order to save them ('SK'). Using 'RK' they are restored. On the other hand, using 'XK' interchanges the current values and the stored values.

Input orientation matrix

IMAT $m_{11}, m_{12}, \ldots, m_{33}$;

The orientation matrix (which is usually calculated by the program automatically) may be entered directly (columnwise). No default values are provided. The program checks, whether the determinant of this matrix has the value 1 (± 0.0001). Else, the matrix will be rejected.

Print orientation matrix

OMAT ;

The current orientation matrix will be printed.

Rotate coordinate system for vertical vector

DKSV c_0, c_1 ;

The free coordinate system is rotated around the z axis until the vector $c_0 \rightarrow c_1$ specified by the codes c_0 and c_1, points vertically on the screen (towards the top). The reason for this command is to place tilted pictures in an upright position especially after using 'FM'.

Rotate and redraw

D a, w, plt, opt_1, opt_2 ;

'D' is a built-in macro. The following commands are executed:
DK a, w (or DKV 0 0 w if $a = 4$);
BF if $opt_1 \neq 0$;
STPU if this word (or S) was specified for plt, otherwise EPU;
BUFN 0 0 if $opt_2 \neq 0$.

After the re-drawing, the macro waits for input. The following input is accepted: 1, 2, 3, 4, -1, -2, -3, -4, -, 9, or simply "enter" (= return key). If the number entered is different from 9, it will be the new rotation axis. 4 has the meaning that the vector is used as rotation axis defined by the first two codes in the mouse list. The '-' sign inverts the direction of the rotation. If only the return key is pressed, the execution is repeated. The number 9 has a special meaning: The control is switched to another built-in macro 'VDGC'. All other input terminates the macro (and is interpreted as new command) but the current values (for axis, angle etc.) are kept. At the start of the program, the following values are set: axis: $a = 2$, angle: $w = 5$, plt = EPU, $opt_1 = 1$, $opt_2 = 0$.

Hint: If option `PLT 2` is in effect the origin of the free ccordinate system of the foreground structure should lie on the freely selectable axis (4). Otherwise this point will be modified, but not the corresponding point of the background structure.

Calculate free cordinates

FK ;

During the execution of some commands, the internally calculated and saved free coordinates are overwritten while the drawing remains on the screen. If so, the commands using the mouse will no longer work correctly. 'FK' recalculates the free coordinates, and a redrawing may be unnecessary.

Drawing area and scaling

Size of drawing area

ZF $xdim, ydim, brdr$;

The dimensions of the drawing area $xdim$ and $ydim$, together with the width of the border $brdr$ are defined in cm. The command 'BF' uses these values to calculate the scaling factor. The factor is determined by demanding that the centers of the atoms fall on the border inside the drawing area either in x or in y. Note that the program does not prevent crossing these limits.

Default values are: 24,19,1 on pc's; 33,21,1 on the RS/6000. The maximum values allowed for the HP laserprinter are 27.09,20.22.

Exchange drawing field dimensions

XZF ;

Before replacing the values used with 'ZF', the old values are saved. Using 'XZF' interchanges the current and the saved values. This is useful when temporarily switching to a different output device (e.g. printer).

Half resp. double drawing area

HF resp. DF ;

When generating a stereoscopic pair of pictures, one has to remember that only one half of the drawing area is available for each picture. 'HF' and 'DF' are useful for switching from single to stereo plots and vice versa. The scaling factor will also be recalculated. But note that the ratio $xdim : ydim$ is altered; thus, a better way is to recalculate the scaling factor using 'BF'.

Shift field

VF x, y ;

The origin of the drawing field is shifted by the values x, y. This command is used e.g. when generating stereo plots. Defaults are: *xzf*,0 and 0,*yzf*, respectively depending on the value of the option 'STPO'. Here *xzf* and *yzf* are the current values of the drawing field (initial setting 24.0, 19.0 with the DOS version).

Field origin

FO x, y ;

Similar to 'VF' the origin of the drawing field is redefined. But in contrast to 'VF' the definition is done "absolutely" (in cm). Defaults are 0,0.

Example: Suppose one needs a stereo plot, but in addition the atoms 'C' should be labeled with their code. The plot area dimensions and the scaling factor are supposed to be set correctly. The following macro does this job.

```
macr 9
ls
vf;dk 2 -3;srt 3;expu;pscd -.2 -.1 C
fo;dk 2  6;srt 3;expu;pscd -.2 -.1 C
dk 2 -3
endm
```

Origin on the drawing area

UR x, y ;

Using 'UR' one can explicitely specify the position of the origin of the free coordinate system on the drawing area (in cm). Default values are: 10.25,6.25 . Note that 'BF' recalculates these values.

Viewing distance

H h ;

The viewing distance h has to be entered in cm. The centers of the atoms are projected in central projection from this viewing point $(0, 0, h)$ onto the x-y-plane. Default value is 60.

Print viewing distance

OH ;

The current viewing distance will be printed.

Factor definition

F f ;

The free coordinate system is an Angstrom-system. The plotting field in contrast is a cm-system. Hence, a scaling factor has to be supplied defining the ratio cm/Å (initial setting: 1.0). Usually this scaling factor is determined automatically by 'BF', but sometimes it is necessary to define this scaling factor f directly e.g. when comparing two structures. Note that this value is also applied to the viewing distance (c.f. command 'H').

Best fit (scaling factor)

BF ;

The scaling factor (see 'F') and the coordinates of the origin of the free coordinate system (see 'UR') are calculated. This is done, in order to fit the model (stored in the code list) to the drawing field as defined with 'ZF'. After applying 'BF', the centers of the outermost atoms lie on the border of the drawing field.

Alter scaling factor

AF df ;

The current scaling factor is multiplied by df.

Factor for sphere

KF f ;

A global factor f (initial setting 1) applying to all spheres may be defined before producing a drawing. This does not change any list.

Print plot settings

OZB ;

The current values are printed for the following parameters: Size of the drawing area, border, the coordinates of the origin, the scaling factor, the viewing distance, and the letter size.

Pixel and cm specification for a graphic terminal

PXCM p_x, cm_x, p_y, cm_y ;

If the setting of the ratio pixels/cm of the graphic terminal does not correspond to the values used by the program the image will be distorted. 'PXCM' resets the values initially set for the program. One has to specify the numbers of the highest addressable pixels p_x and p_y, and the corresponding cm for x and y, cm_x and cm_y, respectively. When a zero is entered for p_x or p_y, the current value is not changed. An automatic change of the values used by 'ZF' does not take place.

Plotting commands

Letter size

SH h ;

A letter size h (in Å) is defined. This value will be used as default in most commands dealing with text. Initially h is set to 0.2 .

Pen definition (default color)

NPEN nr ;

A color number nr is set which is used as default for commands expecting a color specification (initial setting 15). Devices which are not able to handle colors ignore this specification.

Thickness (of lines)

STRD nr ;

Some devices are able to produce lines of different thicknesses. nr (initial setting: 3) is the default number for plot commands expecting such a specification. For pixel graphics, an odd number may be entered $1, \ldots, 9$. For HPGL code, the number is divided by 10 and converted to a 'PW' command.

Plot balls (spheres)

PK n_1, n_2, s, p ;

This command is stored in the table of plot commands. When executing 'EPU' or another command of that kind, the codes of the atoms stored in the code list are checked, whether they have atom numbers in the parameter list between n_1 and n_2. (The actual position in the code list does not matter.) If the condition is matched, a circle is drawn around the projected center, i.e. a polygon having s vertices (default: 30). The radius of this "sphere" is calculated from the radius given in the parameter list multiplied by the global factor for spheres (set in 'KF') and the perspective distortion. Default values are $n_1 = 3$, n_2 = number of the last atom, and $s = 30$. (s must lie between 2 and 60.) If for s a negative number is entered, atoms will be labeled automatically. However, a cleaner way to achieve this is to use 'BSF'. The parameter p points, if given, to a control block which can be defined with 'SKG' and contains further details how to draw the atoms (default: 0). Other details are taken from the ATF block to which the atom in the parameter list refers.

Plot spheres by symbol

PKS sym, s, r, n_1, n_2, p ;

PK commands will be created for all codes in the code list belonging to atoms with the symbol sym. In addition all radii of the atoms with the symbol sym may be set to r. A value of zero (default: 0) does not change the radii. The generation of plot commands may be restricted to a certain part of the parameter list, ranging from n_1 to n_2. Defaults are: $n_1 = 3$ and n_2 = last atom. The parameter p points, if given, to a control block which can be defined with 'SKG' and contains further details how to draw the atom (default: 0). For s, c.f. 'PK'. Other details are taken from the ATF block to which the atom in the parameter list refers.

Shade spheres

SKG $nr, c, t, iflag, d, w_0, w_1, rep, l_1, l_2$;

With the command 'SKG' up to 6 control blocks containing the details of the shading (using hatched patterns) of the spheres can be defined. nr: number of the block, $1, \ldots, 6$; this data must correspond to the parameter p in the corresponding 'PK' command. No default values are provided. c: color of the lines; t: thickness of the lines (0 means no hatching); $iflag$: 0: values for d, l_1, and l_2 in Å; 1: values in cm; d: distance of the hatching lines; w_0: starting angle of the hatching; w_1: increment of the angle; rep: repetition of rotating. If values are entered for l_1 and l_2 dashed lines are drawn, where l_1 is the length of the line and l_2 the length of the space between two dashes.

Print shading control blocks

OSKG ;

The control blocks that have been defined by 'SKG' are printed.

Filling pattern

FLM *nw,nl,pattern* ;

Some versions of KPLOT, which can create pixel graphics, can fill the interior of the spheres, sticks, and polyhedra planes with a filling pattern. The filling pattern is defined by 32 bit numbers (words) where a one corresponds to a dot and a zero to an empty space. Three filling patterns are predefined: 0: empty; 1: black (full); 2: grey (light). Further filling patterns may be defined. nw: number of 32 bit numbers after which a repetition is performed; nl: number of lines of nw words; *pattern*: hexadecimal numbers.

Example: The pattern no. 2 'grey' is defined as follows: `FLM 1,4,-55555556,0,55555555,0`

Caution! On VAX computers the bytes must be specified in the order 4,3,2,1.

Filling pattern in bytes

FLMB $b_1, \ldots, b_8$;

To simplify the input of a filling pattern, it can be specified by 8 bytes. A square of 8×8 pixels is defined by $b_1, \ldots, b_8$ which will be used as pattern (no default). For example, the pattern for grey is given by:

```
FLMB C0 C0 0 0 0C 0C 0 0
```

Labeling flag

BSF n ;

When creating a plot the atoms may be labeled automatically. The value of n controls the manner of labeling: $n = 0$ (default): no labeling (except if the number of vertices at 'PK' has been entered as a negative number); $n = 1$: label atoms using the first two letters of the symbol, the second letter is converted to lower case; $n = 2$: the first two letters of the extension (i.e. the second string of the atom name) is used as name; $n = 3$: same as $n = 1$ except no conversion takes place; $n = 4$: the position of the atom in the parameter list is taken as label. If this number is greater than 99, it is continued using letters A,..,Z i.e. A0=100,...,Z9=359. Numbers greater then 359 will be left out. $n = 5$: same as $n = 4$, but only atoms having codes $xxx55501$ will be labeled; $n = 6$: same as $n = 5$, but only atoms having codes $xxx555yy$ will be labeled.

Size of label

BSN n ;

Usually, atoms are labelled in drawings with at most two letters (c.f. 'BSF'). This number can be changed using 'BSN'. n may be 1, 2, 3, or 4.

Do not plot

NP resp. NPN $c_{11}, c_{12}, c_{21}, c_{22}, \ldots, c_{n1}, c_{n2}$;

Sometimes it is desirable, not to plot selected atoms and/or bonds, although they would be drawn due to the current plot commands. The input describing such atoms or bonds must consist of pairs of codes. Two consecutive non-zero codes specify a bond between those atoms, while a non-zero code followed by a zero refers to an atom. Using 'NPN' instead of 'NP' the position(s) of the atom in the code list is used instead.

Do not plot according to mouse list

NPM ;

Atoms and/or bonds which are not supposed to be plotted, may also be identified via the mouse list (see 'LGC'). To select a bond the two atoms involved are clicked on, and to select an atom, this atom is clicked on twice. (Again, an even number of codes must be contained in the list to be valid).

Clear non-plot list

LNP ;

The list created by 'NP' resp. 'NPN', that contains those atoms and bonds which are not to be drawn, is cleared.

Tapering factor

TAPR f ;

If bonds are represented by sticks in a plot, the spatial impression can be enhanced by drawing a cone instead of a stick. The value by which the radius of the stick is increased resp. decreased is $(1 \pm f)d$ for the upper and the lower atom respectively. d is the difference in z (in Å). The default value for f is 0.3.

Define bonds

VB or VBS or VBR $u_1, u_2, z_1, z_2, dmin, dmax, r, p$;

Atoms will be connected by single lines or two lines ("sticks"), if the following conditions are fulfilled: The atoms must be in the code list, and one of them have a number in the parameter list between u_1 and u_2 ("origins of the bonds") while the other has to be between z_1 and z_2 ("end points of the bonds"), respectively. Finally, the distance must lie between *dmin* [0] and *dmax* (no default). Using 'VB' the line is drawn from center to center of the atoms. Using 'VBS' the line ends at the surface of the spheres. Respecting hidden lines the line at the atom nearer to the viewer begins at the border. When using 'VBR' two lines will be drawn ("sticks") which will be tapered (see also 'TAPR'). r is the radius of the stick, default: $r = 0.06$. r has a special meaning in connection with 'VB' and 'VBS': $|r| = 10n + v > 20$ is interpreted as a dashed line with n dashes and $n - 1$ spaces, where the ratio of the two lengths, line:space, is v. If a negative number is entered for r, the meaning of line and space is interchanged. p is a pointer to an ATF block (default: 11).

Join directly

VD or VDS or VDR $r, p, n_1, n_2, n_3, c_1, c_2, \ldots, c_n[, 0, d_1, d_2, \ldots, d_m]$;

Atoms having codes $c_1, \ldots, c_n$ in the code list are joined in this order, resulting in a line consisting of $(n-1)$ straight pieces. Here, no distance is specified. Further chains may be specified, separating them by inserting a zero. r and p have the same meaning as for 'VB'. n_1, n_2, and n_3 are not used anymore in KPLOT-versions 7 and higher; enter *** for n_1, n_2, n_3.

Join directly using the mouse

VDM cmd, r, p ;

This command corresponds to the command 'VD' etc., but here the codes are entered via the mouse list. One can specify 'VD' (default) or 'VDS' or 'VDM' for *cmd*. To indicate that a new chain is going to start with the next code, click twice on the terminating atom of the previous one.

Join according to symbols

VS or VSS or VSR $s_{ur}, s_{zl}, dmin, dmax, r, p, x_1, x_2, x_3, n_1, n_2$;

Atoms (in the code list) having symbol s_{ur} are joined to those having the symbol s_{zl}, if they are present in the code list and the distance between the codes lies in the interval $(dmin, dmax)$. Note that this command is not stored as given, but is converted to 'VD', 'VDS', or 'VDR' respectively). One can restrict the search to parts of the parameter list by entering values for n_1 (default: 3) and n_2 (default: last atom). r is the radius of the stick, default: $r = 0.06$. r has a special meaning in connection with 'VS' and 'VSS': $|r| = 10n + v > 20$ is interpreted as a dashed line with n dashes and $n - 1$ spaces, where the ratio of the two lengths, line:space, is v. If a negative number is entered for r, the meaning of line and space is interchanged. p is a pointer to an ATF block (default: 11). x_1, x_2, and x_3 are not used anymore in KPLOT-versions 7 and higher; enter *** for x_1, x_2, x_3.

Example: The command

```
VSR C C 0.1 1.7 0.06
```

will generate the 'VBR' commands necessary to join all C atoms having a distance between 0.1 and 1.7 Å. Several 'VBR' commands are generated, if the C atoms are stored in several blocks in the parameterlist.

Plot polyhedra

cmd $u_1, u_2, z_1, z_2, dmin, dmax, c_1, c_2, d_n, d_x, tol, ndcb, opt$;

Around atoms which are in the code list having numbers in the parameter list between c_1 and c_2 (central atoms) atoms are searched for in the code list, which have numbers in the parameter list between u_1 and u_2

or z_1 and z_2 (usually 0,0; default) and which fall into the distance interval d_n, d_x (referring to the distance of the corners of the polyhedra to the central atoms). (*dmin* and *dmax* are no longer used in KPLOT-versions 7 and higher; they are present to be compatible with earlier versions. One usually would enter the default values: 0,0) The corners selected in this way are connected by lines, if they belong to outer planes.

Example: The Ti atom of Brookite (TiO_2) is surrounded by a distorted octahedron of oxygen. Let Ti have the number 3 and O the numbers 4 to 5 in the parameterlist. The greatest Ti - O distance is less than 2.1 Å. A possible command would be: `VEU 4 5 0 0 0 0 3 3 2.1`

The representation of the polyhedra is done as follows: Using 'VEU' the polyhedra are opaque; using 'VEH' they are half transparent, i.e. one can see the interior of the polyhedra but not what lies behind them; using 'VE' the polyhedra are completely transparent.

The program also checks, whether planes might be definded that contain more than three vertices. For such a plane to exist, all the atoms involved must belong to this plane within the tolerance *tol*. More precisely: Take an arbitrary set of three points which define the plane. The remaining points must lie within that plane within the tolerance $\pm tol$. The default value for *tol* is taken from the setting of 'VEO' and is set initially to 0.1 Å. Further details may be specified via a data control block no. *ndcb* (see 'SF'). The parameter *opt* (see 'VEO') controls the sequence in which the planes are drawn.

Plot planes

PF or PFU $u_1, u_2, z_1, z_2, dmin, dmax, c_1, c_2, d_n, d_x, tol, ndcb, opt$;

Since it is impossible to plot single planes using 'VE', 'VEH', or 'VEU', there exist special commands, 'PF' and 'PFU', respectively. The parameters are the same as for 'VE' etc. Using 'PF'/ 'PFU' the planes will be transparent / opaque, respectively.

Plot plane via codes

PFD $ndcb, c_0, c_1, \ldots, c_n$;

This command allows the direct description of a plane via the codes $c_1, \ldots, c_n$. The projections of the codes $c_1, \ldots, c_n$ onto the x-y-plane are connected thus describing the area on the screen enclosed by this polygon (it is up to the user whether it makes sense). Up to eight codes $c_1, \ldots, c_n$ are allowed. c_0 is an input that "controls" the execution of the plot: If all atoms above the (opaque) plane should be visible, c_0 has to be an atom farther from the viewer than $c_1, \ldots, c_n$. To generate a transparent plane, enter for *ndcb* (see 'SF') a negative number, otherwise the plane will be opaque.

Polyhedra options

VEO *opt,tol,dsh-code,veu-opt* ;

Hidden planes or parts of them are a special problem when dealing with polyhedra. Every plane is handled as an individual object. Therefore the sequence of plotting the planes is important. The parameter opt (initial setting 1) is the default value when giving one of the commands 'VE', 'VEH', 'VEU', 'PF', or 'PFU'. One atom is assigned to each plane which controls the plot of the plane, i.e. the plane is drawn when this atom is reached. Usually the best choice is an atom near the center of the plane ($opt = 1$). $opt = 0$ selects the atom of the plane which is farthest from the observer. If $opt = 2$, for the back planes the atom farthest from the observer is chosen, and for the front planes the nearest one. This may be useful, if atoms in the interior of the polyhedra should be visible. A special meaning has $opt = 3$ when using 'PF' or 'PFU': The z coordinate of the central atom is compared with the central point of the plane (polygon); if the central atom lies before the central point (larger z) plotting the plane is controlled by the atom of the plane having the largest z coordinate, otherwise the smallest one.

There exists another problem due to the fact that in general four (or more) points do not lie in a plane. *tol* gives a tolerance (initially: 0.1) how far an atom may be away from the plane while being still treated as if it belonged to the plane. The value given here is used as default for 'VE' etc., and may be overwritten in these commands.

The edges belonging to two back planes may be represented by dashed lines with 'VE' or 'VEH'. To do so, define a *dsh-code* s (real number) defined by $s = 10n + f$ (default: 0.0). The lines will be drawn as n dashes and $(n-1)$ spaces where the ratio of the lengths, space:dash, equals f. Example: $s = 51$ means 5 lines and 4 spaces with equal legth. If the *dsh-code* is negative, dash and space are interchanged.

Usually back (invisible) planes are not drawn when using 'VEU'. But there are cases where planes are highly distorted (e.g. consisting of more than 3 points). If veu-opt is given a value unequal zero these planes will be drawn nevertheless.

Shade polyhedra planes

SF $nr, c_r, s_r, c_s, s_s, c_f, n_f, dist, v_1, v_2, v_3, w, n_t, n_s$;

Up to six data control blocks may be defined containing details of how to draw the planes of the polyhedra. nr is the number of the block. Defaults are given only for the first block; all others are set to zero. The parameters have the following meaning (defaults given in brackets):

c_r	color of the edge lines (15)
s_r	thickness of the edge lines (0)
c_s	color of the hatching lines (15)
s_s	thickness of the hatching lines, (0: no hatching)
c_f	color of the filling pattern, (15)
n_f	number of the filling pattern (0: no filling)

If $s_r = s_s = n_f = 0$ the control block is treated as undefined. *dist* is the distance of the hatching lines in Angstroms when shading. v_1, v_2, v_3 are the components of a vector in the triclinic system. The hatching will be done parallel to the edge having an angle nearest to 90° to that vector. If w and n_t have values $\neq 0$ (initial setting: 0,0), the hatching will be repeated n_t times and rotated around the angle w. n_s is the starting number (initial setting: 0).

Print hatching data control blocks

OSF ;

The hatching data control blocks will be listed.

Type of labeling for atoms

BST n ;

The labeling of atoms with 'BA' can be done in one of three different ways which is controlled by n. If $n = 0$ (initial setting), the atoms are labeled with their name only; if $n = 1$, the name and extension are concatenated to one name; using $n = 2$, the extension is put in brackets (if not blank) and concatenated to the name. *Examples:*

n	Name	ext.	Appearance
0	Na	1	Na
1	Na	1	Na1
2	Na	1	Na(1)

Label atoms

BA $c, t, s, c_1, x_1, y_1, \ldots, c_n, x_n, y_n$;

The atoms having codes $c_1, \ldots, c_n$ are labeled with the symbols as given in the parameter list. x_i and y_i are offsets in Å relative to the center of the atom c_i. There are no defaults. c is the color, t the thickness, and s the size of the letters in Å. Note that this command only stores the values. Like 'PK' it is executed when 'EPU' or a similar command is given.

Label atoms using the mouse

BAM ;

Setting labels is much simpler when using the mouse. If the command 'BAM' is given, the mouse cursor is shown. Now the atom to be labeled can be clicked on (left button). On the top of the screen the data of the atom appear, and the mouse cursor takes the shape of an L. This is the position of the lower left corner of the first letter of the text. Pressing the left mouse button will set the label. If one is not satisfied with the position, a better one may be selected and clicked on. The earlier label will be erased and moved to the new location. This procedure can be repeated as often as desired. Pressing the right mouse button signals the acceptance of the choice. Now the mouse cursor takes on the shape of a cross, and a new atom may be selected. Pressing the right button terminates the process. Don't forget to set the 'BST' parameter before using 'BAM'! The parameters for color, thickness, and lettersize are taken from the current settings 'NPEN' (initial setting 15), 'STRD' (initial setting 3), and 'SH' (initial setting 0.2).

On Atari computers additional user input by pressing keys is possible: A (or a): do not label this atom; G (or g) increase letter size by 10 %; K (or k) shrink letter size by 10%.

Remove group of codes from the labeling commands

EGCB ;

The list containing the plot commands is checked for 'BA' commands. If such a command contains a code which is also on the mouse list, this code will be removed from the 'BA' command.

Label with text

BT c_1, c_2,*xoff,yoff,* $s, text, c, t$;

The *text* (up to 40 letters) is written parallel to the vector $c_1 \rightarrow c_2$ given by the codes c_1 and c_2 (no defaults). The program calculates the midpoint between c_1 and c_2 and adds the given values *xoff* und *yoff* (in Å, defaults 0,0). This is the lower left corner of the first letter of the text. s is the letter size (in Å, default: see 'SH', initial setting 0.2). The default values for color c and thickness t are taken from the current settings of 'NPEN' (initial setting 15) and 'STRD' (initial setting 3).

Label with text using the mouse

BTM *text* ;

This command works similar to 'BAM'. The *text* entered (up to 40 letters) is positioned by the mouse. Before placing the text both atoms must be clicked on. If the *text* refers to only one atom, this atom must be clicked on twice, i.e. $\text{code}_1 = \text{code}_2$.

Label vector with its length using the mouse

BVLM t, n ;

If one wants to label a vector (usually a bond) by its length, this can be done conveniently using 'BVLM'. The vector is defined by two mouse clicks. From this information the length is calculated and written parallel to the vector as label. The positioning of the label is done in the same way as with 'BTM' (in fact, the length is converted to a string and 'BTM' is used). $t = 0$ (initial setting: 0): Give length in Angstroms, otherwise ($t \neq 0$) in pm. n (initial setting: 2) is the number of digits after the decimal point. The values entered are kept and are the defaults for the next time 'BVLM' is used.

Shift label

VSFT n, m ;

Sometimes it is necessary to shift labels slightly, expecially the automatically generated labels of the coordinate cross (see 'GNKO'). The number n of the plot command which is responsible for drawing the label must be given, and in case of a 'BA' command the position m within the command. If the input is valid, the mouse cursor is shown. By clicking on two arbitrary points on the screen a vector is defined which will be added to the offset. The values entered are stored, in order to be available, if this procedure has to be repeated.

Plot numbers resp. codes

PN resp. PCDS n_1, n_2, *xoff, yoff* ;

All atoms in the code list, which have numbers in the parameter list between n_1 and n_2 are labeled. With 'PN' the current position in the code list is used as label, and with 'PCDS' the complete code, respectively. *xoff* and *yoff* are offsets (in Å) of the lower left corner of the first digit of the label, referring to the center of the atom. Defaults: $n_1 = 1$, n_2 = last atom, *xoff* = *yoff* = 0.

Label symbols by numbers codes respectively

PSN resp. PSCD [*xoff, yoff,*] $s_1, s_2, \ldots$;

Atoms which are in the code list having one of the symbols $s_1, s_2, \ldots$ (no defaults) are labeled with their position in the code list or with the code. The values *xoff* and *yoff* are an offset (in Å) with respect to the center of the atom. Defaults are the values used last, initially 0.5,0.5. The commands 'PN', 'PCDS', 'PSN', and 'PSCD' are not stored but executed immediately instead.

Print plot commands

OPK n_1, n_2 ;

The plot commands are printed in the sequence in which they have been entered, beginning with n_1 and ending with n_2. If only one number is given, only that plot command is printed ($n_2 = n_1$ as default). If no number is entered, all stored plot commands are printed.

Remove plot commands

LPK ;

The list of plot commands is cleared. This is done automatically when starting the program.

Remove plot command(s)

EPK n_1, n_2 ;

Plot commands having numbers from n_1 to n_2 will be removed. If only one number is given, only that plot command is removed ($n_2 = n_1$ as default).

Mask plot command(s)

MPK resp. UMPK n_1, n_2 ;

Sometimes it is desirable to inactivate certain plot commands without removing them from the list. MPK masks all plot commands from n_1 to n_2. If only one number is entered, only that command is masked (default: $n_2 = n_1$). The masked plot commands will not be executed with 'EPU' etc. When printed, masked plot commands are preceded by an asterisk. 'UMPK' removes the mask.

Modify plot command (parameters)

APK n, par ;

The parameters of the plot command at the position n (no default) in the list of plot commands may be changed. This is done in such a way that all parameters associated originally with the plot command are used as defaults.

Change plot command (type)

APBF n, cmd ;

In contrast to 'APK', this command allows the replacement of the plot command n by a similar one, denoted *cmd*. The following list shows the interchangeble commands.

- VB, VBS, VBR;
- VD, VDS, VDR;
- VE, VEH, VEU, PF, PFU.

Change color pointer

AFZ n, n_a, n_b ;

Defaults: $n = 0$, $n_a = 1$, $n_b = 999999$. Plot commands which are stored in the list from n_a to n_b, and which have a color pointer are altered such that this color pointer is set to the number n. This may be useful when comparing structures by plotting them simultaneously to use different colors for both.

Move plot command(s)

MVPK $n_1[, n_2], n_3$;

When drawing bonds, the sequence of plot commands is important: only the first matching command is executed. 'MVPK' may be used to reorder the plot commands, by moving the block of commands from n_1 to n_2 before the position n_3. n_2 may be omitted. If a number n_3 is specified which is greater than the number of commands stored, this is interpreted as 'move to the end' of the list.

Start plotter

PST ;

On some computers (IBM) graphics software must be initialized separately. Ask your system manager for more details.

Generation of the drawing

Plot resp. do not plot frame

PR *opt* resp. NPR ;

Using the command 'PR' with $opt = 1$, a frame with the dimensions of the drawing field is plotted after the completion of a drawing. Note that according to a DIN rule the atom nearest to the observer has to lie in the plane of the frame when generating a stereo plot (see 'KUSP' and 'COC'). Option 'NPR' suppresses plotting of a frame. (PR 0 = NPR) A special meaning has $opt = 2$: When a pixel graphic is generated, two

dots are placed in the left upper and in the right lower corner. This is done in order to enforce some white space around the graphic. The default is $opt = 1$.

Plot options

PLT n;

The option 'PLT' controls whether or not graphical output is produced at all. Initially `PLT` $= 1$ is set i.e. graphical output is produced. If one wants to suppress all graphical output, `PLT` $= 0$ must be set. `PLT` $= -1$ is similar to `PLT` $= 1$, but bonding sticks are drawn as closed polygons in KPLOTH. This option may be useful, if the HPGL code generated is to be processed by another program. A special meaning has `PLT` $= 2$: in this case both structures (background and foreground) are plotted as an overlay image.

Plot output

PO resp. NPO ;

Beside the graphical output KPLOT optionally generates a list with 'EXP' and similar commands indicating, which atoms are to be drawn und to be connected. 'NPO' (initial setting) suppresses the generation of this list.

Clear screen

LS ;

If used with a graphics terminal, the plot will be removed. Other devices will ignore this command.

Execute plot commands

EXP bzw. EXPU ;

Most plot commands are not executed immediately but stored in a list. Entering 'EXP' or 'EXPU' start their execution. If 'EXPU' is used, hidden lines are taken into account.

Hidden lines are only correctly taken care of, if the code list has been sorted with respect to the z coordinate in the free coordinate system right before executing these commands . Therefore the command 'SRT 3' has to be given if sorting has not already occurred. Note that sometimes there is a conflict regarding the proper order of drawing lines when dealing with very "long" ones, e.g. the edges of a unit cell.

Stereo plot

STP resp. STPU ;

These commands are constructed as macros, because the procedure to generate a pair of stereoscopic pictures is always the same. The following macro is assigned to the commands:

```
LS                (clear screen)
VF xdim,0         (shift field)
DK 2 -w/2         (rotate by half the stereo angle)
SRT 3             (sort according z (STPU only))
EXP or EXPU       (execute plot commands)
DK 2 w            (rotate by stereo angle)
SRT 3             (sort according z (STPU only))
VF -xdim,0        (Shift field)
EXP or EXPU       (execute plot commands)
DK 2 -w/2         (restore the original orientation)
```

Here w is the stereo angle as defined using 'SW' (initial setting 6). $xdim$ is the width of the plot field (c.f. 'ZF').

Stereo angle

SW w ;

In the real world objects are seen stereoscopically because they appear rotated differently with respect to each eye. In KPLOT the corresponding angle w (initial setting: 6) may be set to a suitable value. Note that this value should be consistent with the distance of the observer.

Stereo plot option

STPO n ;

If n (initially 0) is given as 0, the plots for the right and the left eye are generated side by side. If $n = 1$ is specified the plot for the right eye is generated above the one for the left eye. Note that the program does not take into account, whether the drawing field has been specified correctly.

Single plot respecting hidden lines

EPU ;

This command works like 'STPU', but no rotation is performed and only one picture is created.

Graphics

G or GG ;

Since the command sequences BF;EPU and BF;STPU are very common, there exists a macro 'G' that fulfills the same role. If the last command was 'EPU' or 'HF', 'G' corresponds to BF;EPU, while after 'STPU' or 'DF' 'G' corresponds to BF;STPU. 'GG' corresponds to BF;STPU always.

Multi-Buffer Plot

MBUF n ;

If KPLOT is installed on a machine where the graphics may be generated using more than one buffer (and supposedly enough computer power is available), one can create animated drawings. Using 'MBUF' the creation of n buffers is initiated. The program will return the number of available buffers and enter the multi-buffering mode.

Buffer number

BUFN n, m ;

If KPLOT is supposed to use the multi-buffering mode, e.g., to animate drawings, the basic technique is to "fill" a certain buffer while displaying a different one. If the filling is complete, a switch has to be performed to show the filled buffer and supply a new one for filling. This switching is done by 'BUFN'. n is the buffer number to be displayed and m the one to be filled. If 0 is specified either for n or for m (defaults: 0,0), this is interpreted as 'next' buffer, i.e. $n+1$ or $m+1$ respectively if possible, and 1 otherwise.

The following example shows the progamming of an animation using a macro (explanations in brackets):

```
MACR 1
SETC %2          ( Number of plots to be generated )
DKV 1 156501 %1  ( Rotation axis is the b axis )
EPU              ( Generate plot in background )
BUFN 0 0         ( Switch buffers )
DMNZ 2           ( Continue at line 2 if counter not zero )
WAIT             ( Here the macro may be terminated ... )
MVTO 2           ( ... or continued. )
ENDM
```

The macro is invoked e.g. by the command

```
1 2 300
```

300 plots will be generated while every time the model will be rotated around the b-axis by 2°. If the enter key is pressed after the macro finishes, the process will be continued. All other input terminates the macro. On some operating systems the escape-key is supported, e.g. for DOS. If so, the macro may be simplified by programming an endless loop. Then the macro may be terminated just by pressing the escape key. The macro then would look like:

```
MACR 1
DKV 1 156501 %1  ( Rotation axis is the b axis )
EPU              ( Generate plot in background )
BUFN;            ( Switch buffers )
MVTO 1           ( continue)
ENDM
```

Initialize animation

AINI $vmod, mbuf, p_1, p_2$;

'AINI' is an built-in macro for initializing an animation for DOS. The video mode will be switched to *vmod* [1] (see 'VMOD'). $mbuf$ [2] buffers will be requested for the graphics, and the buffer sequence set to p_1, p_2 [1, 2]. For more details see 'MBUF' and 'BUFN'.

Plot format

PFMT n ;

When generating a plot on a laser printer or when generating HPGL code, this can be done in landscape format ($n = 0$) or portrait format ($n = 1$, default).

Generate HPGL code

HPGL *file* resp. HEND ;

HPGL code may be generated using the programs AKPH (Atari) or KPLOTH (PC's), as follows. After the command 'HPGL' has been given, all plot commands are not only executed on the screen, but also a driver will be invoked writing a *file* containing the HPGL code of that plot. A special meaning has the file name 'PRN' or 'PRN:': Then the HPGL code is not written to a file but sent to the printer port (PC's). 'HEND' stops the generation of HPGL code.

Generate code for Pixel graphics

PXLG $file$ resp. PXLE ;

Those versions of KPLOT which can generate special code for the EPSON printer LQ-500 (AKPE), for the HP laserprinter II P (AKPX, KPLOT), or the HP DeskJet 550C (KPLOTF, AKPF) achieve this as follows. A driver is invoked, if the command 'PXLG' is given. The file is opened, if the file name is not 'PRN' or 'PRN:'. In those cases, the information is sent to the printer port. In contrast to the generation of HPGL code, the plot is not only generated on the screen but also internally as a bit map. This bit map is *not* automatically written to file or sent to the printer port; this has to be done explicitly using 'WPBF'. 'PXLE' terminates the driver.

Write plot buffer

WPBF n ;

As mentioned above pixel graphics are not automatically sent as output to the printer or to a file, because the computer does not "know", if the drawing is complete. $n = 0$ (default) produces code for the printer HP laserprinter II P (or higher). $n = 1$ produces a .PCX file and $n = 2$ a .IMG file. If with 'PXLG' filenames with the appropriate extensions have been specified, $n = 1$ resp. $n = 2$ will be used as defaults. A given n overwrites the default settings.

Miscellaneous commands

Length

L c_1, c_2 ;

The length of the vector $c_1 \to c_2$, specified by the codes c_1 and c_2 (no defaults) is calculated and printed. The result is also stored serving as default for several other commands.

Alter length

AL f, s ;

The length calculated last by 'L' is altered via the formula $l' = lf + s$. The result replaces the old value. Defaults: $f = 1$ and $s = 0$.

Normalize vector

VNRM c_1, c_2 ;

The vector $c_1 \to c_2$, specified by the codes c_1 and c_2 (no defaults) is scaled three times such that the x, the y, or the z coordinate take on the value 1 in turn. The results are printed.

Angle

W c_1, c_2, c_3 ;

The angle which is enclosed by the vectors $c_2 \to c_1$ and $c_2 \to c_3$ is calculated and printed. c_1, c_2, and c_3 are codes (no defaults). The result is stored and used as default for the angle in many commands.

Angle by four points

WVP $c_{1a}, c_{1b}, c_{2a}, c_{2b}$;

The vector $c_{2a} \to c_{2b}$ will be shifted until the points having the codes c_{1b} and c_{2a} are the same. Then the angle $c_{1a} - c_{1b} - c'_{2b}$ is calculated, printed and stored, just as in the command 'W'. Caution! This angle is *not* the torsion angle.

Alter angle

AW f, s ;

The angle calculated last by 'W' is altered according to the formula $w' = wf + s$. The result replaces the old value.

Torsion angle

TW c_1, c_2, c_3, c_4 ;

The torsion angle which is determined by the four codes $c_1, \ldots, c_4$ (no defaults) is calculated and printed. It is the angle $c_1, c_2 = c_3, c_4$ which is produced when projecting these four points onto a plane perpendicular to the $c_2 \rightarrow c_3$ vector. If c_2, c_1 is mapped onto c_3, c_4 by a clockwise rotation (in the shortest way), the angle is taken positive (right handed screw), otherwise negative. The result is stored and used as default for the angle in many commands.

Angle between planes

EW $p_1, p_2, p_3,\ q_1, q_2, q_3$;

The angle which is enclosed by the two planes defined by the points having codes p_1, p_2, p_3 and q_1, q_2, q_3, respectively, is calculated and printed. The result is stored and used as default for the angle in many commands.

Angle between vector and plane

WVE c_1, c_2 ;

The vector $c_1 \rightarrow c_2$ given by the two codes c_1 and c_2 (no defaults) is projected onto the x-y-plane of the free coordinate system, and the angle between the vector and the projected vector is calculated. If a null vector is produced by this operation, an angle of 90° is assumed. The result is stored and used as default for the angle in many commands.

Plot point

PP ;

The point in the point register will be projected onto the drawing plane, and the cross hair cursor moved to this position.

Plot coordinate system

ZK x, y, s ;

The point on the point register will be projected onto the drawing plane and a coordinate system drawn at this position. x and y are the length's of the axes (default: 1.5) in Angstroms, s the number of marks (like on a ruler) per Angstrom (default: 10).

Clear outside window

LAF c_1, c_2, f_r, s_r ;

The points corresponding to the codes c_1 and c_2 in the code list are projected onto the plotting field defining a rectangular window in the process, with c_1 and c_2 being on opposite corners. All graphics not in this window will be erased (when used with the pixel graphic version of KPLOT). A frame enclosing the window will be plotted using color number f_r (default: 15) and thickness s_r (default: 3). $s_r = 0$ suppresses the plotting of a frame.

The command 'LAF' is executed immediately. The parameters f_r and s_r are stored for future use of this command.

Using 'LAF' with the vector version (KPLOTH), the procedure is somewhat different. Usually one may follow the following steps:

```
LS          ! Clear screen
LAF ....    ! Block outer window area
SRT 3       ! Sort codes
EXPU        ! Execute plot commands
(LAF)       ! It may be necessary to repeat 'LAF'
```

Remove parameters according to group list of codes

EGCP ;

The atoms in the parameter list, which are referred to in the group list of codes (mouse list) are removed. At the same time all the codes are removed from the code list, which have atom numbers belonging to the eliminated atoms in the parameter list. The codes in the code list and the atom numbers in the parameter list are adjusted automatically.

Atoms in the cell

AIDZ n_1, n_2, opt ;

The entries in the parameters list with numbers from n_1 (default: 3) to n_2 (default: last atom) are printed together with the multiplicity as calculated from application of the symmetries if $opt \neq 0$ (default: 0). Finally all atoms with the same name are added up and the resulting sum formula is given.

Print options

OOPT ;

All options in effect will be printed (e.g. DLG or NDLG).

Print pointer

OPTR ;

All file assignments in effect will be shown (see 'EAE'). In addition the pointers AE (number of atoms in the parameter list) and CE (number of codes) are shown.

Minimize overlap

MINO $g, n,$ *dist* ;

When generating a drawing, a frequently occurring problem is the overlap of the atoms in the projection. 'MINO' rotates the model around the x- and the y-axis in steps of g [6] degrees in positive and negative direction. Each time the reciprocal values of the squares of the distances of those projected atoms which have a distance less than *dist* [0.25] are summed. The rotations $ng, (n-1)g, \ldots, -ng$ (default for n: 2) will be executed. The orientation with the smallest sum will be taken as the new orientation. If a sum of zero is calculated at an earlier stage, the execution is stopped. To increase the quality of this optimisation, one can increase *dist* and n and decrease g.

Minimize overlap (vector)

MNOV $p_1, p_2, w, n, dist$;

In a similar way as with 'MINO' the free coordinate system is rotated to minimize the overlap of atoms. An angle w (in degrees, initial setting: 10) is divided into $n-1$ (initial setting: 5) intervals and a range $\pm w/2$ is scanned by rotating the free coordinate system around the vector $p_1 \rightarrow p_2$ specified by codes p_1 and p_2 (initial setting: b-axis). The calculation of an "overlap" function is done as in 'MINO' at n points. The default value of *dist* (c.f. 'MINO') is initially set to 100.

Exchange color number

XC n_{old}, n_{new} ;

The color number n_{old} is replaced by n_{new} (no defaults) at the following locations: ATF block: first entry; BA and BT commands; value for 'NPEN'.

Set null black or white, respectively

SNS resp. SNW ;

When developing a color graphic and using a black background, the printout on white paper may not turn out to be satisfactory. To get a more realistic impression of the final outcome, the color number 0 (usually black on pc's) is interchanged with 15 (usually white). Thus, using 'SNW', the background color is changed to white (0 indicates now white and 15 black). The drivers are switched correspondingly. Using 'SNS', one can switch back to the original setting.

Definition of the color table for .PCX

DFTB $n_0, \ldots, n_{15}$;

If the colors in a .PCX file processing program do not agree with the colors shown on the screen, the translation table may be redefined. 16 numbers have to be specified. If fewer are entered, the default definition is restored. The default definition is: `DFTB 7, 4, 2, 6, 1, 5, 14, 8, 9, 12, 11, 13, 10, 15, 3, 0;`

Video mode (PC's only)

VMOD n ;

On a PC the video mode may be selected. Input a number n: 0=Default-Mode; 1=VGA 640 x 480; 2=SVGA 800 x 600; 3=SVGA 1024 x 768.

System exit (PC's only)

SYS '*DOS-Cmd*' ;

A command string *DOS-cmd* can be passed to the operating system DOS. No other platforms are supported at the time.
Example: `SYS 'dir *.kpl'`

SYS Wait

SYSW n ;

If running KPLOT under DOS, and having executed a DOS command using 'SYS', the screen is cleared, and the next KPLOT command is expected. But this behavior may be unwanted because some results of interest should have remained displayed on the screen *before* entering the KPLOT command. Choosing $n = 1$ [initial value: 0] a break is inserted in every subsequent SYS command, and execution of KPLOT is continued after pressing the <CR> key. Entering $n = 0$ this additional break is cancelled.

Escape interrupt control

ESC n ;

On some computers (PC, Atari) it is possible to interrupt many time consuming commands without terminating KPLOT. If this feature decreases the performance too much, because the escape key is checked too frequently, one can set a number n of executed commands (default: 1), after which it is checked, whether the escape key has been pressed. E.g. when drawing a structure, after each atom an escape-check is performed. If n is set to 3, this test is made after every third atom.

Drill parameter for ball and stick model

DRMO $file, f, rt, mt$;

If a ball and stick model is supposed to be constructed, the command 'DRMO' allows the calculation of the lengths of the sticks, and the angles which need to be set on the drilling machine. The plot commands and the codes stored are used for this calculation, and a file is created using the name *file* containing instructions for the model to be built which correspond to the model which is drawn. f is the scaling factor (default: 2.5 cm/Å). rt is the remaining depth (default 0.4 cm) in the ball from the midpoint to the beginning of the stick. mt is the machine type: 0 (default) Bonn, 1 Bayreuth.

Parameter for thin band

UDST $u_0, u_1, type$;

In order to make the surfaces of the spheres and the bonds more visible, one can add a border to the spheres and the sticks. Then the spheres and the sticks appear larger, with a thicker boundary layer. This enlargement is a function of the depth of the structure. The atom which is farthest away from the viewer, is assigned a boundary of thickness u_0 in Angstroms (default: 0) and the atom nearest to the viewer is given one of thickness u_1 (default: 0.08) respectively. The widths of the borders of all the other atoms are found by interpolation. Initially the *type* option is not active, i.e. $type = 0$. If type is set to 1 the enlargement applies to the bonds only; if given as -1 also to the spheres. Note that 'UDST' improves the spatial impression but is rather expensive computationally.

Which atom?

WA ;

If supported, the cross hair cursor will be shown. This cursor may be positioned on an atom in the picture drawn last, using the mouse. If 'STPU' has been used, the last picture is the left stereo plot. If clicked on, the atom, together with its parameters, are shown on top of the screen. In addition the coordinates are stored in the point register for further use. This may be repeated, until the right mouse button is pressed.

Generate coordinate cross

GNKO l_1, l_2, l_3 ;

Often, in addition to the model a representation of a coordinate cross is required, representing the directions of the lattice. 'GNKO' generates the parameters and the plot commands necessary for creating such a cross. Starting from the point on the point register which is given the name '0' three further points are generated

named a, b, and c, having distances l_1, l_2, and l_3 from '0'. Default values for the l_i is 1.2/scaling factor (chosen to achieve a size which is approximately equal in each drawing). The codes of these points are also put in the code list, and the corresponding plot and labeling commands will be generated. In order to avoid multiple crosses, a replacement takes place: if an atom is found in the parameter list having the name '0' followed by 'a', 'b', and 'c', it is assumed during the execution of GNKO that these "atoms" represent a coordinate cross, and they are removed and replaced by the new ones.

Show axes

ZA a_f, s_f ;

Sometimes, one wishes to know the location of the axes of the unit cell on the screen, without having to generate a coordinate tripod (see 'GNKO'), which would change the parameter list and the list of the plot commands. With 'ZA', the projected coordinates of four points are calculated, which correspond to the origin of the coordinate system, and the axes $a \cdot a_f$, $b \cdot a_f$, and $c \cdot a_f$ (a, b, c are the lattice constants). At these points the characters 0,a,b,c are written. Initial value for a_f is 1.05. The letter size (see 'SH') is multiplied by the factor s_f (default 2).

Show axes always

IZA n ;

Calling 'IZA' with $n \neq 0$ results in KPLOT executing command 'ZA' every time a picture has been drawn. This option is turned off by entering $n = 0$ with the command 'IZA'.

Volume of a molecule

MVOL n_1, n_2 ;

The volumina of all atoms currently in the code list and having atom numbers between n_1 [3] and n_2 [last atom] in the parameter list are added together. The atoms are treated as spheres with the radii given by the values stored in the parameter list. Of course, overlap regions will be subtracted.

Generate reciprocal cell

GRZ r, p ;

Three atoms will be stored in the parameter list having the radius r (default: 0.3) and color pointer p (default: 0) which represent the reciprocal lattice vectors. One may use them e.g. for special definitions of the orientation refering to the reciprocal space.

Generate orientation matrix

GOMX n ;

Three points which may have been generated by 'GRZ' will be written to the file having the logical number *ntpch* in the format of an orientation matrix of a CAD4 file. n is the first of the three points which must be listed consecutively in the parameter list. This command may be useful when using the program H5 (Hundt, 1995), if the twinning law is known but not the orientation matrices. The following example should illustrate the usage of GOMX:

Assume a monoclinic crystal is given, that is twinned by a rotation of 180° around the diagonal axis 101.

```
Z 11.3 14.2 11.3 90 92.1 90  ! Assumption for this example.
AE 2 ; SE 1
GRZ .3 2 ! Generate reciprocal points ...
K 2 2 265601 2 256501 ! ... define the coordinate system, ...
DG 3 5 1 180  ! and rotate those points generating new ones.
OPEN 7 M1.DAT
GOMX 3 ! Generate first matrix.
CLSE 7
OPEN 7 M2.DAT
GOMX 6 ! Generate second matrix.
CLSE 7
...
```

Import orientation matrix

IOMX r, p ;

The orientation matrix given in form of a CAD4 file or in form of a Kappa-CCD file is read in from the unit

$ntxr1$. Three atoms will be generated with the given radius r and color pointer p. Defaults are 0.003 and 0 respectively. This command may be useful when investigating a twinned crystal and a new orientation matrix has to be constructed from a given one.

Sequence for clearing the screen

SQLS $x_1, x_2, \ldots, x_n$;

Some graphics terminals require a different sequence to clear the screen. If so, the standard setting may be replaced by a user defined sequence (according to the owners manual of the terminal). Enter up to 32 characters as hexadecimal numbers. If no numbers are entered (SQLS ;), the screen will not be cleared automatically.

Sequenz for switching to text mode

SQTM $x_1, x_2, \ldots, x_n$;

Some terminals employ two levels: a graphics and a text level. Usually after the completion of a drawing, a sequence is sent to the terminal to switch to text mode. Similar to 'SQLS', this sequence may be redefined if not valid for the terminal one uses.

Unit cell transformations

Transform cell

TZ $s_{11}, s_{21}, s_{31}, s_{12}, s_{22}, s_{32}, s_{13}, s_{13}, s_{33}$;

A matrix **S** is required (c.f. Intern. Tables section on 'Unit-cell-transformation') whose coefficients have the following meaning:

$$\begin{aligned} a_{new} &= s_{11}a_{old} + s_{21}b_{old} + s_{31}c_{old} \\ b_{new} &= s_{12}a_{old} + s_{22}b_{old} + s_{32}c_{old} \\ c_{new} &= s_{13}a_{old} + s_{23}b_{old} + s_{33}c_{old} \end{aligned}$$

Here a_{old}, b_{old}, and c_{old} are the lattice constants entered last (treated as vectors), and a_{new}, b_{new}, and c_{new} are the lattice constants resulting from the transformation. The new lattice constants will overwrite the old ones. Also the coordinates of the atoms which are currently in the parameter list are transformed. Note that the same matrix **S** is also used to transform the Miller indices h, k, l.

The symmetries used by the program and the anisotropic temperature factors are also transformed. The following points have to be taken into account:

- Symmetries may be created which do not correspond to the standard setting.
- If the unit cell is increased, centering symmetries may have to be added. This is done automatically when using 'TZC'.
- Due to floating point imprecision, numbers may be calculated which are no integers or simple fractions but should be ones. These round-off errors are removed when executing the command 'OSY'.

Transform cell using codes

TZP c_1, c_2, c_3 ;

This command works like 'TZ' in principle. Instead of using the s_{ij}, here those coordinates are used which are calculated from the codes entered. I.e. the points described by the codes c_1, c_2, and c_3 should become the corners of the new unit cell. After a successful transformation, the code c_1, c_2, and c_3 have the triclinic coordinates (1,0,0), (0,1,0), and (0,0,1), respectively.

Transform cell origin

TZUR x, y, z ;

The origin of the cell is moved to the point (x, y, z). The parameters of the atoms and the symmetries are transformed according to this new origin. For side effects on the symmetries see also 'TZ'.

Transform cell origin

TZUP c ;

The unit cell is transformed in such a way that the coordinates of the code c given become those of the new origin. The parameters of the atoms and the symmetries are transformed according to this new origin. For side effects on the symmetries see also 'TZ'.

Cell and transformation of parameters

ZTAK $a, b, c, \alpha, \beta, \gamma$;

The input is to be entered in the same way as for the command 'Z'. In addition, here all parameters present in the parameter list are transformed in order to keep distances and angles unchanged. The command is provided in order to move molecule structures (finite sets) from one unit cell to a different one, e.g. in order to be able to compare similar molecules given in different unit cells. Warning! The translational symmetry is lost. Another application may be to move a set of atoms given in Cartesian coordinates to an arbitrary cell.

Print transformations up to now

OTM n ;

All transformations performed until now (TZ, TZUR, ...) are combined and displayed in form of a TZUR-plus a TZ-command. This is e.g. useful, if for SHELX-calculations the transformation matrix is needed, which is supposed to be entered with the SHELXL-command HKLF. $n \neq 0$ resets to the initial values. Default is $n = 0$.

Example: The data-set for the compound $La(NSO_2)_3$ was recorded in the monoclinic crystal class, and the structure has been solved and refined monoclinically. With SFND, a three-fold axis has been detected, followed by a cell idealization with ZIDL. Repeating SFND followed by RGS has yielded the space group $R\overline{3}m$ (166). However, RGS transforms to the primitive rhombohedral cell, which after importing of the RGS-results with the command 'RTHO' can be transformed to the hexagonal setting. 'OTM' produces now (in the form of a 'TZ' command) the matrix, which has to be entered with the HKLF-command. Caution: After using RZ, these data are invalid.

Transform to Subgroup

TUG n ;

If a structure in a certain space group is to be transformed into a subgroup, several steps have to be performed. The structure is made triclinic followed by an origin shift and a transformation of the axes (if necessary). If the cell has become enlarged, centering symmetries have to be introduced, and the structure has to be made triclinic again. Finally the target space group is introduced and the redundant atoms removed from the parameter list.

For all space groups in standard setting - i.e., for the monoclinic space groups (3 - 15) those with unique axis b, and for space groups where there are two settings (48, 50, 59, 68, 70, 85, 86, 88, 125, 126, 129, 130, 133, 134, 137, 138, 141, 142, 201, 203, 222, 224, 227 and 228) the one having the center at the origin - the maximal non-isomorphic subgroups and one route for this transformation are already stored and available. If a zero is entered for n [0], these subgroups will be listed, and no action performed. If $n > 0$ is given, n should be the space group number of the target subgroup, which may have an extension of two digits. If n is preceded by a minus sign, n is interpreted to be the position in the list.

Transform isomorphically

TIM $s_{11}s_{21}\ldots s_{33}$;

When transforming isomorphically (i.e. to the same space group but enlarging the unit cell) the same steps are needed as for transforming to a subgroup. Since the transformation in general depends on the details of the problem, no table is given. One has to enter a 3×3-matrix as explained at 'TZ'.

Search path to subgroup

SPUG *spg1 spg2 n opt* ;

When investigating relationships between structures, frequently the problem arises to find a path from one spacegroup *spg1* (supergroup) to a different space group *spg2* (subgroup) which is *not* a maximal non-isomorphic subgroup. Note that one has to enter space group *numbers*, not symbols. 'SPUG' uses the maximal non-isomorphic subgroups as given in the Int. Tables to find paths in maximal n [4] steps. Maximal allowed value for n is 8. While executing this command the number of steps is incremented $1, \ldots, n$. If *opt*

= 0 (default) is given, the process stops if one or more solution(s) are found for a certain number of steps ($\leq n$). If $opt \neq 0$ was given, the search is continued until n is reached.

Example:

```
>spug 225 15
(  1)  Fm-3m      225  R-3m     16601  C2/m      1202  C2/c      1502
(  2)  Fm-3m      225  R-3m     16601  R-3c     16701  C2/c      1502
```

Which space group number?

WRGN *spg* ;

Enter the (short) symbol of a space group *spg*, and the space group number will be printed.

Search for common supergroups (aristotype)

SGOG *spg1 spg2 n opt* ;

It is tested whether there exist space groups that are supergroups to both *spg1* and *spg2*. The parameters n [4] and *opt* [0] refer to the command 'SPUG' which is used to perform this search. If supergroups exist, the results will be printed.

Recall cell

RZ ;

When a new cell is entered or a transformation of the unit cell is performed, the old lattice constants are saved. 'RZ' restores the old unit cell by overwriting the current values. Warning! No transformation of parameters or symmetries will be performed.

Import translational symmetries

ITS $n_1, n_2, tol, opt_1, opt_2$;

Directly after the commands 'SFND' or 'TSZ', the translational symmetries that have been detected can be dealt with using the command 'ITS'. They are added as centerings to the symmetry list, and afterwards the parameters are reduced using the command RPSY. For the necessary entries in 'ITS' see 'RPSY'. Finally the cell is reduced to a primitive cell ('RDZ').

Reduce cell

RDZ ;

The command 'RDZ' is used to reduce the cell, in particular centerings are removed. The primitive reduced cell is determined and the structure transformed to this cell, which no longer contains a centering. This cell is determined as follows: The two shortest non-parallel axes are selected, which enclose an angle that is as close as possible to 90°. These are identified with the directions of the x-axis and y-axis of the free coordinate system. Finally, a third vector is selected by demanding that it is the shortest one that has a positive z-component. If necessary, one transforms the cell in addition such that in the triclinic case all angles are smaller or larger than 90°.

This assumes that the "atom" 2 in the parameter list has the coordinates (0,0,0). In addition, the centerings must be present as symmetries in the symmetry list. If these conditions are not fulfilled, a preliminary step is necessary: make the structure triclinic and enter the centerings via the appropriate lattice-type command.

Rhombohedral to hexagonal obverse

RTHO ;

With this command, a rhombohedral setting in the hexagonal (trigonal) obverse sense can be achieved. The following commands are executed automatically:

```
TZ  -1  0  1   1  -1  0   1  1  1
GTY Q
```

Be careful! This command does not check, whether the original cell corresponds to a rhombohedral setting.

Transform Symmetries

TRSY $s_{11}, s_{21}, s_{31}, s_{12}, s_{22}, s_{32}, s_{13}, s_{13}, s_{33}$;

This command works analogously to 'TZ', but only transforms the symmetries.

Example: A structure is created in the space group Pbnn. According to the International Tables (4. ed, p. 58), this is space group Pnna (No. 52) in the variant (cab). The symmetries in this setting are found using the command: `RG 52; TRSY 0 0 1 1 0 0 0 1 0`.

Reduce parameters according to symmetries

RPSY $n_1, n_2, tol, opt_1, opt_2$;

If there are atoms in the parameter list which occupy the same position within a tolerance *tol* [0.5] (in Å) after symmetries have been applied, the duplicate ones may be removed by this command. This reduction starts with the atom n_1 [3] and ends at the atom n_2 [last atom in parameter list]. Two atoms are treated to be identical, if the smallest distance between two corresponding codes is less or equal *tol*. If *tol* is given as a negative number, it will be interpreted as $tol \times tol_s$ (for tol_s, c.f. 'SFND' p. 78).

If opt_1 [−1] is set to zero, only atoms will be removed. If $opt_1 = 1$, in addition the center of mass of the two atoms is calculated and these coordinates overwrite those of the atom remaining in the list. In this way an atom located near a special position may be moved there. $opt_1 = -1$ is similar to $opt_1 = 1$, but here the option 'AGLP' is active instead of 'GLP'. This is preferable, if distances of less than 0.01 Å should be idealized. If opt_2 [−1] is set to a value greater zero, all symmetries except the first one (the identity) are hidden. Thus, only duplicate atoms will be removed. If opt_2 is given as a negative number only atoms having the same name are removed.

The command is especially useful in the context of cell transformations. One usually proceeds as follows: Using 'MTRI', a transformation takes place to space group P1, i.e. a "triclinic set" of atoms are brought into the parameter list. Then the cell is transformed (see 'TZ' and 'TZUR'), and new symmetries are introduced. Using 'RPSY' after these steps, the parameters will be reduced to an asymmetric set.

Idealize parameters according to symmetries

IPSY n_1, n_2, tol ;

After introducing new symmetries, it may happen that some atoms only lie near a special position instead of exactly on it. Using 'IPSY', all atoms in the range n_1 [3] to n_2 [last atom] in the parameter list are selected, and equivalent atoms in a sphere of tol [0.5] Å are searched for, and their center of mass is calculated. The result replaces the coordinates of the atom.

Example: In a structure (cell constants less than 10) an atom has the coordinates 0.01,0.02,-0.01. If space group Pmmm is introduced, 'IPSY' will idealize the coordinates of this atom to 0,0,0.

Condense cluster

CNCL $n_1, n_2, tol, opt_1, opt_2$;

Suppose, one has a triclinic structure where atoms are clustering in order to emulate a special position. These atoms can be condensed to one atom using 'CNCL'. n_1 [3] and n_2 [last atom] define the part of the parameter list where this action has to take place, *tol* [0.5] is the action radius in Å. opt_1 and opt_2 do have the same defaults and meanings as with 'RPSY' (see there).

The "condensation" uses the following strategy: The parameter list will be reordered in order to obtain blocks of atoms having the same name. Any (not yet removed) atom of a block is the starting point of a 'AUW' command: `AUW` n_1 n_2 n_1 n_2 *tol*. No more than 48 atoms should be found. The center of mass is calculated, and the result replaces the coordinates of the starting atom. All other atoms found are deleted.

Make triclinic

MTRI n_1, n_2 ;

From the atoms in the parameter list beginning at n_1 [3] and ending at n_2 [last atom] a complete set of atoms is constructed by applying all current symmetries and selecting those atoms, which lie in the unit cell, i.e. $0 \leq x, y, z < 1$. This set replaces the original atoms. The space group is set to P1, and the code list is cleared.

Files

Besides entering data using the keybord one may want to read and write data from and to files. E.g., a picture may be saved with the command 'PUTC' by creating an ASCII file, which contains the information needed to reproduce the picture. This file can also be manipulated with an editor. Using 'GET' such a file is read in again.

Input / output units

EAE *ntin, ntout, ntpch, ntscr, nxr1, nxr2, nbilf, ntlog, ntdisp ;*

Every file which is read or written to is assigned a number. The default numbers are given below. 'EAE' serves to alter them.

ntin	Standard input	(5)
ntout	Standard output	(6)
ntpch	Standard punch	(7)
ntscr	Standard scratch	(8)
nxr1	XRAY / SHELXL data	(1)
nxr2	Auxiliary	(2)
nbilf	KPLOT commands (PUT(C))	(12)
ntlog	Setup	(4)
ntdisp	Standard error	(6)

Usually there is no need to change these numbers, except *ntin* and *ntout*. If *ntin* is set to a number, an attempt is made to read from the file assigned to that number. If no file is open and assigned to this number, the program asks for a name. If *ntout* is changed, the output may be redirected to a file in the same manner.

Read a file

GET resp. IMP *name* ;

Example: `GET NACL.DAT` The file specified with *name* must contain KPLOT commands. It is advisable to begin this file with the command 'NDLG' and end with `DLG ; EAE 5` (or `CLSE; DLG`) on the last line to pass control back to the keybord.

In all versions except for IBM the program asks for a file name if the numbers for *ntin* and/or *ntout* have been changed, and the new numbers do not point to files currently open. A special meaning has the *name* '.' (dot). If given as name, the last name in the register is used. See also 'OFN'. The difference between 'GET' and 'IMP' is that with 'GET' the given file name is stored but not with 'IMP'.

Write a file

PUT resp. PUTC *name* ;

All information available concerning the current structure will be written to the file *name* using the logical number *nbilf*. The difference between 'PUT' and 'PUTC' is that with 'PUTC' *nbilf* is closed in case it was open, and after writing the file is closed again. Using 'PUT' information may be appended.

Save Cartesian coordinates

PUKC *name* ;

Analogous to 'PUTC', a file called *name* is written which *only* contains atoms of the code list, but having Cartesian (free) coordinates instead of relative coordinates.

Write a logfile

LOG [*no,*]*name* ;

When working in dialogue mode one may want to keep a copy of all entries and all responses of the program on a file. When giving the command 'LOG', such a file with the name *name* is created, or overwritten if it already exists. The logical number *no* is set by default to $ntlog = 4$ and may not be used otherwise. If $no = 0$ is specified, logging is stopped, but the logfile remains open. To resume logging, one would enter `LOG;`. Commands executed while reading a file and while processing a macro are not written to the logfile.
Example: `LOG SESSION.LOG`

Open file (not IBM)

OPEN *nr, name* ;

In all versions except for IBM (CMS or MVS) files are handled as follows: When the program is started,

only the files with the numbers *ntin* (5) and *ntout* (6) are opened. If other files are to be used during the execution of a command, the program will ask for the name of the particular file at the first attempt to read it or write to it. Alternatively, the files may be opened using the command 'OPEN' before working with them.

Example: A file with the name NACL.DAT contains the commands for creating the drawing of the NaCl structure. This file may be processed by:

```
OPEN 1 NACL.DAT
EAE 1
```

All numbers *nr* between 1 and 99 except 5, 6, and 20 are allowed. The default *name* is FOR0*xx*.DAT where *xx* is the given number. A special meaning has the name '.' (dot). If entered where a name is expected, the last name used with the commands 'GET' or 'PUT' or 'PUTC' is substituted. See also 'OFN'.

Show numbers of files open (not IBM)

OOPN ;

All file numbers currently open will be shown. No new files can be opened using these numbers.

Close file (not IBM)

CLSE *nr* ;

The file that is currently open under the number *nr* will be closed. A different file may be opened now under this number.

Show current file name

OFN ;

The file name used last with 'GET' or 'PUT' will be shown.

Rewind

RW *nr* ;

The unit allocated to the number *nr* will be rewound. If one had written to this unit, the content (of the file/buffer) will be lost. Default for *nr* is the value for *ntscr* (see 'EAE').

Writing KPLOT-commands to file

In order to save the current KPLOT commands, a variety of commands are provided. Most of them begin with the letter 'Q' and are similar to the 'O' commands, but the info is written to the file having the logical number *nbilf*. The most frequently used commands are 'PUT' and 'PUTC' which will write (almost) all information currently available and thus, e.g., can be used to save a drawing. Using 'PUTC' followed by a file name *name*, it is assumed that a new file is to be created. Therefore, *nbilf* is closed (if currently open), then the file *name* is written and closed. This file is called Q-file below.

Write text to Q-file

QK '*text*' ;

Free text, e.g. commands may be written to file. Note that quotes inside the text must be specified by two quotes, i.e. the command

```
QK 'T ''Structure of NaCl'''
```

will produce the line:

```
T 'Structure of NaCl'
```

The rest of the 'Q' commands are summarized in the following table.

QT	Title
QZ	Cell
QPK	Plot commands
QORT	Three DK commands for orientation
QUP	Defining the origin of the coordinate system
QC	Codes in form of ACIM commands
QZB	Size of the drawing field
QSY	Symmetries in symbolic form
QA	Atoms (parameters)
QTF	Temperature factors
QSF	Data control blocks for shading
PUT	All data
PUTC	Like PUT, but close afterwards

Rewind Q-file

QRW ;

The file having the logical number *nbilf* is closed and then opened again, i.e. re-set to the starting point.

Backspace Q-file

QBS n ;

The file having the logical number *nbilf* is backspaced n logical records. The goal is to selectively correct records by overwriting them.

Set number for Q-file

QAE n ;

This command is equivalent to `EAE * * * * * *` n and just a more convenient way to set the number *nbilf* to n.

Write free coordinates to Q-file

QFK n_1, n_2 ;

The free coordinates and the names of atoms which are in the code list (not parameter list!) from n_1 to n_2 are written to the file *nbilf* using the format `(3F15.6,4X,2A4)`. Defaults: $n_1 = 1$ and $n_2 =$ last code.

Control 'ZF' output

ZFQ resp. NZFQ ;

If the option 'NZFQ' is in effect (default), no 'ZF' command is written to the Q-file with 'PUT' or 'PUTC'. 'ZFQ' cancels 'NZFQ'.

Save macros

MSAV n ;

When giving the option 'MSAV' with $n \neq 0$ [initial value 1] then by executing 'PUT' or 'PUTC', all macros currently defined are included in the output file.

Group list of codes (mouse list)

Beside the code list, there is another list where codes may be stored (more or less temporarily). This list can be thought of in some way as a "notice board". It is called group list of codes or mouse list (see below). Codes which are stored here are used in the context of many commands, if a zero is specified in those commands where one or many codes are expected. Up to 120 codes can be stored. Note: whenever codes are going to be stored in the mouse list by 'LGC' or 'M' or 'CTGC', all codes currently in the list are removed.

Load group of codes

LGC $c_1, c_2, \ldots, c_n$;

The mouse list is cleared and the codes specified added. The input of the codes occurs in the same way as with 'ACIM' (see p. 19).

Load group of codes using the mouse

M ;

If a mouse is supported, the entries in the list of group of codes can be created by clicking on the atoms on the screen. After the input of 'M', a cross hair cursor is shown. Now atoms may be clicked on in the drawing currently on the screen. If the left mouse button is pressed, the code will be added and the atom is marked by a cross. Pressing the right mouse button terminates the selection. In contrast to the code list, the mouse list may contain the same code several times.

Add to group list of codes

AGC $c_1, c_2, \ldots, c_n$;

This command works like 'LGC', but here the group list is not cleared at the outset, i.e. the codes specified will be added to the current mouse list.

Add to group list of codes using the mouse

AGCM ;

Works like 'M', but here the group list is not cleared at the outset, i.e. the codes specified will be added.

Add group of codes as atoms

AGCA ;

Codes which are currently in the group list are converted to atoms and added to the parameter list.

Add group of codes to code list

AGCC v, opt ;

The codes which are currently in the group list (mouse list, see 'LGC') will be added to the code list if not already present. In addition a translation will be performed on the new codes in the code list, if v is not 555 (default), e.g. $v = 565$ leads to a translation in direction b. If *opt* [0] is unequal zero, the codes in the group list will also be shifted.

Color pointer for group of codes

FGC n ;

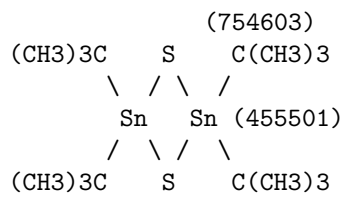
Those atoms in the parameter list to which the codes of the group list (mouse list, see 'LGC') point, will be assigned the color pointer n (default: 0). This command is useful in connection with 'SFRB' and 'SRTP'.

Add to group list of codes a tree

AGCT c_1, c_2 ;

The codes c_1 and c_2 will be brought to the group list of codes to positions 1 and 2. Then beginning with c_2 the codelist is searched for atoms which are connected to c_2 according to the 'VB' commands, i.e. a bond would be drawn if 'VB' were executed. If found they are added to the group list unless already present. All new atoms found in this way are used as origins in the same way until no longer new atoms detected. Defaults are: $c_1 = c_2 = 0$. The atom c_1 is *not* used as origin. If found by the search process, a warning will be issued.

Example: The following molecule is assumed to be in the code list.



The numbers in brackets are the codes of the Sn- and of the C-atoms. The command `'AGCT 4 754603'` brings to the mouse list the Sn atom and the t-butyl group connected to it. Usually one will click on both "starting" atoms, and then enter: `AGCT;`. The goal of this command is to prepare a group for the command 'VDGC'.

Codes to group list of codes

CTGC n_1, n_2 ;

The group list of codes is cleared and codes from the code list having numbers in the parameter list between n_1 (default: 3) and n_2 (default: last atom) are copied into the mouse list if space is available.

Print group list of codes (full)

OGC resp. OGCF ;

The content of the group list of codes is printed; a short form is used with 'OGC' i.e. only the codes are printed. If 'OGCF' is used, the output is more detailed, i.e. the name, extension, triclinic coordinates, and free coordinates of the codes are also given.

Ortep

Starting with KPLOT version 7.0 some parts of the program ORTEP are implemented. Those parts in ORTEP dealing with the automatic labeling of atoms and bonds are not implemented.

Ortep mode on/off

ORTP n ;

If for n one enters 1, those atoms that have anisotropic temperature factors are drawn as ellipsoids. The parameters for the representation of the ellipsoids are stored in the 'ATF' control blocks (see there and 'EPAR'). Bonds are drawn in the ORTEP mode, if for both atoms to be connected temperature parameters are available. To turn off the ORTEP mode, set $n = 0$.

Convert radii to temperature factors

CVRT n_1, n_2 ;

In order to represent atoms that do not have anisotropic temperature factors in ORTEP style, their radii are converted into pseudo isotropic temperature factors using 'CVRT'. Atoms having numbers in the parameter list between n_1 and n_2 (defaults: 3 resp. last atom), which do not have anisotropic temperature factors yet, are assigned values r/s_2 where r is the radius of the atom and s_2 the probability factor (see 'WFKT'). This is done in order to preserve the KPLOT radius r while drawing.

Ellipsoid parameters

EPAR $nr, type, a_0, a_1$; or EPAR $nr, 5, n_1, n_2, n_3, n_4, a_0, a_1$;

Ellipsiod parameters are stored in ATF blocks as numbers $n_1, \ldots, a_2$. Using 'EPAR' these numbers are set in such a way that they correspond to the ORTEP instructions $71x$ with $x = type$. If $type = 5$ is given, $n_1, \ldots, n_4$ can be defined by the user. The meaning is as follows:

		Front of ellipsoid
n_1:	$= 0$	no ellipsoid components
	$= 1$	boundary ellipse only
	$= 3$	principal ellipses only
	$= 4$	boundary and principal ellipses
		Back side of ellipsoids:
n_2:	< 0	solid line back side
	$= 0$	back side omitted
	$= 3$	4 dots on back side
	$= 4$	8 dots on back side
	$= 5$	16 dots on back side
	$= 6$	32 dots on back side
		Principal axes and shading:
n_3:	$= 0$	no forward axes shading
	$= 1$	forward principal axes only
	$= n$	forward principal axes and $(n-1)$ line shading
		Reverse principal axes
n_4:	$= 0$	no reverse axes
	$= n$	dashed reverse axes with n dashes

It is possible to vary the thickness of the boundary ellipse line by making the line width a function of z, the height of the atom above the drawing plane. This option is usually used when drawing the boundary only, but will work for all types of lines. Entries are put in as a_0 and a_1 to specify the coefficients of

$$d_r(z) = a_0 + a_1 z$$

where d_r is the increase in radial dimension to be added to the width of the single pen line, a_0 equals d_r for an atom at $z = 0$, and a_1 is the rate of increase in radial size with z.

Example: It is assumed that the z coordinates of the atoms of the model are in the range ± 5 cm. The thickness of the line is 0.2 mm. If the thickness of the line of the ellipse nearest to the viewer is to be five times the thickness of the farthest ellipse, one has to set:

$d_r(-5) = 0, d_r(5) = 0.02(5 - 1) = 0.08,$
resulting in $a_0 = 0.04$ and $a_1 = 0.008$.

The program stepwise widens the line by radially increasing the width in increments of d, which may be set by 'DISP'. Initially d is zero; therefore d must be defined before using this feature.

Displacement value

DISP d ;

Displacement d (default: 0) needed, if the thickness of the ellipsoids is to be increased (see above).

Probability value

WFKT s_2 ;

The principal axes of the ellipsoids are scaled by s_2. This feature is used to represent the atoms as ellipsoids having a contour surface of equal probability density. The default value of s_2 is 1.54 corresponding to 50% probability. For further details see "ORTEP: A FORTRAN THERMAL-ELLIPSOID PLOT PROGRAM FOR CRYSTAL STRUCTURE ILLUSTRATIONS" (Carroll K. Johnson, Juni 1965).

Rotate coordinate system for ORTEP-stereo plot

DKOS a, w ;

In the program ORTEP there exists (beside the free coordinate system, here called working system) a further coordinate system called reference system. Usually these two systems are identical. But when generating a stereo plot there is the problem that certain details must be identical in both plots. Therefore, these details are provided using the reference system, all other drawing commands using the free (Ortep: working) system. Only the working coordinate system, not the reference system, will be rotated around the axis specified (enter $a = 1$ for x-axis, $a = 2$ for y-axis, $a = 3$ for z-axis). The rotation angle w has to be given in degrees. Default value is the angle calculated last by 'W' or 'AW'.

Table for critical values for probability ellipsoids
of a trivariate normal distribution

p	s_2	p	s_2	p	s_2
0.01	0.3389	0.41	1.3842	0.81	2.1824
0.02	0.4299	0.42	1.4013	0.82	2.2114
0.03	0.4951	0.43	1.4183	0.83	2.2416
0.04	0.5479	0.44	1.4354	0.84	2.2730
0.05	0.5932	0.45	1.4524	0.85	2.3059
0.06	0.6334	0.46	1.4695	0.86	2.3404
0.07	0.6699	0.47	1.4866	0.87	2.3767
0.08	0.7035	0.48	1.5037	0.88	2.4153
0.09	0.7349	0.49	1.5209	0.89	2.4563
0.10	0.7644	0.50	1.5382	0.90	2.5003
0.11	0.7924	0.51	1.5555	0.91	2.5478
0.12	0.8192	0.52	1.5729	0.92	2.5997
0.13	0.8447	0.53	1.5934	0.93	2.6571
0.14	0.8694	0.54	1.6080	0.94	2.7216
0.15	0.8932	0.55	1.6257	0.95	2.7955
0.16	0.9162	0.56	1.6436	0.96	2.8829
0.16	0.9386	0.57	1.6616	0.97	2.9912
0.18	0.9605	0.58	1.6797	0.98	3.1365
0.19	0.9818	0.59	1.6980	0.99	3.3682
0.20	1.0026	0.60	1.7164	0.991	3.4019
0.21	1.0230	0.61	1.7351	0.992	3.4290
0.22	1.0430	0.62	1.7540	0.993	3.4806
0.23	1.0627	0.63	1.7730	0.994	3.5280
0.24	1.0821	0.64	1.7924	0.995	3.5830
0.25	1.4045	0.65	1.8119	0.996	3.6492
0.26	1.1200	0.66	1.8318	0.997	3.7325
0.27	1.1386	0.67	1.8519	0.998	3.8465
0.28	1.1570	0.68	1.8724	0.999	4.0331
0.29	1.1751	0.69	1.8932	0.9991	4.0607
0.30	1.1932	0.70	1.9144	0.9992	4.0912
0.31	1.2110	0.71	1.9360	0.9993	4.1256
0.32	1.2288	0.72	1.9580	0.9994	4.1648
0.33	1.2464	0.73	1.9804	0.9995	4.2107
0.34	1.2638	0.74	2.0034	0.9996	4.2661
0.35	1.2812	0.75	2.0269	0.9997	4.3365
0.36	1.2985	0.76	2.0510	0.9998	4.4335
0.37	1.3158	0.77	2.0757	0.9999	4.5943
0.38	1.3330	0.78	2.1012	0.99999	5.0894
0.39	1.3501	0.79	2.1274	0.999999	5.5376
0.40	1.3672	0.80	2.1544	0.9999999	5.9503

Building structures

In this section, commands are described dealing with geometrical aspects of crystal structures. While during a typical crystal structure analysis coordinates of atoms are found using e.g. Fourier methods, here coordinates are determined by known geometrical relations.

For the calculation of points in space, a point register is provided. After each calculation, this register contains the triclinic and the free coordinates of a point. The point register may be thought of as the storage of a desk calculator containing the results of previous calculations. The content is volatile.

In addition to commands manipulating the point register, there are other commands working on the parameter list directly. When executing these commands, the point register usually will be used and its content is therefore overwritten and destroyed (volatility). This is also the case, when the free coordinate system is redefined.

Most commands dealing with the parameter list directly, e.g. 'DG' (rotate group), either produce new atoms or overwrite the coordinates of the old ones. When new atoms are generated, they will be added to the parameter list at the end. This choice is controlled by the options A and R, add and replace, respectively. If no option is given in the input of a specific command, a default value is used which may be changed using 'GOPT'. Initially, this value is set to R.

Group option

GOPT A resp. R ;

The value of the default option is set (A=add, R=replace) which will be used with most group commands, when the input is omitted. The initially setting is R.

Point with free coordinates

P x, y, z ;

A point having the coordinates x, y, z in Angstroms is placed into the point register. The previous value in the register is destroyed.

Point with free coordinates shifted

PREL x, y, z ;

The values x, y, z (defaults: 0,0,0) are added to the free coordinates currently on the point register. The result replaces the old values.

Point in triclinic coordinates

PT x, y, z ;

The point having triclinic coordinates x, y, z, i.e. coordinates with respect to the lattice, is loaded into the point register.

Load point

PL c ;

The coordinates of the point having the code c are loaded into the point register.

Add point as atom

AA s, e, r, n, p ;

The point currently in the point register will be stored in the parameter list with the name s, extension e, radius r, at the position n with the color pointer p. Default for n is last atom+1, i.e. the atom is added, otherwise the atom at the position n is replaced. The default values are taken over from the atom that is being overwritten; otherwise, they are s = UNDE, e = FINI, $r = 0.3$, $p = 0$.

Remove point(s) from the parameter list

EPL resp. DELA n_1, n_2 ;

The specified range from n_1 (no default) to n_2 (default: n_1) will be removed from the parameter list. If only one number is specified, only that atom will be removed. Note that usually a renumbering of the atoms in the parameter list takes place, unless n_2 specifies the end of the parameter list. The codelist and some plot commands are updated taking care of this renumbering.

Transform point to reciprocal system

PTTR ;

The triclinic coordinates of the point currently in the point register will be replaced by the coordinates with respect to the reciprocal system. The free coordinates are not changed. This command may be useful when dealing with twinned crystals.

Print point register

OP ;

The triclinic and the free coordinates currently in the point register are printed.

Enforce length

FL d ;

O denotes the origin of the free coordinate system. The point P, which is currently in the point register, is moved along the direction O $\rightarrow$ P such that the distance $|\mathrm{O} \rightarrow \mathrm{P}|$ equals d. Default for d is the last value computed using the commands 'L' or 'AL' (initial setting: 0).

Rotate point

DP a, w ;

The point currently in the point register will be rotated around the axis a ($1 = x$, $2 = y$, $3 = z$, no default) of the free coordinate system about w degrees . The rotation is counter-clockwise when seen from the top of the axis. Default value for w is the last value calculated by 'W' or 'AW' (initial setting: 0).

Rotate a point around a vector

DPV c_1, c_2, w ;

The point currently in the point register is rotated around a vector $c_1 \rightarrow c_2$ (defaults for the codes: 0,0) about w degrees. The rotation is performed counter-clockwise, with the vector pointing towards the observer. Default for w is the last angle that has been computed with the command 'W' or changed with the command 'AW'.

Shift point

VP c_1, c_2, d_a ;

The point in the point register is shifted in the direction of the vector $c_1 \rightarrow c_2$ given by the codes c_1 and c_2. The amount of shift is d_a in Angstroms. Default for d_a is the length last calculated using 'L' or 'AL' (initial setting: 0).

Shift point (vector units)

VPV c_1, c_2, d_v ;

The point in the point register is shifted in the direction of the vector $c_1 \rightarrow c_2$ given by the codes c_1 and c_2. The amount of shift is d_v times the length of the vector. Default for d_v is 1.

Center of gravity

SP $c_1, g_1, c_2, g_2, \ldots, c_n, g_n$;

One has to specify codes $c_1, \ldots, c_n$ and weights $g_1, \ldots, g_n$. The g_i (default values: 100) must be given as integers.

Center of gravity via codes

SPC n_1, n_2 ;

The weighted center of gravity is calculated using all atoms in the code list having numbers in the parameter list between n_1 and n_2 (defaults: 1, last atom). The radii of the atoms selected are taken as weights.

Center of gravity via group of codes

SPGC *opt* ;

The non-weighted (*opt* = 0, default) or the weighted (*opt* = 1) center of gravity is calculated of all atoms being in the group list of codes (mouselist, see 'LGC'). For *opt* = 1, the radii of the atoms are taken as weights.

Point by three distances

PDA $c_1, c_2, c_3, l_1, l_2, l_3$;

A point in space is determined and its coordinates stored in the point register. The point has the distance l_1 from the point with the code c_1, the distance l_2 from the point with the code c_2, and the distance l_3 from the code c_3. Due to the fact that in general there are two solutions if the point is not in the plane defined by c_1, c_2, c_3, that point is selected from which the codes c_1, c_2, c_3 appear in a counter-clockwise order. If no solution exists, an error message is issued. The content of the point register has no meaning in this case. Default values for the l_j are the distance calculated last by 'L' or 'AL' (initial setting: 0). If for l_2 a negative number or zero is entered, l_2 is replaced by l_1, and analogously for $l_3 \leq 0$, l_3 by l_2.

Point by distance and two angles

PAW c, c_1, c_2, l, a_1, a_2 ;

A point P in space is determined and its coordinates stored in the point register. This point has the distance l from the point with the code c. Furthermore the angles $c_1 - c - P$ and $c_2 - c - P$ equal a_1 and a_2, respectively. Due to the fact that in general there are two solutions if the point is not in the plane containing c, c_1, c_2, that point is selected from which the codes c, c_1, c_2 appear in counter-clockwise order. If no solution exists, an error message is issued. The content of the point register has no meaning in this case. Default value for l is the distance calculated last by 'L' or 'AL' (initial setting: 0) Default value for the a_j is the angle calculated last by 'W' or 'AW' (initial setting: 0). If one enters a negative number or zero for a_2, a_2 is replaced by a_1.

Point via two distances

PZA resp. PZAA c_1, c_2, c_h, l_1, l_2 ;

A point in space is determined and its coordinates stored in the point register. This point has the distances l_1 and l_2 from the points with the codes c_1 and c_2, respectively. Furthermore, the computed point lies in the plane containing c_1, c_2, c_h. Due to the fact that in general there are two solutions for this problem, the point nearer (farther) to c_h is selected when using 'PZA' ('PZAA'). Default values for the l_j are the distance calculated last by 'L' or 'AL'(initial setting: 0). If for l_2 a negative number or zero is entered, l_2 is replaced by l_1.

Point via vector and distance

PVA c, c_1, c_2, d ;

A point in space is determined and its coordinates stored in the point register. This point has the distance d from the point with the code c, and lies on the axis defined by the vector $c_1 \rightarrow c_2$, given by the codes c_1 and c_2: $c(\lambda) = \lambda(c_2 - c_1) + c_1$. Due to the fact that in general there are two solutions for this problem, the point with the larger λ is chosen. Default value for d is the distance calculated last by 'L' or 'AL' (initial setting: 0).

Point via plane and two distances

PEZA $c_1, c_2, d_1, d_2, c_{ur}, c_{x1}, c_{x2}, c_{y1}, c_{y2}$;

A point in space is determined and the coordinates stored in the point register. This point fulfills the following conditions:

- The distance from the point having the code c_1 is d_1 (default: last calculated length);
- the distance from the point having the code c_2 is d_2 (default: last calculated length; if d_2 is set to zero, d_2 is replaced by d_1);
- the point lies in the x-y-plane of the coordinate system defined by the codes $c_{ur}, \ldots, c_{y2}$ (see 'K');

Due to the fact that in general there are two solutions for this problem, that point is chosen which is nearer to c_{ur}. If $c_{ur} = 0$ is entered, the current free coordinate system is used.

Point of intersection vector - vector

SVV $p_1, p_2,\ q_1, q_2$;

The point having the minimal distance from the two lines containing the vectors $p_1 \rightarrow p_2$ and $q_1 \rightarrow q_2$, defined by the codes p_1,p_2 and q_1,q_2, is calculated and stored in the point register. If the lines intersect, the intersection point is given.

Intersection point vector - plane

SVE c_1, c_2 ;

The intersection point of a vector and a plane is determined and stored in the point register as follows: A least squares plane is fitted using the points stored in the group list of codes (mouse list). Now the intersection point with the axis containing the vector $c_1 \rightarrow c_2$ defined by the codes c_1 and c_2 is calculated. If zeros are entered for c_1 and/or c_2, the first two points in the group list define the vector and the remainig ones the plane. The points taken from the group list to define the vector are removed from the list. (See also 'PVE')

Point by mouse

PM ;

The cross hair cursor is moved to a certain location. If the left mouse button is pressed, the point at that position (x,y) is stored in the point register. The value in the register for the z coordinate remains unchanged.

Point via line and mouse

PGM c_1, c_2 ;

A point in space is determined and its coordinates stored in the point register. This point lies on the line defined by the codes $c_1 \rightarrow c_2$ and is as close as possible to the position clicked on. In this way a point may be defined by "pointing". This may be useful when drawing crystals (see 'EMP').

Length of a side of a triangle

LSWS a, γ, b ;

The length c of a triangle defined by the sides a and b, and the enclosed angle γ is calculated using the cosine formula and stored in the length register to be used as a default length in subsequent commands.

Copy group

CPG n_1, n_2 ;

All points located in the parameter list from n_1 to n_2 (n_1 and n_2 are included) [no defaults] are copied and appended (in the original order) to the end of the parameter list.

Rotate group

DG n_1, n_2, a, w, opt ;

All points stored in the parameter list from n_1 to n_2 (no defaults) will be rotated around the axis a (1=x, 2=y, 3=z) (free coordinate system) by w degrees (default: angle calculated last by 'W' or 'AW', initial setting: 0). The rotation is counter-clockwise when seen from the top of the axis selected. The resulting atoms are appended to the parameter list, if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Rotate group around a vector

DGV $n_1, n_2, c_1, c_2, w, opt$;

The codes c_1 and c_2 define a vector $c_1 \rightarrow c_2$. All points stored in the parameter list from n_1 to n_2 (no defaults) will be rotated around this vector by w degrees (default: angle calculated last by 'W' or 'AW'). The rotation is counter-clockwise when seen from the top of the vector. The resulting atoms are appended to the parameter list, if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Shift group

VG resp. VGV $n_1, n_2, c_1, c_2, d, opt$;

The codes c_1 and c_2 define a vector $c_1 \rightarrow c_2$. All points stored in the parameter list from n_1 to n_2 (no defaults) will be shifted in the direction $c_1 \rightarrow c_2$ by d Å with 'VG' (default: distance calculated last by 'L' or 'AL'; initial setting 0), or d times the length of the vector with 'VGV' (default here: $d = 1$), respectively. The resulting atoms are appended in the parameter list, if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Shift and rotate group

VDG $n_1, n_2, c_1, c_2, c_3, c_4, d, beh, opt$;

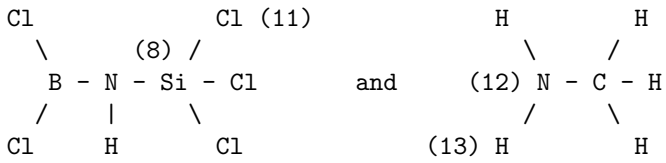
The goal of 'VDG' is to pin a fragment onto another fragment by shifting and rotating it. The group given by the numbers $n_1, \ldots, n_2$ in the parameter list is shifted and rotated in such a way that the "bond" $c_4 \rightarrow c_3$

falls on the "bond" $c_1 \rightarrow c_2$ (target). c_2 and c_4 are the "outer" atoms. After this operation, the distance (c_1, c_3) will be d Angstroms. If $d = 0$ is given, the distance will be taken from a list which may be specified by 'VDGD'. If opt = R is entered, the coordinates of the group itself will be overwritten, otherwise (opt = A, default) new atoms will be generated and in addition the new group also added as codes. The redundant atoms c_2 and c_4 are removed from the code list and also, if $beh = 0$ (default), from the parameter list. Else, if $beh \neq 0$, they are not removed from the parameter list. Default for c_1 and c_2 is 0,0.

In order to be able to make some corrections, all codes necessary for 'VDGC' are stored in the mouse list. The first two will be c_3 and (the new) c_1. Now one will be able using 'VDGC' to rotate the new group leaving c_3 fixed around the axis a (a=1, 2, or 3), or around the axis $c_3 \rightarrow c_1$ (a=4). Note that with 'VDG' opt = R must be specified, if a replacement of the group is desired. The value of 'GOPT' is *not* used here.

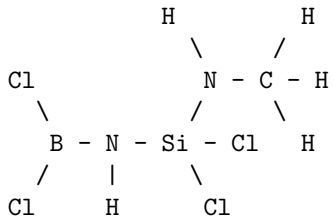
Example:

Assume the following molecules are in the parameter list and the labeled atoms have the numbers shown:



3 - 11 12 - 18

By `'VDG 12 18 8 11 12 13 1.6'` the second molecule will be joined together with the first molecule generating a new group. Atom no. 11 and the atom coming from no. 13 are deleted in the process.



To make a repetition of this process easier, the parameters used in 'VDG' are stored: Clicking on a new target-bond (inner to outer atom) with 'M' followed by 'VDG' without parameters, a new "condensation" will take place.

Data for 'VDG'

VDGD s_1, s_2, d ;

Due to the fact that distances for certain pairs of elements do not change when using 'VDG', they may be stored in a list. Up to 16 entries can be stored. s_1 and s_2 are the names, d the distance to be given. *Example:* `VDGD B N 1.47`

Print Data for 'VDG'

OVDD ;

The list generated by 'VDGD' commands will be printed.

Shift group of codes

VGC resp. VGCT x, y, z ;

The atoms in the group list of codes (mouse list) are shifted by the vector (x, y, z) in free coordinates with 'VGC', and by the vector (x, y, z) in triclinic coordinates with 'VGCT', respectively. Note that the coordinates of the atoms are changed to achieve a shift of the codes as specified. Each atom number is taken into account only once, even if found more than one time in the mouse list (the first code found for an atom number is used). The coordinates of the atoms in the parameter list are overwritten in this process; new atoms are not generated. The values for x, y, z are kept and used as defaults for the next application of 'VGC' or 'VGCT'.

Shift group of codes using the mouse

VM ;

This command works in principle like 'VGC'. In contrast, here the shift vector (x, y) is defined by two mouse clicks. The z-coordinate is always set equal 0, i.e. the shift vector is parallel to the plane of the screen. The plane where the move actually takes place is defined by the z-coordinate of the current point on the point register.

Example: Assume a coordinate cross has been defined by 'GNKO'. The numbers of the atoms are assumed to be $11, \ldots, 14$. This coordinate cross may be repositioned conveniently by loading the atoms into the mouse list (`LGC 11 -14`) and then (after 'VM') clicking first on the cross and then on the new position.

Rotate group of codes

DGC a, w ;

The atoms in the group list of codes (mouse list) are rotated w degrees around the axis a (1, 2 or 3; no initial setting) of the free coordinate system. Default value for w is the angle calculated last by 'W' or 'AW'. Note that the coordinates of the atoms in the parameter list are changed to achieve a rotation of the codes as specified. Each atom number is taken into account only once, even if found more than once in the mouse list (the first code found for an atom number is used). The coordinates of the atoms in the parameter list are overwritten in this process; new atoms are not generated. The parameters are kept and used as defaults for the next time.

Rotate group of codes around a vector

DGCV c_1, c_2, w ;

The atoms in the group list of codes (mouse list) are rotated w degrees around the vector $c_1 \rightarrow c_2$, specified by the codes c_1 and c_2 (no initial settings). Default value for w is the angle calculated last by 'W' or 'AW'. Note that the coordinates of the atoms are changed to achieve a rotation of the codes as specified. Each atom number is taken into account only once, even if found more than one time in the mouse list (the first code found for an atom number is used). The coordinates of the atoms in the parameter list are overwritten in this process; new atoms are not generated. The parameters are kept and used as defaults for the next time.

If 'MORE' is in effect (with a parameter > 0), the rotation matrix is printed. This may be useful when dealing with complicated twinning problems.

Apply factor directly / reciprocally to group of codes; or set length explicitly

FDGC, FRGC or FLGC c, f ;

All vectors are calculated, which connect the atom with the code c to the atoms in the mouse list. The lengths of these vectors are multiplied with the factor f or $1/f$ for the commands 'FDGC' and 'FRGC', respectively, and the atoms in the mouse list are moved accordingly. When using 'FLGC', the lengths of all the vectors are set equal to f. Default for f is the last value computed using the commands 'L' or 'AL' (initial setting: 0). Note: the coordinates of the original atoms in the parameter list are changed. If an atom is referred to several times in the mouse list, only the first code is used in the calculation.

Shift and rotate group of codes

VDGC $s, dir, w, a, c_1, c_2, plt, opt_1, opt_2, opt_3, m$;

This command is a built-in-macro containing several commands which can be used to move the group of codes within a structure. All input values are kept when leaving this macro except *dir* and a.

One shifts by s (initial setting: 0) in Angstroms in the direction *dir* [555]. The direction *dir* must be specified as a three digit number, analogously to the digits 3-5 of a designator code. E.g., 655 means $+x$ direction 545 $-y$ etc. Usually the shift is performed with respect to the free coordinate system. If for s a negative number is entered, the shift is replaced by its absolute value, but the shift vector is oriented with respect to the crystallographic axes (unit cell). The rotation about the angle w (initial setting: 0) is performed around the axis a [0] (1, 2, or 3) of the free coordinate system shifted into the atom having the code c_1 on the mouse list. If $a = 0$ no rotation takes place. If $a = 4$, the rotation axis is the vector $c_1 \rightarrow c_2$ as specified by the codes c_1 and c_2. Initial setting: 0,0.

After the shift or the rotation has been performed, the drawing is redone. *plt* can be given as EPU or STPU (initial setting: EPU); $opt_1 = 0/1$: do not/do calculate a new scaling factor before plotting (initial setting: 0); $opt_2 = 0/1$: single / multi-buffering mode, i.e. after EPU (or STPU) the command BUFN 0,0 is issued.

Usually one does not want to shift and rotate at the same time ($opt_3 = 0$, initial setting), i.e. the input of a rotation axis cancels shifting (sets $dir = 555$), and input of a three digit number cancels rotations (sets $a = 0$). To allow for a simultaneous shift and rotation, set $opt_3 \neq 0$.

After the drawing is finished, a user defined macro (the name of macro is the number m $1, \ldots, 9$; initial setting: 0, i.e. do not execute any macro) may be executed. Usually this option is used to do some calculations of distances and angles.

After performing the above steps the program waits for input. If the input is simply the return (or enter) key, the last operation is repeated. If 0, 1, 2, 3, or 4 is entered, the rotation axis is redefined. Upon entering a three digit number, the direction of the shift is redefined. Entering a minus sign (which may be followed by a number of an axis) inverts the rotation direction (and defines a new rotation axis). If a '9' is entered, the macro 'VDGC' is stopped, and the command 'D' is executed. Note that at this stage, the macro 'VDGC' may be re-activated by entering '9' again.

Mirror group

SPG n_1, n_2, a, opt ;

All points (atoms) stored in the parameter list from n_1 to n_2 (no defaults) will be mapped into their mirror images with respect to a plane orthogonal to the axis and containing the origin of the free coordinate system. This is achieved by calculating all free coordinates and then changing the sign of the coordinate specified by a [3] (1=x, 2=y, 3=z). The resulting atoms are appended to the parameter list, if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Displace group using coordinate system

KTG n_1, n_2, opt ;

All points stored in the parameter list from n_1 to n_2 (no defaults, but entries are kept and used as defaults next time) will be transformed when the free coordinate system is changed; they are "given a lift". The procedure is as follows: Define the free coordinate system in a suitable way; save the system ('SK'); define a new coordinate system: that implicitly defines a transformation; transform the selected group using this transformation with the command 'KTG'. Thus, the coordinates of the group are changed in such a way that the new free coordinates are (numerically) the same as those the atoms had with respect to the old (saved) coordinate system. The resulting atoms are appended to the parameter list if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Shift group like three points

VGDP $c_1, t_1, \ldots, c_3, t_3, n_1, n_2, opt$;

A transformation matrix will be calculated having the following property: The point with the code c_1 will be shifted to the point with the code t_1, the point with the code c_2 as well as possible (on the connecting line) to the point with the code t_2, and the point with the code c_3 as well as possible (in the same plane) to the point with the code t_3. An additional shift will be done making the centers of gravity of both systems identical. This transformation will be applied to all atoms of the parameter list having numbers from n_1 to n_2 (no defaults). If the letter 'R' is specified for opt, the new coordinates replace the old ones. Otherwise new atoms at the end of the parameter list will be created.

Apply factor directly or reciprocally to group

FDG or FRG or QDG $n_1, n_2, ccentr, f, opt$;

Using these commands, the group in the parameters list between n_1 and n_2 can be stretched or shrunk with respect to a center *ccentr*. The group of points (atoms) stored in the parameter list from n_1 to n_2 (no defaults) will be rescaled as follows: All vectors connecting the atom associated with the code *ccentr* (no default) with the atoms $n_1, \ldots, n_2$ in the parameter list are multiplied by f or $1/f$ when using 'FDG' or 'FRG', respectively. Default value for f is the length calculated last with 'L' or 'AL' (initial setting: 0) The resulting atoms are appended to the parameter list, if opt = A (initial setting) is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'. For the command 'QDG', the vectors are multiplied by f/(current default). (This is equivalent to applying 'FRG' followed by 'FDG'.)

Distort group

VZG n_1, n_2, a, f, opt ;

The group of points (atoms) stored in the parameter list from n_1 to n_2 (no defaults) will be (anisotropically) rescaled by multiplying only one coordinate (x-axis=1, y-axis=2, and z-axis=3; no default) with respect of

the free coordinate system by f. Default value for f is the length calculated last by 'L' or 'AL' (initial setting: 0) The resulting atoms are appended to the parameter list, if opt = A is specified. If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

'AFG' - Affine group

AFG n_1, n_2, opt ;

A matrix **A** is calculated from the entries in the mouse list that represents an affine mapping

$$p' = \mathbf{A} \cdot p$$

The mouse list contains pairs of codes $c_1, t_2, \ldots, c_n, t_n$ (at least four and up to sixty). The affine mapping minimizes the sum of the squares of the distances of the pairs, $\sum d^2(c_i \rightarrow t_i) = minimum$. In other words, the affine mapping should map the c_i as well as possible to the t_i. The mapping is applied to all atoms in the parameter list from n_1 to n_2 (no defaults, but the entries are kept). If the letter 'R' is specified for opt the resulting atoms replace the original ones. Otherwise they are added at the end of the parameter list. The default is taken from the setting of the 'GOPT' command.

Hint: Usually the mouse list is defined by using the mouse. If not all atoms can be seen when viewing the structure, the structure on the screen may be rotated and the procedure continued by 'AGCM' in order to add new pairs.

'AFGC' - Affine group completely

AFGC $n_1, n_2, t_1, t_2, d, dinc, itimes, opt, opt2$;

While the affine mapping calculated using 'AFG' is usually based on a selection among the atoms under consideration, one may want to use all atoms at the end. When using 'AFGC', an attempt is made to automatically find the pairs which map the atoms in the parameter list from n_1 to n_2 (*not* applying any symmetry operation) to target atoms in the parameter list from t_1 to t_2 using the following strategy: A sphere with the radius d [initially 0.4 Å] around each n_i is searched for target atoms. After every assignment cycle, it is tested which ones have not yet been assigned to be a target atom. If such one exists, it is assigned to be the image of n_i. After processing the list, it is checked whether all n_i have an image t_j. If not, d is incremented by $dinc$ [initially 0.2 Å] and the n_i without partners are again subjected to the same search procedure. This is repeated $itimes$ [Default: 20] times or until the assignment has been completed. If opt [-1] is given as a number $\neq 0$, the two atoms belonging to a pair must be the same (i.e. have the same name). If a positive number p is entered, p of the n_i do not need to have a partner. (E.g. the atoms that describe the outline of the cell which should only be transformed for visualization purposes, but do not need a matching partner.)

After a successful search, the affine mapping matrix is calculated analogously to 'AFG', and the mapping is performed. If the letter 'R' is specified for *opt2*, the resulting atoms replace the original ones. Otherwise they are added at the end of the parameter list. The default is taken from the setting of the 'GOPT' command.

Hint: When the distortion is too large, some atoms may be assigned incorrectly. However, this error is usually corrected when repeating the command.

Generate points

GP n, a, s, e, r, i_1, i_2 ;

A set of n (no default) points is generated by rotating the point currently in the point register around the axis a (x=1, y=2, z=3; no default) of the free coordinate system. The circle (360°) is divided by n, defining the rotation angle (= $360/n$). The points will be added to the parameter list, with the symbol s [UNDE], extension e [1], and radius r [0.3] given in the input. The extension e must be given as an integer, which will be incremented. The first point in the set is the original point in the point register. Using i_1 [1] and i_2 [last generated point] the range of points to be stored in the parameter list may be restricted.

Example: A phenyl group is to be generated. Place the free coordinate system in the center of the ring, the ring lying in the x,y-plane. Assuming that the first C atom has the number 23, the following sequence will generate the rest of the group:

```
PL 23
GP 6,3,C,1,.25,2
```

Generate two (resp. three) points in tetrahedral enviroment

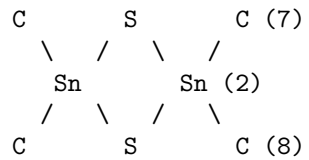
GZTU resp. GDTU $c_z, c_n, s, e, r, l, w, c_h, n_1, n_2$;

These commands are provided mainly to generate hydrogen atoms. With respect to the central atom with the code c_z two (GZTU) or three (GDTU) points, respectively, are determined and added to the parameter list, fulfilling the following conditions:

- The names of the atoms s (initial setting: H) and the extensions e (initial setting: ' ') and the radii r (initial setting: 0.2) are as specified.
- The distance from the center c_z to the atoms generated is l (initial setting: 1.08).
- The angle w (initial setting: 109.47) is set for $c_n - c_z - c_i$ (c_i : generated atoms) with 'GDTU'. With 'GZTU' the angle is set for $c_1 - c_z - c_2$.
- With 'GDTU', the code c_h together with c_z and c_n is used to define a plane that is supposed to contain one of the atoms c_i. Given the plane there exist two possible locations for a c_i. From these solutions that one is chosen where c_i is farther away from c_h. With 'GZTU' the center of gravity c_g between c_n and c_h is determined and the two generated atoms positioned in such a way that the plane $c_1 - c_z - c_2$ together with c_g is perpendicular to $c_n - c_z - c_h$.
- If no input is given for c_n, the program automatically searches for a suitable neighbor of c_z in the range n_1 [3] to n_2 [last atom]. With 'GZTU', a missing input for c_h is replaced by the next but one atom; with 'GDTU' a missing c_h is replaced by the first atom, thus allowing the definition of a coordinate system.

Example:

Assume the following molecule is known and the numbers given are the positions in the parameter list.



To construct H atoms around the C atom (7), enter the following command: `GDTU 7 2 H 1 .2 1.08 * 8 .` If the default values have not been changed up to now, the command `GDTU 7` would actually be sufficient.

Generate one atom in tetrahedral enviroment

GETU $c_z, s, e, r, l, c_1, c_2, c_3, n_1, n_2$;

Similarly as with 'GZTU' and 'GDTU', the position of one hydrogen atom in a tetrahedron around c_z is determined using 'GETU'. Initial settings: s = H, e = ' ', $r = 0.2$, $l = 1.08$, $n_1 = 3$, n_2 = last atom in the parameter list. The codes c_1, c_2, and c_3 describe the first three atoms in the tetrahedron around c_z. If for c_1 the value 100000 (default) is entered, the program tries to find these neighbour atoms automatically. In this case, a sphere of radius 3 Å around c_z [0] is searched for atoms in the parameter list from n_1 [3] to n_2 [last atom] (that are already in the code list), and the nearest three atoms are selected. The vectors $c_z \to c_1$, $c_z \to c_2$, and $c_z \to c_3$ will be normalized to 1 and added vectorially. The desired location of the H atom is now exactly opposite to the resulting vector a distance l from c_z.

Generate phenyl surrounding

GPHU $c_1, \ldots, c_6, s, r, l$;

The goal is to construct the H atoms belonging to a phenyl ring. $c_1, \ldots, c_6$ are the codes of six C atoms. Enter those codes that lack an H atom, as negative numbers. Defaults: name s = H, $r = 0.2$, $l = 1.08$. The extensions of the H atoms are taken from the C atoms to which the H atoms are bonded.

Centric group

ZG $n_1, n_2, ccentr, opt$;

All points stored in the parameter list from n_1 to n_2 will be mapped onto their images under an inversion, with the center being the point having the code *ccentr*. The resulting atoms are appended to the parameter list, if *opt* = A is specified (initial default setting). If *opt* = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Group relative

GREL $n_1, n_2, x, y, z, f, opt$;

All points stored in the parameter list from n_1 to n_2 will be shifted by the vector $(x, y, z) \cdot f$ with respect to the free coordinate system. The resulting atoms are appended to the parameter list if opt = A is specified (initial default setting). If opt = R is entered, the new coordinates replace the old ones. Default may be set using 'GOPT'.

Idealization of parameters

When studying crystal structures, one might wish to compare two sets of atoms. E.g. a molecule is rotated and shifted in the space to coincide as well as possible with a second one. The criterion for the quality of this fit is that the sum of squares of distances is a minimum (after shift and rotation). This problem can be handled by an algorithm given in Acta Cryst (1984) A40, 165-166. A disadvantage is that a rotation about 180° must be treated as a special case.

The result of the calculation is a rotation angle and the three direction cosines of the rotation axis and the RMS value. The transformation matrix mapping one group onto the other is stored and may be used for other calculations (c.f. 'TSYI'). One can easily check, whether this matrix corresponds to a crystallographic symmetry. One application of this procedure is the idealization of certain groups, resulting from crystal structure determination, e.g. the angles in a t-butyl group are usually not precisely 109.47 degrees, and the distances are often unequal, especially if the experimental data are poor. In order to idealize such groups, the parameters of an ideal model are entered (or constructed using the commands provided). Then the model is rotated and shifted, until a best fit is obtained. In the output, the shifted and rotated coordinates are printed ("ideal" points), together with the target points and the distance between the original and the idealized positions in Angstrom.

Idealize list

IDL $i_1, i_2,\ r_1, r_2, ..., r_n$;

The points in the parameter list at the positions $i_1, i_1 + 1, \ldots, i_2$ are shifted and rotated by the method mentioned above until the best match with the positions of $r_1, r_2, \ldots, r_n$ is found. The corresponding points are: $i_1 : r_1, i_1 + 1 : r_2, \ldots, i_2 : r_n$. To shorten the input, points of a sequence may be specified by a run $r_i - r_j$. E.g. the input 17 3 -6 9 is equivalent to: 17 3 4 5 6 9. The points will be weighted by the radii assigned to the ideal points. In the output, the rotation matrix, the translation vector, the three direction cosines of the rotation axis, the rotation angle, a list of distances, followed by the RMS value are printed. The direction cosines are also stored in the point register. If saved using 'AA', the rotation axis remains available for further use.

The parameters in the parameter list remain unchanged, if the option 'NRRI' (initial setting) is in effect. If 'RRI' is in effect a replacement of the "real" by the "ideal" points takes place. If 'CRRI' has been set, the replacement is not performed in case of a rotation angle of 180°.

Idealize

ID i_1, i_2, r_1 ;

This command performs the same task as IDL, but it provides a simpler way for the input, for the case that the sequence $i_1, \ldots, i_2$ matches $r_1, r_1 + 1, \ldots, r_1 + (i_2 - i_1 - 1)$, i.e. the points have to be in the correct order from the outset.

Idealize group

IDG $i_1, i_2, r_1, d, opt, nz$;

When using the command 'IDL' one has to know the corresponding points already beforehand, and with 'ID' even the sequences have to agree. In case of sufficient agreement of both groups with respect to their topology, 'IDG' first tries to find the corresponding points by itself. In case of success, 'IDL' is called.

The following steps are undertaken to find corresponding points:

- The weighted centers of gravity (c.o.g.) are calculated for both groups.
- From the "ideal" group three atoms are selected which are nearest to the center of gravity (but not nearer than 1 Å), and two of them chosen, using as criterion that the angle with the c.o.g. $\angle(\text{atom} - \text{c.o.g.} - \text{atom})$ be closest to 90°.

- From the "real" group up to six atoms are selected nearest to the c.o.g.. They serve as candidates for the following procedure:

  The "ideal" group is transformed using 'KTG', such that both centers of gravity coincide, and the chosen atoms from the "ideal" group agree as well as possible with two candidates from the "real" group. If successful, the solution is assigned a number and a value (via the mean distance between "real" and "ideal" group). Each 'KTG' generates a copy of the original group which remains unchanged in the process.

  Next it is checked, whether for the transformed "ideal" group the following is true: For each atom of this group there exists exactly one atom of the "real" group in a sphere of d [0.25] Å. If so, the next step is performed depending on the value of the number *opt* [1].

$opt = 0$:	No further action.
$opt = n$:	The command 'IDL' will be executed employing solution n.
$opt < 0$:	All possible solutions are investigated. No further action is taken. This may be useful when qualitatively different solutions exist.

The procedure may fail due to a poor agreement of the group transformed by 'KTG' with the "real" group. In this case a message is issued what the maximal number of corresponding atom pairs has been. If this number is sufficiently large, it might make sense to fix this set in a new matching attempt by setting nz [number of atoms in the whole group] to this number. Keeping these atoms fixed, the program then tries, by enlarging d (up to eight times), to find a matching for the remaining (not yet paired) atoms. If all atoms can be paired up, the next action is taken as specified by *opt*.

Add idealized coordinates

AI $i_1, i_2, r_1, r_2, \ldots, r_n$;

The coordinates of the idealized (rotated and shifted) points $i_1, \ldots, i_2$ overwrite the coordinates $r_1, r_2, \ldots, r_n$ (see 'IDL'). This is also done, if 'NRRI' is in effect. If all input $i_1, \ldots, r_n$ is omitted, the last input of 'IDL' is used.

Replace real by ideal parameters

RRI resp. CRRI resp. NRRI resp. KIG ;

These options control what happens after the transformation matrix and the translation vector have been found by 'IDL' (or 'ID'). If 'RRI' is in effect, the "real" coordinates are overwritten by the transformed ideal ones. With 'NRRI' (initial setting) active, nothing is done. If 'CRRI' (Conditional Replace) is set, a replacement takes place, except if the matrix is singular and the angle had been set to 180°.

Another effect is produced by the option-command 'KIG' (Keep Ideal Group), which is only of relevance in connection with the command 'IDG'. Since 'IDG' uses a copy of the "ideal" group only, this copy is no longer available after the command has been executed. If the option 'KIG' is activated, the copy of the group remains stored at the end of the parameter list, and e.g. can be plotted "on top of" the real group, in order to visualize their differences.

Control output while idealizing

OI resp. NOI ;

If 'OI' is in effect (initial setting), output of the "real" and transformed "ideal" points and the distance is printed. If 'NOI' is set, only the rotation angle, the direction cosines, and the RMS value is printed.

Overlay fragment

UFR $h_1, h_2, h_3, tol_d, tol_w, z_1, z_2, opt$;

Initial values: $h_1, \ldots, h_3 = 0$, $tol_d = 0.25$, $tol_w = 4$, $z_1 = 3$, $z_2 =$ number of the last atom of the background structure, and $opt = 1$. Entries will be kept.

The goal of this command is to overlay a fragment of the current (foreground) structure in several ways with the background structure in order to decide visually whether or not such an overlay is possible. The fragment has to be stored in the (foreground) code list, while a sufficiently large part of the background structure should be selected and stored in the (background) code list.

Three codes h_1, h_2, h_3 have to be selected to serve as a handle. From the atoms of the background structure, which have numbers from $z_1, \ldots, z_2$, suitable sets of 3 target atoms are selected such that within the tolerance tol_d the vectors $h_1 - h_2$ and $h_1 - h_3$ match with the corresponding ones connecting the target

atoms. Then combinations are created such that within the tolerance tol_w the angle $h_2 - h_1 - h_3$ of the handle agrees with the one formed by the vectors connecting the target atoms. In case such an angle is found, the command 'VZDP' is internally executed, i.e. the background structure is rotated and shifted in order to obtain a best fit of the handle. If the command 'UFR' is entered again but without parameters (i.e. `UFR;`) a new possible combination is searched for. This may be repeated until all combinations have been processed.

If *opt* is entered as a number $\neq 0$, it is checked in addition, whether the working names of the handle atoms agree with the names of the corresponding target atoms. (See 'AAN' p. 77.)

Overlay fragment automatically

UFRA $h_1, h_2, h_3, tol_d, tol_w, tol_n, z_1, z_2, opt, nlsg;$

Initial values: $h_1, \ldots, h_3 = 0$, $tol_d = 0.25$, $tol_w = 4$, $tol_n = 0$, $z_1 = 3$, $z_2 =$ number of the last atom of the background structure, $opt = 1$, and $nlsg = 1$. All entries will be kept.

When using the command 'UFR', it is up to the user to find a set of target atoms that are in agreement with the fragment. Instead, with 'UFRA' an automatic search is performed within the conditions given.

The code list of the background structure is cleared, and the command 'UFR' is internally executed using the parameters given. Then atoms of the background structure having numbers from z_1 to z_2 are selected that are located within a sphere with the radius tol_d around the atoms belonging to the foreground structure. If the number of atoms in the foreground fragment is equal to (or, if `IGFA 1` was set, is equal to or smaller than) the number in the part of the background structure under consideration, and $tol_n > 0$ had been entered, the search is repeated using this radius. If no further atoms are found, the fragment is identified as a success and the counter for solutions is incremented. In case this number equals $nlsg$, the solution is printed and the search procedure is stopped. The code list of the background structure contains the atoms corresponding to the fragment. A special meaning has $nlsg = 0$. In that case all possible solutions are searched for and printed.

Compare clusters

CCL $tol_d, tol_w, f, z_1, z_2, opt, nsol;$

Initial (default) parameter values: $tol_d = 0.25$, $tol_w = 4$, $f = 0$, $z_1 = 3$, $z_2 =$ number of the last atom in the current structure, $opt = 0$ and $nsol = 1$. Changes in the entries are saved for the next comparison.

The goal of the command is the comparison of two clusters ($f = 0$), or to search for the cluster of the foreground structure inside the background structure ($f \neq 0$). The cluster(s) is (are) saved in the parameter list(s) of the foreground- and the background structure at the positions from z_1 to z_2. If $f = 0$ is set, a scale factor is computed in a first step, and the lattice constants of the structure with the smaller volume is multiplied by this factor. In this fashion, clusters consisting of different types of atoms can be compared. Else, f is employed as the scale factor for the foreground structure, e.g. if the user wants to investigate whether the cluster exists in some fashion inside the background structure (which can be a periodic structure or another but bigger cluster).

In the second step, a suitable handle (set of three atoms) is selected, and, together with the other entries in the command, the handle is used as input, in order to execute the processes associated with the command 'UFRA'. If *opt* has a value different from 0, then the working names of the atoms have to agree for a successful mapping. If a successful mapping has been found, the transformation necessary is displayed in form of a 'VZDP' command. $nsol = 1$ indicates that only one solution is to be recorded; else, all solutions found will be displayed.

Cover with fragment and fill if empty

UFFL $tol_d, tol_w, opt, p, name, opt_2, nl, tol_c;$

Initial values: $tol_d = 0.25$, $tol_w = 4$, $opt = 1$, $p = 0$, $name =$' ', $opt_2 = 0$, $nl = 1$, $tol_c = 1.0$. Changes made to these entries remain until a restart occurs.

The purpose of this command is to map a structure fragment (usually an empty coordination polyhedron around some atom p) stored in the code-list of the foreground structure into the background structure such that all atoms of the fragment can be mapped to corresponding ones in the background structure. If such a subset of atoms is found, the position is calculated where the atom p (in most instances the atom coordinated by the polyhedron) would be mapped to, and one checks, whether within a sphere of radius tol_c around this location in the background structure an atom belonging to the background structure exists. If this is *not* the case, a new atom with the name *name* is generated at this position in the background structure. The

numbers $1, 2, \ldots$ serve as distinguishers. If the entry for *name* is blank (initial value), the name of the newly generated atom equals the name of atom p.

Internally, the KPLOT command 'UFR' is executed with the parameters tol_d, tol_w and opt. A set of three suitable handle-atoms is automatically selected from the content of the codelist of the foreground structure (the fragment) according to the criterion that the area of the triangle spanned by the three atoms should be maximal.

Since for the newly generated atoms the symmetries of the space group have not been taken into account (unless one wants to break the symmetry of the background structure, adding a new atom should automatically generate several symmetry-equivalent ones), the result usually needs to be refined, e.g. by using the command 'RPSY' afterwards.

If $opt_2 > 0$ is entered, it is checked, whether for all atoms in the fragment the working names of fragment atoms and image atoms in the background structure agree. If one sets $opt_2 = 2$, then no image atoms of p are generated, and instead the process stops after nl solutions have been found. In the case a successful mapping is found, the actual list of which atom of the fragment is mapped to which atom in the background structure is produced using the following command sequence: `cc; ao; auf ****` tol_d `; nao;`

Note that 'UFFL' can be used to compare two clusters of atoms by using the following procedure: load widely spaced periodic arrangements of the two clusters as foreground and background structure, place the content of one unit cell of the foreground structure into the codelist as fragment (equals the whole cluster), and then employ 'UFFL' where the parameters for p and *name* keep the initial values and $opt_2 = 2$, e.g. `uffl *****2*`.

Print overlay matrix

OUM n;

Default: $n = 0$. When using the commands 'UFR' and 'UFRA', but more generally for all commands that rotate or shift the unit cells with respect to each other, the transformation matrix is not printed. With the command 'OUM', the transformation matrix that belongs to the current comparison is printed. Formally the matrix $\mathbf{T}$ and the shifting vector $\mathbf{t}$ are calculated that solve the equation

$$\mathbf{T} \cdot \mathbf{B}_v + \mathbf{t} = \mathbf{B}_h$$

where $\mathbf{B}_v$ is the basis of the foreground structure and $\mathbf{B}_h$ the basis of the background structure respectively. Note that $\mathbf{T}$ is a valid transformation matrix only in case of a valid mapping between the two structures. If for n a value $\neq 0$ is given, the transformation will be executed.

Rotate cell around a vector

DZV c_1, c_2, w;

The cell of the current structure is rotated around the vector $c_1 \rightarrow c_2$ by w degrees. The sense of rotation follows the right hand rule, i.e. to the left as seen from the tip of the vector. No default is provided, but earlier entries are preserved.

Move cell by a vector

VZV bzw. VZV c_1, c_2, d;

The cell of the current structure is moved in the direction of the vector $c_1 \rightarrow c_2$. The amount of the shift is by d Å when using the command 'VZ' and by $d\times$ the length of the vector when using 'VZV', respectively. No default values are provided, but earlier entries are preserved.

Move and rotate cell (macro)

VDZ d, w, opt_1, opt_2;

Initial values: $d = 0$, $w = 0$, $opt_1 =$ `STPU`, $opt_2 = 1$. All entries are preserved. This macro moves and rotates the current cell by the amount d and w, respectively. First a drawing is produced, and then the program waits for input. The following input is used in the macro:

Only Return	The current action is repeated.
$+n$ or $-n$	($n = 1, 2$ or 3): The new rotation axis is the **a**-, **b**- or **c**-axis. Entering a negative value results in the reversal of the sense of rotation.
$-$	The direction of rotation or translation is reversed.
xyz	(Three-digit number) Shift according to the three-digit code (e.g. 556 = shift along **c**)
$+$	Switch from translation to rotation. Axis of rotation is the prior direction of the translation.

All other entries stop the macro, and are treated as a new command.

If one enters `EPU` for opt_1, a single figure is plotted. opt_2 determines, whether the scale factor is recalculated before a drawing is generated: 0/1: no / yes.

Define and transform group

DTG $n_1, n_2, h_1, h_2, h_3, g_1, g_2, g_3$;

If the command 'CMPZ' has failed, one wants (perhaps) to discover the reason for this failure, or one may want to visualize some aspects of the procedure used. The basic idea is to overlay a part of one structure, called "finite group", onto another structure. But some preparatory work is necessary before applying this command.

1. A finite subset of atoms (group **B**) of the first structure in the comparison has to be created as a KPLOT-file.
2. The second structure (**A**) is loaded into KPLOT, and the finite group (**B**) is subsequently imported.
3. A "handle" has to be defined by three atoms (h_1, h_2, h_3) from the finite group **B**.
4. Three sets of atoms, $g_1 = k_{1j}$, $g_2 = k_{2j}$ and $g_3 = k_{3j}$, each consisting of several atoms $k_{11}, k_{12}, \ldots$, $k_{21}, k_{22}, \ldots, k_{31}, k_{32}, \ldots$ belonging to **A** have to be selected. During the matching procedure, h_1 matches one atom of g_1, h_2 one of g_2, and h_3 one of g_3, respectively.

The numbers n_1 and n_2 (no defaults) are the positions of the first and the last atom of the finite group in the parameter list, respectively. The h_i are pointers to the parameter list, while each g_i is entered as an explicit list of codes k_{ij} taken from the code list. The first code in each set g_i, k_{i1}, is indicated by a minus sign placed before the code. All codes until the next code with a minus sign belong to the same set. Here, the first minus sign may be omitted. If the second and third sets are identical, one may abbreviate the third set by simply entering a single zero.

Now one combination $k_{ij} \in g_i$ $(i = 1, 2, 3)$ is selected that is contradiction-free, i.e. from each g_i one k_{ij} is selected, and no two of the k_{ij}'s are the same. Then the finite group is transformed via 'VGDP' such that the h_i match the selected k_{ij}. The parameters passed to 'VGDP' are printed. When the command 'DTG' is given without any argument, this is interpreted as: generate the next combination of k_{ij}. If no new combination is found, a message is printed, but the k_{ij}'s are reset to the starting condition, and the procedure may be continued.

Example: `DTG 5 73  7 69 44  3 -465506 465402 4 455505  0`

It is advisable to write a short macro in order to view the results:

```
MACR 9
DTG;
GG
WAIT
MVTO 1
ENDM
```

In order to simplify the entries for 'DTG', the following shortcuts are provided when dealing with the most common case. If exactly two numbers (n_1, n_2) are specified, the atoms of the handle are expected to be in the mouse list. If there are exactly three atoms in the mouse list, the sets $g_1, \ldots, g_3$ are expected to be in the codelist at the positions $c_1, \ldots, c_n$, i.e. $g_1 = \{c_1\}$, $g_2 = \{c_2, \ldots, c_n\}$ and $g_3 = g_2$. If there are more than three atoms in the mouse list $(c_4, \ldots)$ these atoms will be interpreted to be the sets $g_1, \ldots, g_3$ etc.

Best planes

Fit of a least squares plane

EFTP resp. EFTC n_1, n_2 ;

Using a least squares algorithm, the free coordinate system is redefined in such a way, that the x-y-plane becomes the "best" plane, i.e. the sum of squares of the distances of the defining points from this plane is minimal. The defining points are all the points in the parameter list from n_1 to n_2 with 'EFTP', and all the codes in the code list with atom numbers in the parameter list from n_1 to n_2 with 'EFTC'. The radii of these atoms are used as weights.

Fit of a least squares plane by group of codes

EFTG ;

Works in principle like 'EFTC', but here the codes are taken from the group list of codes (mouselist, see also 'LGC').

Fit of a least squares plane by symbol(s)

EFTS $s_1, f_1, t_1, \ldots, s_3, f_3, t_3$;

Works in principle like 'EFTC', but here the codes are selected from the code list obeying the condition that the name in the parameter list is s, and the range in the parameter list is (f, t) (defaults: $f = 3$, $t =$ last atom). Up to three "kinds" of atoms can be selected.

Sorting

Sort according to free system

SRT k ;

The codes in the code list are sorted in ascending order according to their values in the direction k in the free coordinate system (x: $k = 1$, y: $k = 2$, z: $k = 3$).

Sort according to triclinic system

SRTT k ;

The codes in the code list are sorted in ascending order according to their values in the direction "k" where k is the triclinic coordinate (x: $k = 1$, y: $k = 2$, z: $k = 3$).

Sort according to points

SRTP n_1, n_2, m ;

The atoms in the parameter list at positions $m, m + 1, \ldots, m + n_2 - n_1$ are sorted in such a way that they match as well as possible to the atoms $n_1, \ldots, n_2$. Default for m is $n_2 + 1$. This command may be used to check for symmetry elements. If e.g. a twofold axis is expected to be present, the atoms in the parameter list may be rotated around this axis using 'DG'. Then this new group is added to the parameter list, and can be sorted using 'SRTP', followed by a comparison of the two groups using 'ID'.

Sort codes

SRTC ;

All codes in the code list are treated as numbers which are sorted in ascending order.

Sort according to names

SRTN n_1, n_2 ;

The entries in the parameter list are sorted in alphabetical order. This sorting is done beginning at n_1 (default: 3) and ends at n_2 (default: last atom). Note that the code list is not adjusted, and may become meaningless.

Sort using list

SRTL $[n_1[, n_2]][s_1[, s_2, \ldots]]$;

The entries from n_1 (default 3) to n_2 (default last atom) in the parameter list are sorted into groups according to a list of symbols $s_1, s_2, \ldots$ *Example:* `SRTL 3 15 SN C O H ;` The atoms in the range 3 to 15 are sorted in such a way that tin, oxygen, and hydrogen atoms appear in groups. The sequence within a group is not

changed. Note that the code list is not adjusted, and may become meaningless. If no symbols are specified, the parameters will be sorted into groups according to the different atom names, in the order of appearance.

Sort according color pointer

SRTF n_1, n_2 ;

The entries in the parameter list from n_1 to n_2 are sorted in such a way that the color pointers are listed in ascending order. Defaults: 3 and last atom. Note that the code list is not adjusted, and may become meaningless.

H atoms to the bottom

HNH n_1, n_2 ;

The entries in the parameter list from n_1 to n_2 are sorted in such a way that the hydrogen atoms are moved to the end of the parameter list. Defaults: 3 and last atom. The sequence of atoms within the groups is not changed. Note that the code list is not adjusted, and may become meaningless.

Handling of planes

A plane in space is defined by its normal vector m, $|m| = 1$ together with a distance d to the origin, i.e., the plane consists of all the points whose coordinate vectors v have the property $vm = d$. The direction of m is chosen in such a way that d is a positive number. The commands described in this section are used to define planes in various ways, and show how to deal with planes when performing calculations. Planes are stored as "pseudo-atoms" in the parameter list in the following way: The x, y, z coordinates are the Cartesian coordinates of the normal vector m, and the radius is the distance d to the origin. Planes are adressed by their position in the parameter list.

Plane via Miller indices and d

EML resp. FACE h, k, l, d, n, s, e ;

Using the command 'EML', a plane is defined by the Miller indices h, k, l, and the distance d (no defaults) from the origin. The plane will be stored in the parameter list at the position n [last atom+1, i.e. append the pseudo-atom] overwriting any previous entry with the name s [UNDF] and the extension e [' ']. If no name / extension is entered, the previous name / extension at the position n in the parameter list is not changed. The command 'FACE' is identical except that d is multiplied by a factor defined using the command 'FACF' (initial setting 10) before storing the plane.

FACE factor

FACF f ;

Definition of the factor f used in the command 'FACE' (initial setting $f = 10$).

Plane via Miller indices and a point

EMP h, k, l, c, n, s, e ;

Using this command a plane is defined by the Miller indices h, k, l and a point with the code c (no defaults) contained in the plane. The plane will be stored in the parameter list at the position n [last atom+1, i.e. append the pseudo-atom] overwriting any previous entry with the name s ['UNDF'] and the extension e [' ']. If no name/extension is entered, the previous name/extension at the position n in the parameter list is not changed.

Plane via normal vector and a point of intersection

ENP $c_1, c_2, f, name, ext., n;$

Defaults: $c_1 = 0$, $c_2 = 0$, $f = 0$, n : first free location in the parameter list, *name, ext.:* no defaults. The plane is defined by a normal vector from code $c_1 \rightarrow$ code c_2, which intersects the plane at a point lying on the vector at the position $(c_1 + c_2)/2 + (c_2 - c_1) * f$, i.e. by default at the middle of the vector. The vector does not need to be normalized. It is stored in the parameter list at the position n.

Plane via three points

EP c_1, c_2, c_3, n, s, e ;

A plane is defined by three points that lie in the plane, having the codes c_1, c_2, and c_3 (no defaults). The plane will be stored in the parameter list at the position n [last atom+1, i.e. append the pseudo-atom], overwriting any previous entry with the name s ['UNDF'] and the extension e [' ']. If no name/extension is entered, the previous name/extension at the position n in the parameter list is not changed.

Intersection point of three planes

PEEE n_1, n_2, n_3 ;

The intersection point of three planes is calculated and stored in the point register. n_1, n_2, and n_3 (no defaults) are the positions of the planes in the parameter list.

Generate intersection points of planes

GESP p, e_1, e_2, tol ;

'GESP' is used to support drawing crystals. Starting with the description of a crystal by a set of planes $(h, k, l; d)$ (see 'EML'), the points needed to draw the crystal are generated as follows: All intersection points by all combinations of three planes which had been stored in the parameter list in the range e_1 [3] to e_2 [last entry in the parameter list] are calculated. From this set those points are selected which lie on the same side as the point with the code p [2] with respect to all planes (this is similar to the Wigner-Seitz cell around p), using the tolerance tol [0.001]. In other words: the polyhedron with minimal volume is determined.

Generate polyhedron

GPOL $n_1, n_2, tol, c_{opt}, p_{opt}, e_1, e_2$;

Points in the range from n_1 [no. of the first plane e_1 specified in 'GESP'] to n_2 [no. of the second plane e_2 specified in 'GESP'] in the parameter list are used to generate a polyhedron. All faces belonging to the convex hull are determined, and the appropriate plot commands are generated. A point is treated as belonging to a face, if its distance from the plane is less than tol [0.001].

The options c_{opt} and p_{opt} control the generation of codes and plot commands: if set to values unequal zero, they will be generated automatically; if set to negative values (default), in addition the code list and the list for plot commands are cleared before starting. In addition an automatic labelling of the faces may be generated, if with e_1 [0] and e_2 [0] the range is specified where the planes are stored (see 'GESP'). The center of gravity of each face is calculated and stored in the parameter list as an additional point. Each of these points is given a name, which is the number where the corresponding face is stored in the parameter list. If no matching face is found in the range $e_1, \ldots, e_2$, the name '?xxx' is used where xxx is the number of the plot command stored in the parameter list. This is an error message, where "?" indicates that the Miller indices could not be assigned when executing the command xxx. A 'BA' command is generated, such that the lower left corner of the first letter coincides with the point assigned to the face.

The following example produces an octahedron:

```
NDLG
Z 10
AE 2
SE 1
EML  1  1  1   1  * E 1  ! Input of planes
EML  1  1 -1   1  * E 2  ! on pos. 3 - 10
EML  1 -1  1   1  * E 3
EML  1 -1 -1   1  * E 4
EML -1  1  1   1  * E 5
EML -1  1 -1   1  * E 6
EML -1 -1  1   1  * E 7
EML -1 -1 -1   1  * E 8
GESP 2 3 10              ! Calc. intersection points
GPOL 11 16 * * * 3 10    ! Generate VD- and BA-commands
DK 2 15                  ! Define orientation
DK 1 15
BF ; EPU
DLG ; EAE 5
```

Wigner-Seitz Surrounding

WSU $u_1, u_2, z_1, z_2, dmax, dmin, tolg, tola$;

Default values: $u_1 = z_1 = 3$, $u_2 = z_2 =$ number of the last atom in the parameter list, $dmax = 3$, $dmin = 0$, $tolg = 0.001$ and $tola = 0.3$. The Wigner-Seitz Surrounding is defined as the set of atoms, whose Wigner-Seitz planes with the central atom contribute to the facets of the Wigner-Seitz cell around the central atom.

(Note: A Wigner-Seitz plane is defined as the plane, that lies halfway between two atoms and is orthogonal to the vector connecting these two atoms.) These atoms are determined using the following procedure:

1. The code list is emptied, and a central atom is selected from the atoms in the range $u_1 \dots u_2$ in the parameter list. This atom is placed into the code list. Next, we determine all the atoms in the range $z_1 \dots z_2$ in the parameter list, whose distance to the central atom is between *dmin* and *dmax*. *dmax* must be chosen sufficiently large.

2. Using these atoms, the Wigner-Seitz planes with the central atom are determined, and from among these the Wigner-Seitz cell around the central atom is constructed. This is achieved by internally applying the commands 'ENP', 'GESP' and 'GPOL' with the tolerance *tolg*.

3. For every facet of the Wigner-Seitz cell we construct a set of mirror points that are generated by reflection of the central atom at the plane than contains the facet. Each of these mirror points should be very close (within tolerance *tola*) to an atom from the range $z_1 \dots z_2$ in the parameter list. Note that within each sphere of radius *tola* around a mirror point, exactly one atom should be found.

This procedure is then repeated for the next atom in the range $u_1 \dots u_2$ in the parameter list.

The atoms found in this way constitute the Wigner-Seitz Surrounding of the central atom. They are printed together with the area of the facet they generate. The last such set of atoms is still in the code list after the 'WSU' command has been executed. The first line of each printed Surrounding gives the current central atom together with the numbers n_1 and n_2, which denote the range in the parameter list, where the corners of the Wigner-Seitz cell are listed. If one enters the command sequence `AE` n_2; `ACI` n_1 n_2; directly after 'WSU', one can visualize the last Wigner-Seitz cell.

Change VD parameters

CVDP $n_1, n_2, c_{or}, tol, r, p$;

In order to enhance the three dimensional impression of a polyhedron, the parameters of those 'VD' commands describing the back planes may be modified, e.g. the planes may be drawn by hatched lines or by thin lines. The list of plot commands is searched for 'VD' commands, beginning at n_1 [1] and ending at n_2 [999]. The points appearing in a matching 'VD' command are used to calculate a "least squares" plane, i.e. the plane to which the points are closest. Then one checks, whether the reference point with the code c_{or} [255501] is on the same side of the plane as the observer. In this case, the least squares plane is a back plane. The rest of the parameters r, p overwrite those of this 'VD' command (for their meaning c.f. 'VD'), and the changed command is moved to the end of the list. The parameter *tol* has no meaning here (it is included, in order to make the format agree with that of 'MFCE' - whatever is entered at this position will be ignored). *Example:* `CVDP **** 71` Back planes of the polyhedron will be drawn using hatched lines.

Calculate volume of a polyhedon

PVOL c ;

The command 'PVOL' is used to calculate the volume of a pre-defined polyhedron that contains an atom with the code c [255501]. It is assumed that the list of plot commands already contains all the VD commands necessary to draw all the faces of the polyhedron of interest. (A common way to generate these VD commands is to use the command 'GPOL'.) If the polyhedron does not already contain an atom that can be used to specify c, one can use the command 'SPC' to generate a code located at the center of mass of the polyhedron, and then create a dummy-atom at this location with the command 'AA'. 'PVOL' searches the list of plot commands for VD commands that describe a convex polyhedron containing the atom with the code c. If the commands belonging to such a polyhedron are found, the volume is calculated and printed (in $Å^3$).

Project group of codes onto plane

PGCE n ;

Sometimes problems arise when calculating intersection points using 'GESP', if the points lie approximately in a plane, but the program treats them as belonging to different planes. To move them such that they lie on a single plane, one might proceed as follows: 1) All points which should belong to one plane are placed into the group list of codes (using 'M' or 'LGC'). 2) A "least squares" plane is fitted through these points ('EFTG'). 3) The points are projected onto this plane ('PGCE 3'). n is the number of the coordinate which is set to zero (during the projection): 1=x, 2=y, and 3=z. Default for n is 3. The coordinates in the parameter list will be recalculated automatically.

Generate FACE instructions

MFCE n_1, n_2, mx, tol ;

The command 'GESP' generates a VD command for each plane. These 'VD' commands may be used by 'MFCE' to generate 'FACE' instructions, e.g. for the program HABITUS. This is especially useful, if the shape of the crystal has been modified by 'PGCE', because some indices may no longer be correct.

The list of plot commands in the range from n_1 [1] to n_2 [999] is searched for 'VD' commands. If found, a corresponding 'FACE' instruction is written to the file having the logical number *ntpch*. The Miller indices are determined as follows: The smallest integer numbers up to the order mx [50] matching best to the actual (real) values are calculated. A prerequisite is that a number exists which, when multiplied with the actual values, produces numbers which do not differ more than *tol* [0.05] from the integer values above. If no suitable number is found, the best solution found is accepted (no warning is given).

Point by vector and plane

PVE p_1, p_2, e ;

The intersection of the vector $p_1 \rightarrow p_2$ and the plane e is determined, and the coordinates of this point are stored in the point register. p_1 and p_2 are codes and e a number in the parameter list denoting a plane. There are no defaults.

Distance of a point from a plane

LEP n_e, n_c ;

The distance of the point having the code n_c from the plane listed as the number n_e in the parameter list is calculated and given as output. At the same time this value is kept and used for several other commands as default value (see 'L'). There are no defaults.

Coordinate system by two planes

KEE c_{or}, e_1, e_2 ;

The (orthogonal) free coordinate system is defined in such a way that the origin lies in the point with the code c_{or} (no default), the x-axis corresponds to the intersection line of the two planes e_1 and e_2 (in the parameter list), and the y-axis is lying within the plane e_1. Of course the planes have to be shifted so that the x- and y-axis intersect at c_{or}. The direction of the x-axis is defined in such a way that the point with the triclinic coordinates 0,0,0 possesses a positive x coordinate in the free coordinate system.

This command is meant to support the drawing of crystals, together with other "plane" commands. One can proceed e.g. as follows: Assuming that the Miller indices of the planes and the lengths of the edges are known, first three planes are defined by 'EML', then the intersecting point (c_{or}) is determined by 'PEEE', and the coordinate system is defined with the origin at this point and the x-axis parallel to the intersection line of two of the planes. The next vertex is now defined with the command 'P' and the coordinates $(k, 0, 0)$, where k is the length of the edge. The next plane can then be placed through that point by 'EMP'.

Coordinate system by Miller indices

KML $h, k, l, [, c_{or}[, c_1, c_2]]$;

The free coordinate system is defined in such a way that the x-y-plane of the free coordinate system has the Miller indices h, k, l. c_{or} is the code of a new origin and $c_1 \rightarrow c_2$ is used to define the x-axis. If no code is specified for c_{or}, the origin remains unchanged. If no values are specified for c_1 and c_2, the program selects a suitable vector automatically.

Definition of a KML range

DKML $h, k, l, h_{min}, k_{min}, l_{min}, h_{max}, k_{max}, l_{max}, h_{inc}, k_{inc}, l_{inc}$;

If a region is to be scanned in order to determine a sequence of Miller indices to be used in the command 'KML', the scanning range is defined by 'DKML'. The meaning of the parameters is: h, k, l are the current values, which will be incremented by 'NKML'; subscripts *min*, *max*, and *inc* are the minimal, maximal, and incremental values, respectively. Default values for the last 9 numbers are 6×0 and 3×1. If 'DKML' is entered without parameters at all, the current values are shown.

Next KML

NKML ;

This command works like 'KML', but the Miller indices are taken from an array defined by 'DKML', and incremented before being passed to 'KML'. The sequence of the h, k, l is: $0, 1, -1, 2, \ldots$ as long as these numbers lie within the range chosen with 'DKML'.

Store the coordinate system via a normal vector

SKE n, s, e ;

The vector normal to the x-y-plane of the free coordinate system is stored in standard format in the parameter list on the position n (default: first free location), with name and extension given by s (no default) and e (no default).

Miller indices of x-y-plane

MILL *tol* ;

The Miller indices of the x-y-plane of the free coordinate system are calculated and printed. The program tries to idealize them to integer numbers up to the order 5. A number is treated as an integer, if its difference from the nearest integer is less than *tol* (default: 0.01).

Miller indices of a plane

MLLE n_1, n_2, tol ;

The Miller indices of the planes stored in the parameter list from n_1 to n_2 (defaults: 3, last "atom") are calculated and printed analogously to 'MILL'. This command is useful when drawing crystals, because the labeling of the planes is done by using the positions of the planes in the parameter list.

Load plane

EL n ;

The normal vector corresponding to the plane stored at the location n (no default) in the parameter list is loaded into the point register. When manipulating planes, it is important to take into account that planes are stored in Cartesian coordinates in the parameter list.

Store plane

ESP n, s, e ;

The point currently in the point register is treated as the end point of a normal vector of a plane going through the origin. The Cartesian coordinates are calculated and stored in the parameter list at the position n (default: first free location). If no name is given, the name at that position is not changed. Default if no name is at n is "UNDEFI".

Conversion of plane parameters

CVNT and CVNK, respectively, n_1, n_2 ;

As has been mentioned before, the *Cartesian* coordinates of the normal vectors of the planes are stored in the parameter list. However, this can lead to errors during transformations. When planes are to be transformed, they must first be transformed into the triclinic system ('CVNT'). After the transformation, they need to be transformed back into the Cartesian system ('CVNK'). The planes that are to be transformed are located between n_1 and n_2 in the parameter list. Default values are $n_1 = 3$, $n_2 = 0$.

Example: Assume that in the parameter list there are planes at the locations 3 - 10 that are to be transformed. This can be achieved using the following sequence of commands:

```
CVNT 3 10    Convert to triclinic.
TZ ...       Perform transformation.
CVNK 3 10    Convert back to Cartesian.
```

Afterwards, the transformed planes can be printed as FACE-commands.

Print FACE-line

OFCE n_1, n_2, tol ;

The entries in the parameter list from n_1 to n_2 are interpreted as planes and are written as FACE-commands to the unit with the logical number *ntpch*. Default for n_1 is the number of the last "atom". Default for n_2 is 0. Attention! The program does not check, whether the Miller indices are integers. This should be checked beforehand, if necessary, using the command 'MLLE'. *tol* has the same meaning as for the command 'MILL'.

Angle between two planes

WEE e_1, e_2 ;

The angle between two planes given in the standard format at the locations e_1 and e_2 (no defaults) in the parameter list is calculated and saved (analogous to 'W'). Warning! During the calculation it is not checked, whether the vectors describing the planes are normalized.

Comparison of two cells

Sometimes the problem arises to compare two quite different unit cells, if one suspects that they describe the same lattice or that one lattice is a subset of the other. In order to find out, some commands are provided by KPLOT. The easiest way is to use 'VZAB', because most things are done automatically. 'VZAB' uses 'SZ' and 'SZA', which are mentioned in this section, but these commands need no longer be used.

Definition of a cell A or cell B, respectively

DZA resp. DZB $a, b, c, \alpha, \beta, \gamma, type$;

If two cells are to be compared whether they may be transformed into each other, they have to be defined first. 'DZA' defines a cell A, and 'DZB' a cell B. The values are set initially to 1 1 1 90 90 90 P. These (or the changes) are retained (even if using 'RSTR') to serve as default values. *type* is one of the following letters:

F	face centered lattice
I	body centered lattice
A	A centering
B	B centering
C	C centering
Q	rhombohedral obverse setting on hexagonal axes
S	rhombohedral reverse setting on hexagonal axes

Compare cell a with cell b

VZAB ;

As a first attempt the program tries to transform cell A into cell B. A solution is accepted even if the (reduced) cell A is less than the (reduced) cell B. If there is no success, both cells are interchanged and the trial repeated. As tolerances the values defined by 'SZAT' are used, which are set initally to 1 1 1 4 4 4. If a solution is found, the transformation matrix and its inverse matrix are printed. Caution! There may be no atoms or symmetries present because they will be deleted.

Example 1: Which transformation is necessary to transform a rhombohedral reverse cell into a rhombohedral obverse cell?

```
DZA 10 10 12 90 90 120 S
DZB 10 10 12 90 90 120 Q
VZAB
```

(Answer: `TZ  1 1 0  -1 0 0  0 0 1`)

Example 2 (Eli Kroumova): There is a known phase transition for $Pb_3P_2O_8$ with the change of the cell from monoclinic C to rhombohedral one. The cell constants for the monoclinic structure are a=13.81, b=5.71, c=9.31, $\beta = 102.4$ and for the rhombohedral one: a=b=5.56 c=20.39.

```
DZA 13.81 5.71 9.31 90 102.4 90 C
DZB 5.56 5.56 20.39 90 90 120 Q
VZAB
```

(Result: (B $\rightarrow$ A) `TZ  1/3 -1/3 2/3  1 1 0  -1 1 0`). This result is found because the reduced cell B is smaller.

In fact 'VZAB' is a macro executing following commands:

```
AE 2        All atoms are deleted.
Z           parameters from cell B
SE 1
GTY         setting from cell B
RDZ         Cell reduction and
SZXZ        interchange with search cell.
Z           parameters from cell A
SE 1
GTY         setting from cell A
SZA         Search cell automatically
```

In case of success, the transformation matrix is calculated from the best solution and the transformation matrix which was used for the cell reduction. In case of no success, the procedure is repeated using the interchanged cells.

Print cells A and B

OZAB ;

Both cells which are to be compared are printed. If no cell volume has been calculated yet, it is given as 0.0.

Load cell

LZ n, gty ;

When searching for a suitable unit cell transformation, often more than one attempt is necessary. In order to avoid entering the same data many times, the command 'LZ' may be used to load the lattice parameters internally from one register to another. Note that there are three commands one can use to enter cell constants: 'Z', 'DZA', and 'DZB'. The following load operations can be performed:

$n = 1$	DZA $\leftarrow$ Z
$n = 2$	DZB $\leftarrow$ Z
$n = 3$	Z $\leftarrow$ DZA
$n = 4$	Z $\leftarrow$ DZB

In addition a letter may be specified with $n = 1$ or $n = 2$ to define the lattice type. If nothing is entered, the current lattice type remains unchanged.

Example: (Stefan Schlüter, 2003)

A structure with a "small" unit cell and a disordered Na site undergoes a transition upon cooling, to a structure having a "big" cell and an ordered Na site. This transition is an isomorphic one of index 2 in the space group $P\overline{1}$. But the transformation matrix found with 'VZAB' does not correspond to one given in the Int. Tables at the entry "Maximal isomorphic subgroups of lowest index". Thus, the transformation found must be a combination of one given in the Tables and an additional (trivial) one. How can one find it?

The best way is to write two small macros:

```
macr 1
z 11.7219 12.1969 12.3459 111.66 111.54 101.44 (small cell)
dzb 12.3459 12.8238 18.5172 92.15 90.62 106.03 (big cell)
endm
macr 2
lz 1
vzab
endm
```

Now one may test the transformations given in the Int. Tables:

```
1;
TZ ..... (solution: TZ 1 1 0 -1 1 0  0 0 1)
2;
```

Search cell

SZ $a_1, a_2, b_1, b_2, c_1, c_2, al_1, al_2, be_1, be_2, ga_1, ga_2$;

The following problem may arise: Given a cell (1) with lattice constants $a_1, b_1, c_1, \alpha_1, \beta_1, \gamma_1$ and a cell (2) with lattice constants $a_2, b_2, c_2, \alpha_2, \beta_2, \gamma_2$. Question: Does cell (2) describe the same cell as (1), or a super cell of (1)?

The command 'SZ' supports the search for an answer in the following way: As preparation, those points, which would lie on (and thus define) possible axes through the origin of the sought-after cell, are entered in the parameter list. Points entered at a_1, a_2 are the first and last ones that can be employed to define an a-axis, b_1, b_2 for a b-axis and c_1, c_2 for a c-axis, respectively. The angles are limited to lie in an interval between a minimal and maximal value, given by al_1, al_2, be_1, be_2, ga_1, ga_2 for angles α, β and γ, respectively.

When executing the command 'SZ', all possible test combinations that can be constructed from the first six entries of the command are generated, and the test angles that belong to the resulting cells are calculated. If these do lie within the prescribed angle intervals, the result is printed. In addition, the volume of the test cell is computed and printed. If the volume is negative, the solution corresponds to a left-handed coordinate system. Note that the triclinic coordinates of the points corresponding to the successful test cell represent the transformation matrix to this new coordinate system. (See also 'TZP')

Remark: This command is actually used for internal purposes, and is useful as a stand-alone command in special cases only. See 'SZA' and 'VZAB'.

Search cell automatically

SZA $a, b, c, \alpha, \beta, \gamma$;

Sometimes the problem arises to compare two quite different unit cells, if one suspects that they describe the same lattice or that one lattice is a subset of the other.

The command 'SZA' automatically searches for 'candidates' for the a-, b-, and c-axis which correspond to axes of the desired cell and stores them in the parameter list. In particular, the following steps are executed:

- The code list is deleted. If no symmetries are present, 'SE 1' is executed to obtain the identity.
- An atom having the coordinates (0,0,0) is added to the parameter list.
- This atom is used three times as origin for the command 'AKS' with values $a \pm a_{tol}$, $b \pm b_{tol}$ und $c \pm c_{tol}$ (see 'SZAT') and stores the points found in the parameter list.
- The command 'SZ' is executed using angle intervals $\alpha \pm \alpha_{tol}$, $\beta \pm \beta_{tol}$ and $\gamma \pm \gamma_{tol}$.

Caution! No other symmetries besides the identity (and in certain cases centering symmetries) may be present. The tolerances can be set with 'SZAT'.

The parameters in this command have no initial settings. Once 'SZA' has been used the last values of the parameters serve as defaults.

The following example (taken from an exercise of the Bärnighausen course about symmetry relations in crystal structures) should demonstrate the procedure:

$\alpha - Sm_2O_3$ is unknown in pure form. From other compounds the cell constants were estimated to be a=3.79 and c=5.94 (trigonal system, Z=1). $\beta - Sm_2O_3$ crystallizes monoclinic with a=14.18, b=3.63, c=8.841, and $\beta = 99.92°$, Z=6. Does this cell contain the "small" one?

After starting KPLOT one enters:

```
SE 1 (Identity only)
MORE 1 (To obtain more output about the process)
Z 3.79 * 5.94 * * 120
OZ (Volume should be 73.89 (=V1))
SZA 14.18 3.63 8.841 * 99.92 (V=443 is almost six times V1)
U  (Shows candidates for a, b and c)
TZP 59 82 106 (transforms the small to the big cell)
(The points used are the columns of the transformation matrix;
the inverse matrix is printed.)
```

Search cell automatically - tolerances

SZAT $a_{tol}, b_{tol}, c_{tol}, \alpha_{tol}, \beta_{tol}, \gamma_{tol}$;

The values for the tolerances used in the commands 'SZ', 'SZA', and 'VZAB' are initially 1 1 1 4 4 4. 'SZAT' redefines these tolerances. These values serve as defaults for 'SZAT'.

Interchange cell with search-cell

SZXZ ;

When applying 'SZ' or 'SZA', it is assumed, that the search-cell entered is equal or greater than the cell entered using 'Z'. If this is not true, the cells may be interchanged with 'SZXZ'.

Compare two cells

CMPZ $tol_s, tol_g, tol_w, tol_v,\ ref;$

Initial values: $tol_s = 0.5$, $tol_g = 0.1$, $tol_w = 4.0$, $tol_v = 0.5$, $ref =$ ' '. Two cells *with* atom content are compared, in order to determine, whether the structures are the same (may be isotypical). tol_s is the maximally allowed departure of the corresponding atoms in Å and tol_g is a number, which multiplies the lattice constants, in order to fix an appropriate tolerance interval. tol_w is the maximal allowed deviation of angles in degrees. tol_v finally is the maximal allowed variation $V(1 \pm tol_v)$ of the cell volume during the affin mapping. *ref* is the name of one sort of atoms which should appear rarely in the structure with the smaller cell volume. If it is entered as blank or omitted the program itself searches for a suitable one. The comparison algorithm follows the following strategy:

- A list is produced, how the volume of the larger cell can be transformed into the one of the smaller cell. If necessary, the smaller cell is doubled or otherwise enlarged appropriately.
- For each of the possible transformations, the two cells are shifted such that one atom of the asymmetric unit of one cell has the same location as one of the atoms in the asymmetric unit of the second cell.
- Next, it is checked, whether, within the tolerance tol_s this agreement is also found for all other atoms including the ones generated due to the symmetries of the structures. If this is the case, the transformation matrix and the shift are printed, together with the amount of misfit.

If no agreement is found, a failure notice is printed. The misfit for the atoms is calculated according to

$$mfa = \sqrt{\sum_{i=1}^{n} s_i^2/n}$$

where the s_i are the distances between pairs of corresponding atoms, while for the misfit of the cell due to the affine mapping the following formula is used:

$$mfc = \sqrt{\frac{\sum_{i=1}^{3}(|a_i| - |a_i'|)^2 + (|d_i| - |d_i'|)^2}{6V^{\frac{2}{3}}}}$$

where the a_i are the lattice constants, the d_i the diagonals, and V the volume of the unit cell.

Comparison of two cells - setting of options

CMPO $opt_1, \ldots, opt_8;$

Initial values: $opt_1 = 0$, $opt_2 = 1$, $opt_3 = 1$, $opt_4 \ldots opt_6 = 0$, $opt_7 = 1$, and $opt_8 = 0$. The execution of the command 'CMPZ' is controlled by the following options:

$opt_1 = n$ The determination of the possible transformation matrices is preceded by the calculation of the "number density", i.e. the number of atoms per volume followed by a rescaling of the cell such that a number density of 0.045 atoms per Å^3 results if $n = 3$ is given. This adjustment is usually useful, since the absolute size of the unit cells does not matter in these comparisons. For $opt_1 = 0$, this adjustment does not take place. If $n = 1$ or $n = 2$ is given, the number density of structure 1 or 2, respectively, is selected, and the structure $3 - n$ is rescaled.

$opt_2 = 0|1|2$ This option controls different ways how atom names are taken into account. These names are taken from a specific field that contains for the atoms so-called 'working names' (see below) which can be adjusted according to various comparison strategies. When using $opt_2 = 0$, no check will be done, i.e. all atoms are assumed to have the same working name. In contrast, the names have to agree in case $opt_2 = 1$. When using $opt_2 = 2$ working names will be generated according to the sum formula, and symbols having the same index will be permuted additionally. For each such permutation, a comparison is performed.

$opt_3 = 1$ In case of a successful comparison the result is optimized by using the centers of gravity of both structures, respectively, as matching points. This will minimize the RMS value. Entering a zero suppresses this feature.

$opt_4 \neq 0$ If one has to increase the tolerance tol_s in the command 'CMPZ' very much due to large distortions, it might happen that there are several atoms within a search sphere. If $opt_4 \neq 0$, this is tolerated, and for $opt_2 \neq 0$ it is checked, whether at least one working name in the sphere agrees with the working name of the atom under consideration.

$opt_5 = 1$ If the two structures have no center of symmetry, a second comparison is performed for the centrosymmetrically related structure if necessary.

$opt_6 \neq 0$ After execution of 'CMPZ' the lattice constants will be reset to their initial values. Note that due to the affine mapping the cell may have been distorted.

$opt_7 \neq 0$ In case of a successful comparison there may be more than one solution. If a number $n > 0$ is given here, the program stops after the nth solution has been found. Note that the nth solution is then available for further analysis. If a 0 is specified, *all* solutions will be determined and the corresponding transformation matrices will be printed.

$opt_8 = 0|1|2$ The usual case is that the structure having the bigger cell volume (taking centerings into account) is selected and mapped affinely ($opt_8 = 0$). With $opt_8 = 1$ or 2, the structure number (1 of 2, respectively) may be prescribed.

Idealize Fragment

IDF d, n ;

Default values: $d = 0.5$, $n = 4$. After a successful structure comparison using the command 'CMPZ' one has not reached an optimal 'agreement' between the two structures. However, atoms that correspond to each other lie very close such that they are uniquely associated in space. The command 'IDF' can now be used to rotate and shift the foreground structure in such a way that the sum of the squares of the distances of these corresponding atoms become minimal. These distances are weighted according to the radii of the atoms in the background structure.

In order to find the correct correspondence, the code list of the background structure is re-sorted. Here, pairs of atoms corresponding to each other are generated, which exhibit a distance $\leq d$. If one cannot assign all atoms in this fashion, d is increased by 25%, and one repeats the assignment process for all the not-yet-assigned atoms. This assignment process is repeated up to n times if necessary.

Change working names

AAN *Name*, $Name_{new}, n_1, n_2$;

Default values: $n_1 = 3$, $n_2 =$ last atom in the current structure. In order to avoid conversions, the element names of the atoms that have been entered are stored in a second specific field in capital letters. These names can be modified for specific purposes. In addition, the name change can be restricted to a certain part of the parameter list.

Example: The structures of the two compounds $BiSeAlCl_4$ and $BiTeAlCl_4$ are to be compared. Here, the Bi, Al and Cl atoms shall be mapped into the same atom types, while the Se and the Te-atoms are to be mapped into each other. This can be easily achieved with the following two commands (the current structure is the Te-compound):

```
AAN Te Se
CMPO * 1
```

Reset working names

RAN n_1, n_2;

Default values: $n_1 = 3$, $n_2 =$ last atom in the current structure. The working names of the atoms from n_1 to n_2 of the current structure are reset to their original values.

Move cell according to three points

VZDP $p_1, p_2, p_3,\ q_1, q_2, q_3$;

The p_i and q_i are codes, that refer to points in the current and non-current cell, respectively. The non-current cell is rotated and moved such that the two triplets are brought into the best-possible agreement. Default values are six zeroes, i.e. all codes are taken from the mouse list (which has to be defined prior, of course). Note that the p_i and q_i belong to different cells; thus, the order of the points is important.

Searching symmetries

Find symmetries

SFND $n_1, n_2, ref, tol_g, tol_s, tol_t, n, opt$;

In version 7.4.0 of KPLOT or higher, one can search for symmetries in a crystal structure. Since the algorithm uses other resources of KPLOT, two dummy atoms must be placed on positions 1 and 2 of the parameter list having the coordinates (.5,.5,.5) and (0,0,0) respectively. In order to increase the performance, the tolerances (given in Angstrom) are not used as spheres, but the smallest triclinic box is used that contains this sphere.

One has to specify by choosing a range n_1 to n_2, which atoms are to be included in the test. Default values are: $n_1 = 3$ and $n_2 =$ last atom. ref is a symbol, i.e. the name of the type of atom serving as reference for the initial selection of symmetry candidates. It is advisable to select the type having least atoms in the unit cell. If no input is given, such a suitable type is searched for automatically.

The process of searching for symmetries consists of several steps. First one searches for 2-, 3-, 4- and 6-fold axes matching the lattice. The directions found in this way are candidates for rotation axes, screw axes, or roto-inversion axes. Here the tolerances are $tol_g \times$ length of the rotating vectors. Default for tol_g is 0.1.

In the next step the content $(n_1, \ldots, n_2)$ of one unit cell is placed into the code list. All combinations of two atoms taken from the type ref are used to define the position(s) of the axis, a glide vector or in case of a rotoinversion axis the inversion point. This symmetry is then tested for all atoms. Here the default value is $tol_s = 0.25$. If an axis is found, its type will be printed together with its direction, a point lying on the axis and the glide vector. See also command 'ST'. Finally centers of symmetry are searched for using tol_s (see also 'ZZ'), and possible translation symmetries are detected using tol_t (default: $tol_t = 0.25$) as tolerance. The components of all translation vectors have to be rational with respect to the lattice constants. $1/2, 1/3, 2/3, \ldots, 1/n, \ldots, (n-1)/n$ are possible values. Default: $n = 4$; maximal value: $n = 8$; minimal value: $n = 2$.

If *opt* is not zero (default: 1), a file will be created to logical number *ntpch* containing the results of the symmetry search. This file can be processed by the program RGS directly (read into RGS via 'GET') to derive the space group from the symmetries. The tolerances and the value for *opt* are kept as default for the next application of 'SFND'.

Print symmetry-axes

OSA ;

The result of 'SFND' is stored in order to be used by other commands ('ZIDL' and 'IDA'). This list may be printed by 'OSA'.

Ignore foreign atoms

IGFA n ;

Applying high tolerances when using 'SFND', i.e., when searching for pseudo-symmetry, it may happen that foreign atoms are detected inside the mapped spheres of atoms. Usually the symmetry in question is rejected in such a case ($n = 0$, initial setting). If $n = 1$ is set, foreign atoms are ignored.

Search for symmetries in a slab

SLAB n ;

Initial value when starting the program KPLOT: $n = 0$. If a value $\neq 0$ is entered with the command 'SLAB', the algorithms 'SFND', 'ZFND', etc. for the search for symmetries are modified such that only symmetry elements along special directions are searched for, based on the specific features of the slab. The slab has to be oriented in the following fashion in the coordinate system in KPLOT: The slab is infinitely extended along the (a, b)-plane. Furthermore, the middle of the slab should be at about the z-coordinate $z = 1/2$, and the coordinates of the atoms in the upper and lower boundary of the slab should be still far away from $z = 1$ and $z = 0$, respectively. Furthermore, when the space group is determined using the command 'RGS', we select among the monoclinic space groups the setting, where the c-axis corresponds to the unique axis (first setting in the 'old' tables).

Test for equal sites

TGL $u_1, u_2, z_1, z_2, tol, opt$;

One reason for a failure of 'SFND' may be that two different atoms occupy the same site. 'TGL' tests for

this situation. Distances are calculated from atoms having numbers from u_1 [3] to u_2 [*last atom*] in the parameter list to atoms having numbers from z_1 [3] to z_2 [*last atom*] in the parameter list within a sphere of *tol* [0.0001] Å except to themselves. If such pairs are found they will be printed. If *opt* [0] is entered with a value $\neq 0$ the command `IGFA 1` will be executed too in that case.

Idealize cell

ZIDL *tola,tolw,tole,opt* ;

If the unit cell is strongly distorted, the determination of the space group may fail (see 'SFND'). This can happen, because the symmetries found do not match sufficiently well to the unit cell provided. If e.g. there is a n-fold axis with $n > 1$, there must exist axes exactly perpendicular to this axis. If these are not found within the given set of tolerances, usually the cell is too distorted and needs to be idealized. The tolerances which have to be supplied are: *tola* = tolerance (in Angstroms) within which axes are treated to be equal, initial setting: 0.25; *tolw* = tolerance (angle in degrees) within which an angle found may differ from ideal angles e.g. 90°, initial setting: 4.0; *tole* = tolerance (distance in Angstroms) within which points are treated to lie in a plane, initial setting: 0.7; *opt* see below. 'ZIDL' performs the idealization as follows.

The symmetries found by 'SFND' are used to determine the crystal system (1=triclinic, 2=monoclinic, 3=orthorhombic, 4=tetragonal, 5=hexagonal (or trigonal), and 6=cubic). This number is found in the following way (rotation axes and inversion axes are treated in the same way):

6	There exist two or more fourfold axes, or a threefold axis has the magic angle of 54.74 degrees with a two- or fourfold axis.
5	There exists only one 6-fold axis or only one 3-fold axis.
4	There exists exactly one fourfold axis.
3	There exist more than one twofold, but no fourfold axis.
2	There exists only one twofold axis.
1	There exists no symmetry axis at all.

The (or one) axis with the highest number is selected according to the crystal system, and the plane perpendicular to the axis are searched for grid points. Centerings, if present, are taken into account. From the points found, two suitable ones are selected, and the unit cell transformed to this system. Now the resulting cell is idealized, i.e. angles nearly 60, 90 or 120 degrees will be set to exactly 60, 90 or 120 degrees. Furthermore, axes which should be equal according to the crystal system are set to the corresponding mean values. If *opt* = 0, a transformation of the idealized cell back to the original setting is performed. During these transformations, parameters (atom coordinates relative to the cell) and whatever symmetries are already present are kept unchanged, since only the cell parameters are idealized. In the case of an extremely distorted cell, it is advisable to specify *opt* = 2 (default). Then the transformation back will not occur. Of course, now symmetries and atom parameters must be transformed as well. If the cell volume has been reduced in the process, redundant atoms will be removed (see 'RPSY') and symmetries may be removed (see 'EDL'); in case of an enlargement, centering symmetries will be introduced (see 'C'). If cell reducing symmetries are encountered that do not correspond to a centering symmetry, e.g. $1/2 + x, y, z$, the cell will be reduced. If *opt* = 1 is entered, these actions are not performed automatically, i.e., the user needs to take care of changes of the cell volume himself.

Example:

During the investigation of the energy landscape of MgF_2 the following configuration was encountered (KPLOT input):

```
ndlg
z 3.096 4.020 5.34 89.82 90.03 89.81
ae 2
se 1
atom Mg 1 .3288806 .7613257 .9816873
atom Mg 2 .8304403 .7591122 .4817613
atom F  1 .3300893 .9868472 .6475843
atom F  2 .8282566 .9870328 .1482460
atom F  3 .3307128 .5330645 .3153760
atom F  4 .8298074 .5333360 .8154464
clse;dlg
```

Using 'SFND' the following symmetries were found:

```
Tolg: 0.1000 Tols: 0.2500 Tolt: 0.2500 Ref: MG
2-axis [ 556 ]
2-axis [ 565 ]
2-axis [ 655 ]
A:  2 R: 655 P:  0.8289  0.2613 -0.0183  G:  0.0000  0.0000  0.0000 ( 0.02666)
Z 1    0.32888   0.26133   0.48169 (0.005418)
Z 2    0.07966   0.26022   0.23172 (0.000756)
T 1    0.50000   0.00000   0.50000 (0.010147)
```

A translational symmetry T1 was found which should also be taken into account when searching for symmetry axes. Here it is introduced as a centering symmetry:

```
GTY B  (Introduce T1 as symmetry)
RPSY * * 0.5 -1  (Remove redundant atoms)
```

The following atoms remain:

No	Name	x	y	z
3	Mg 1	0.329660	0.760219	0.981724
4	F 1	0.329173	0.986940	0.647915
5	F 3	0.330260	0.533200	0.315411

Entering 'SFND' again, the following output is issued:

```
6-axis [ 545 ]
2-axis [ 455 ]
2-axis [ 456 ]
2-axis [ 556 ]
2-axis [ 656 ]
Centering symmetries were used. The cell might possibly have to
be reduced.

A:  3 R: 545 P:  0.8297  0.7602  0.4817 G:  0.0000  0.0000  0.0000 ( 0.00337)
A:  2 R: 455 P:  0.5797  0.7602  0.7317 G:  0.5000  0.0000  0.0000 ( 0.00256)
A:  2 R: 456 P:  0.5797  0.7602  0.7317 G:  0.5000  0.0000  0.5000 ( 0.00229)
A:  2 R: 656 P:  0.8297  0.7602  0.4817 G:  0.0000  0.0000  0.0000 ( 0.00292)
Z 1    0.32966   0.26022   0.48172 (0.000975)
Z 2    0.07966   0.26022   0.23172 (0.000975)
T 1    0.50000   0.00000   0.50000 (0.001087)
```

In this situation it is advisable to run 'ZIDL' and rerun 'SFND', in order to pass valid data to the program RGS. The cell will be idealized according to type 5, and RGS will find space group P$\overline{3}$m1.

Search symmetry and idealize

SSI *tols,tolt,tolg,ref* ;

Assuming that the structure is given in a triclinic setting, the steps described above can be performed automatically with the macro 'SSI'. The parameters *tols*, *tolt*, *tolg* and *ref* are used just as in 'SFND' (note: different sequence of the parameters in 'SFND'!). If *tolt* is not specified or the value 0 is entered, then the value for *tols* is applied. The following steps are included in 'SSI':

1. If the structure is not triclinic, it is made triclinic.
2. 'SFND' with the given parameters, together with $n = 3$ and $opt = 0$.
3. If translational symmetry is found, it is imported using 'ITS', followed by a cell reduction. Afterwards, step 2 is applied again.

4. If no translational symmetry is detected, the cell is idealized according to the detected symmetries, using 'ZIDL'.

5. 'SFND' is repeated, using *opt* = 1. If additional symmetries are detected, the steps 4 and 5 are repeated. Once this procedure has finished, RGS can be used to determine the space group.

It may happen that RGS fails in deriving a space group from the symmetries found. The reason is usually a strong distortion present in the original structures. This can lead to inconsistent systems of symmetries. In particular, the mismatch can result in artificially small translation vectors leading to unphysical lattice constants in RGS. A possible way out is a change of the reference atoms to find a consistent set of symmetries.

Another problem is indicated by the message "No matching space group found" in RGS. In such a situation, a common cause is missing translational symmetry within the given tolerances. A recommendation is to increase the tolerances step by step up to ca. 0.5 (higher values have rarely proven to be effective).

If none of these simple procedures help, one should introduce the symmetries found separately, step by step (see 'STA' and 'ST'), and then adapt the atomic positions with 'RPSY'. Often the introduction of a single symmetry will be sufficient to allow a subsequent 'SFND' to find a consistent set of symmetries. Note that this may be an indication of pseudo-symmetry present in the system.

Search only in the special directions

SAR *type, direction, nmax* ;

Usually, the search for symmetries is performed by assuming that the symmetry elements can lie along all possible directions derived from the lattice. With the command 'SAR', this search can be restricted to certain directions and a maximum degree of the symmetry axis. A direction is entered as "code" (the translation part of a designator code, see p. 4), i.e., 556 corresponds to 001, etc. *nmax* is the maximal degree of the symmetry axes in that direction (default 6). For *type*, one of the following letters must be entered: A, E, or L. For A the 'direction' defines a rotation, roto-inversion or screw axis, for E, the 'direction' corresponds to the normal of a mirror plane or glide plane (i.e., only searches normal to the given axis are permitted). When entering L, the whole list of 'SAR' commands is erased. Note that several 'SAR' commands can be active at the same time and that all 'SAR' commands must be given before entering 'SFND'.

Example: NaCl is supposed to be transformed into the hexagonal subgroup R$\overline{3}$m. The two commands SAR A 666 and SAR E 666 restrict the search to the space diagonal and all vectors orthogonal to this axis. As a result R$\overline{3}$m is found as the space group of the (actually more highly symmetric) NaCl-structure.

Print list of special directions

OSAR ;

All presently saved 'SAR' commands are printed.

Delete symmetry direction(s)

SDEL $d_1, d_2, \ldots$;

The $d_1, \ldots, d_n$ are directions of symmetries found as given in the output of 'SFND' e.g. 556. Sometimes it is advisable to exclude certain symmetries found by 'SFND' before 'ZIDL' is run. This may be necessary, if due to distortions, e.g., twofold axes are found that are not perpendicular to each other. If now the program assumes the orthorhombic crystal class, 'ZIDL' will fail. Delete the unwanted directions in this case by 'SDEL'.
Example: `SDEL 456 752`

Idealize structure according type of symmetry

ISST *type,idir,pnt,gld* ;

It may happen that due to strong distortion an inconsistent set of symmetries is found by 'SFND' or 'SSI'. In this case RGS is unable to derive a space group. To overcome this problem the structure may be idealized somewhat by shifting the atoms in order to agree with the symmetry given.

The command 'ISST' is a macro, in principle, executing the following steps:

1. The structure is made triclinic.

2. The symmetry given is introduced using 'ST', where the parameter n (at 'ST') takes values 1, 2, ..., $|type| - 1$, at least 1. The direction *idir* has to be given as code, e.g. 556 instead of 0 0 1 (default: 556). The rest of the parameters have the same meaning as for 'ST' (see p. 11).

3. The parameters are reduced according to these symmetries ('RPSY').

4. The structure is made triclinic again.

Idealize axis

IDA *idir* ;

This command is a short form of 'ISST'. When 'SFND' is executed, the symmetries found are stored and may be addressed by the code *idir* (e.g. 556 instead of 0 0 1). Default: 556. If *idir* is given as a negative number $-n$, the nth center of symmetry is used for idealization.

Search for a space group

RGS *opt* ;

In version 8.4.0 the program RGS has been integrated. But the mechanism of data transfer is still the same as before. Using 'SFND' (or 'SSI') a file is created [`FOR007.DAT`] which is read in by RGS. The result is written to a file [`RGS.OUT`] which may be imported by KPLOT. In fact the command 'RGS' is the call to a sub-system.

An integer number *opt* may be used to control the behavior of RGS. $opt > 0$ [1] is interpreted as if the first RGS command would have been `G * 0` (automatic search). If $opt \leq 0$ is given, the sub-system RGS is started only, and further (RGS) commands may be entered by hand. Using $opt < 0$ RGS is terminated after the 'RTF' command. This allows the user to pass options to RGS because the rest of the line following the 'RGS' command is interpreted by RGS. *Example:* `RGS -1;FS;G * 0`

Additional commands regarding symmetries

Search mirror planes in molecule

SM $n_1, n_2, tol_x, tol_y, tol_z, nhl$;

Atoms currently in the code list are treated as a molecule. This set of atoms will be checked for mirror planes. To save computing time, a range n_1, n_2 should be given, defining a reference type of atoms. These atoms are used to define candidates of mirror planes, but the actual test of mirror symmetry is done for all atoms, of course. The numbers tol_x, tol_y, tol_z are tolerances in Angstrom with respect to the free coordinate system. *nhl* is the number of allowed atoms without an image (default: 0). If a mirror plane is found, the normal vector and the distance from the origin is printed. In addition, if the parameter in 'MORE' is set to 1, a list is printed containing the points mapped onto each other by the mirror plane together with the distance between the molecule and its mirror image. A trivial mirror plane, i.e. if the molecule is planar, is not checked. If more than 8 mirror planes are found, the process stops.

Search for centers of symmetry in the cell

ZZ n_1, n_2, s, tol, n_c ;

The code list is cleared and filled with atoms of one unit cell having atom numbers between n_1 and n_2 in the parameter list. s is the symbol of the type of atoms which is to be used as reference atoms (capital/small letters are treated as equal). Candidates for centers are the atoms themselves and the middle points between two reference atoms. These candidates are checked for all atoms (not only the reference atoms). If the symmetry is accepted, the center is stored in the parameter list with the name 'Z' and a number as an extension. Up to n_c centres are searched for. Defaults are: $n_1 = 3, n_2 =$ last atom, $s =$ name(n_1), $tol = 0.25$, and $n_c = 999$.

Note: The search is fastest, if s is of the type having the fewest atoms in the unit cell. The tolerance *tol* is used as follows: Every atom is mapped onto its image under inversion for a symmetry candidate. If for every case, an atom having an equal name is found in a sphere with radius *tol*, the center is accepted. The ESC key may be used to cancel the search.

Search for translational symmetry in cell

TSZ $[n_1, n_2], s, tol, n$;

With 'TSZ', one checks, whether there exists an additional translational symmetry in the unit cell for atoms in the parameter list between n_1 and n_2 (defaults 3, and last atom, respectively). This may e.g. suggest a transformation of the structure to a different cell. The atoms having the name s are used as reference atoms. Upper/lower letters will be treated as equal. In order to save computer time, it is advisable to select

a type having the fewest atoms in the unit cell. Default is 0.5 Angstrom for the tolerance. n (default 4) is the maximum denominator of the tranlational component.

The following steps are untertaken to find additional translational symmetry:

1. The code list is cleared and the content of one unit cell is loaded, having numbers between n_1 (default: 3) and n_2 (default: last atom). A skin with a thickness appropriate for the given tolerances is added.

2. Within this set all vectors that can be constructed from atoms of the type s are determined. From these, those are selected for which the absolute values of the components are less or equal 0.5.

3. These vectors are added to and subtracted from all atoms in the code list, and one checks, whether at these positions there exist atoms in the code list having the same name. If this holds for all atoms, a valid translation vector has been found. The accepted vectors are stored in the parameter list with names T , and extensions 1, 2, etc.

Note that not only the vector t but also $-t$, $1-t$, $1+t$, etc. are translation vectors. See also 'SFND'.

Find cell

ZFND *tol, opt, ref* ;

Defaults: *tol*= 0.25, *opt*= 3 und *ref*=' '. The current structure is searched for additional translation vectors, and, if such vectors are found, is transformed to a smaller unit cell. In contrast to the command 'TSZ', all possible vectors are searched for, not only the ones that correspond to the rational linear combination of cell vectors of the current cell. This search follows the following strategy: A candidate for a new translation vector is a vector, which connects an atom with another atom of the same working name *ref*. This vector is then added to the position vector of every atom in the unit cell, and it is checked, whether at the endpoint of the resulting vector another atom with the same working name is found, within a tolerance *tol* in Å. If this is the case, the candidate vector is a translation vector.

The translation vectors detected are sorted by length, and the shortest ones are employed in an attempt to construct a new unit cell. If this is possible, the current cell is transformed to the new cell.

The option *opt*= n_1+n_2 is the unique sum of two numbers ($n_1 = 0, 1$, $n_2 = 0, 2$, and thus *opt*= $0, 1, 2, 3$), and it controls which additional commands will be applied after a successful transformation. $n_1 = 1$: The command 'RPSY' is executed, which removes all duplicate atoms that might be present due to the transformation; $n_2 = 2$: The command 'NPZ' is executed, which normalizes all atom coordinates such that they lie between 0 and 1.

In order to save computation time, one should select for *ref* the name of an atom type that is only found rarely in the structure. If an empty string or no value is entered for *ref*, the program chooses a name on its own.

In order to be able to apply the command 'ZFND', the structure has to be given in the space group P1.

Test symmetry

TSY $t_1, s_{11}, s_{12}, s_{13}, t_2, s_{21}, s_{22}, s_{23}, t_3, s_{31}, s_{32}, s_{33}, z_1, z_2, tol$;

Using 'TSY', one can check, whether an assumed symmetry given in the form $t_1, \ldots, s_{33}$ holds for a given structure. The coding is analogous to the one in the command 'S'. This symmetry is applied to all atoms in the code list (initial set). Around points calculated in this way, spheres with the radius *tol* (default: 0.1) Angstrom are searched for atoms having numbers between z_1 and z_2 (in the parameter list). The atoms found are added to the code list. It is *not* checked, whether the new atoms are already in the code list, and duplicate codes may occur. The printout contains the initial set together with the transformed coordinates and the atoms found, if any.

Test symmetry after idealistation

TSYI z_1, z_2, tol ;

This command works in principle like 'TSY'. The symmetry used here is either the one last found during the previous idealization (see 'IDL') or the one defined by the command 'ST'.

Normalizers

When one determines the symmetry of the symmetry elements of a space group, one usually generates a supergroup of the sapce group (c.f. Int. Tables A, chapter 15). Those symmetries of the supergroup that are not part of the space group are of particular interest. E.g., they are needed when several arrangements of an asymmetric unit must be considered for a comparison of two structures. For the application of these additional symmetries, a specific field is provided, which is defined via a standard symmetry list.

Copy normalizers

CNRM;

The content of the symmetry list of the current structure is copied to the list of normalizers. This list can contain up to 24 symmetries.

List normalizers

ONRM;

The symmetries stored in the list of normalizers is printed.

Transform according to a normalizer symmetry

TNRM nr, z_1, z_2, u_1;

The symmetry, which is stored at position nr in the list of normalizer symmetries, is applied to those atoms that are stored in the parameter list on the positions $u_1, \ldots, u_1 + z_2 - z_1$. The new coordinates overwrite the entries of the atoms on the positions $z_1, \ldots, z_2$ of the parameter list.

Example:

The crystal structures of the three compounds $BiEAlCl_4$ with E=S, Se and Te have been determined. All three compounds crystallize in the space group $\mathbf{I}\bar{4}$ with very similar lattice constants. The compound containing S was isotypical to the one containing Se; however, the one with Te could not be brought in agreement with the Se-containing compound, although they looked very similar.

In order to determine, whether the Te- and the Se-compound were isotypical, *all* possible asymmetrical units of the Te-compound need to be generated, and compared with the Se-compound. The Int. Tables provide the following additional generators for the normalizer group of $\mathbf{I}\bar{4}$: $(1/2+x, y, 1/4+z)$, $(-x, -y, -z)$ and (y, x, z). These symmetries have to be placed into the list of normalizers.

```
INIT 1           ! The lattice constants are irrelevant.
C 1/2 0 1/4      ! Adding a centering automatically produces the missing
                   symmetries for a group
GTY Z            ! Missing symmetries added automatically
SY 'Y,X,Z'       ! Take care: No other symmetries added automatically
GTY L            ! Missing symmetries added automatically
CNRM;
```

Now the two structures have to be loaded (as 1 and 2). In this example the file se.kpl contained instructions for generating a complete picture while in te.kpl beside the crystallographic parameters only suitable plot commands were present.

```
RSTR 1;
IMP TE;
NS;
IMP SE;
NS;
```

A second copy is needed for the Te-compound, in order to serve as starting point for generating the various variants. Furthermore, we want to be able to generate a plot of both structures at the same time, in order to judge the degree of similarity.

```
CPG 3 9
PLT 2
```

In order to generate the various variants automatically, one might want to write a small macro, e.g.

```
MACR 1
SETC %1
CC
ACI 2 266601
TNRM & 3 9
ECHO 'NORMALIZOR &'
AUF 3 3 3 3 1
AUF 4 4 4 4 1
AUF 5 5 5 5 1
AUF 6 9 6 9 1
GG
WAIT
IMLE 16 2
ENDM
```

If one nows starts the macro with the command '`1 1;`', then all possible asymmetric units are generated, and the number of the normalizer symmetry used is printed. Furthermore, one searches around the positions of the atoms in the Se-compound for corresponding atoms of the Te-compound. Finally, a stereoplot is displayed. Only in the case of full agreement, every atom finds a corresponding partner.

Valence sums

Compute valence sums

VSUM $u_1, u_2, z_1, z_2,$ *opt* ;

For a selected number of anions (O, F, Cl, Br, I, S, Se, Te, N, P, As und H) valence sums can be computed according to the method of N.E. Brese and M. O'Keefe (Acta Cryst. (1991) **B47**, 192–197) using the formula (i and j denote atoms in a crystal structure)

$$V_i = \sum_j \exp((R_{ij} - d_{ij})/0.37).$$

R_{ij}: Tabulated value
d_{ij}: Distance between anion and cation in a pair

First, the oxidation stages of the participating elements have to be entered (see below). The valence sums are computed for those atoms, which are in the parameter list between numbers u_1 [3] and u_2 [last atom in the parameter list]. Possible coordinating atoms are those with numbers from z_1 to z_2 in the parameter list [same defaults as above]. The maximal distances are taken from the table which has also been employed in the command 'FM' (c.f. page 22). The final result should yield valence sums whose values are close to the oxidation states of the corresponding elements. The output are the computed valence sums and the deviations from the expected values. If for *opt* [0] a number differing from 0 is entered, the valences for the individual bonds are also listed. If the prescribed distances lead to the inclusion of anion – anion- or cation – cation-pairs, these are ignored in the calculation.

Example: Brookite has a titanium and two oxygen atoms in the asymmetric unit. This could lead to the following dialogue:

```
>get brookite
>oa,
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  Ti  1     0.128000  0.098000  0.863000   0.3000       1
    4  O   1     0.008000  0.147000  0.182000   0.3000       2
    5  O   2     0.229000  0.110000  0.530000   0.3000       2
```

```
>ox ti 4
>ox o -2
>vsum 3 5 3 5
Valence sum of Ti  1    No.    3       4.119 ( 0.119 =     3.0%)
Valence sum of O   1    No.    4       2.146 ( 0.146 =     7.3%)
Valence sum of O   2    No.    5       1.974 ( 0.026 =     1.3%)
>end
```

Definition of an oxidation state

OX *name*, s, b, n_1, n_2, k ;

For the element with the name given in the command, the oxidation state s [no default] and an occupation factor b [1] are defined. With parameters n_1 [1] and n_2 [9999] the validity of this definition can be restricted to a subset of the parameter list, e.g. if an element is present in several oxidation states. If for k a value larger than 0 is entered, the entry at the corresponding location is overwritten. Currently, the table can contain twenty entries.

Show/delete list with oxidation states

OOX; and OXL n_1, n_2; , respectively

Using 'OOX' gives the list of oxidation states, and using 'OXL' deletes the complete list or parts of it.

Entry of user-defined R_{ij}-values

RIJ $Name_1, ox_1$, $Name_2, ox_2, R_{ij}$, *pos*;

Usually, the R_{ij}-values for the calculation of the valence sums are taken from the tables given by O'Keefe. However, one can define one's own parameters, which take precedence in such a case. If for *pos* [0] a value larger than 0 is entered, the entry in the corresponding location is overwritten. Currently, the table can contain 20 entries.

Print/delete list with R_{ij}-values

ORIJ; and RIJL; , respectively

Using 'ORIJ' the list with the user-defined R_{ij}-values is printed, using 'RIJL' deletes the complete list.

Computation of R_{ij}-values

RIJB $u_1, u_2, z_1, z_2, R_{ij}$;

For a given structure, one can let the program compute "optimal" R_{ij}-values. However, this gives only useful results, if the atoms which are located in the parameter list between u_1 and u_2, are coordinated by atoms belonging to only one kind of ion, which are in the parameter list between z_1 and z_2. The distances are determined in the same way as with the command 'VSUM'. Since this computation constitutes a non-linear problem in general, an iteration method is employed, which uses the initial (Default: pre-defined [2.0]) R_{ij}-value as starting value of the optimization.

Show R_{ij}-value

RIJZ $Name_1, ox_1$, $Name_2, ox_2$;

The value for the given cation-anion pair with its oxidation states ox_1 and ox_2 is searched for in the table by O'Keefe, and printed if available.

Interface to other programs

Load SHELXL file

SXL *name* ;

A file with the given *name* containing instructions for the program SHELXL is read, and the relevant information extracted. The logical number of this file is *ntxr*1 (see 'EAE'). It is recommended to set '`AE 2`' before having KPLOT read such a file. But it is done automatically in case there are no atoms in the parameter list.

Generate SHELXL atom instructions

GNSX $s_1, n_1, s_2, n_2, \ldots$;

All codes of the code list which agree with symbols $s_1, \ldots$ are written to the file with the logical number *ntpch* as atom instructions for the program SHELXL. The atom with the symbol s_i is given the pointer n_i (to the SFAC instruction). The atom symbols (the first two letters) are taken over and are numbered continuously beginning at 1. The site occupation factor is calculated from the symmetries present.

Generate SHELXL atom instructions

SXAT n_1, n_2 ;

The atoms in the parameter list in the given range (defaults: 3 and last atom) are written to the file with the logical number *ntpch* as atom instructions for the program SHELXL. The atom names for SHELXL are created from the names and the extensions in the parameter list, eliminating all blanks in the process. The site occupation factor is calculated from the symmetries present. All pointers are set to 1.

Generate SHELXL-Job

GSXJ n_1, n_2 ;

This command is similar to 'SXAT', but here not only the atom-instructions in the given range n_1 [3] to n_2 [last atom in the parameter list] are written to the file with the logical number *ntpch* but also all information available to generate (part of) a SHELX job (TITL, LATT, SYMM, SFAC, UNIT, and atoms). The symmetries should be inspected anyway, because there may be errors if they are not taken from the internal list of space groups.

Flag for GSXJ

GSXF n ;

In KPLOT-version 9.4 or higher, the content of the ZERR line in a SHELXL file (standard deviations) is also stored when reading in a SHELXL file. But these values are not automatically taken into account when a cell transformation is performed. If these standard deviations should be transformed in addition, the following conditions must be met: (1) No structure is stored as a background structure, and (2) the flag for 'GSXJ' must have been set by entering the command `GSXF 1` (Default: $n = 0$) *before* reading in the SHELXL file via the command 'SXL'.

Generate file for the program SYMMOL

SYML i_1, i_2, tol_1, tol_2 ;

The program SYMMOL (Tullio Pilati and Alessandra Forni, J. Appl. Cryst. (1998) 31, 503-504) finds the maximal symmetry group in a cluster. In order to generate a file for SYMMOL, proceed as follows: The molecule (or cluster) is compiled in the code list. Setting all masses equal to one, the center of mass is calculated ('SPC') and chosen as origin of the free coordinate system ('KUP'). The numbers $i_1, \ldots, tol_2$ are the specifications for SYMMOL at the second line for `INDWGH,INDTOL,DCM,DCME` (see SYMMOL.MAN). Defaults are: `1 1 0.25 0.25`.

Generate atoms including special site parameters

GASP *name* ;

A file is generated containing the unit cell constants, and the atoms in the asymmetric unit. With each atom both information characterizing the site and the symmetries of that site are given. The file is constructed as follows:

The first line contains the title in `FORMAT(18A4)`. The second line (`FORMAT(3F10.5,3F10.4, 2I6)`) contains the lattice constants $a, b, c, \alpha, \beta, \gamma$, the number n of atoms in the asymmetric unit, and the number of the space group (see page 10). This is followed by n blocks, consisting of the following lines:

The first line (`FORMAT(2A4,2X,3F10.5)`) contains the name, the extension, and the parameters x, y, and z of the atom in the asymmetric unit. The next line (`FORMAT(9I4)`) contains nine numbers: the multiplicity *mult* of the site; a code *icode* characterizing the type of the site (see below); three numbers f_1, f_2, f_3 having the values 1 if the coordinate x, y, or z is fixed, respectively, and 0 otherwise; the final three numbers i, j, and k contain the value (in $n/24$) of the fixed coordinate(s).

icode equals one of the numbers $1, \ldots, 7$, which are interpreted as follows (here, x, y, and, z specify the first, second, and third free parameter):

icode=1 x, y, z

icode=2 x, x, y

icode=3 $x, -x, y$

icode=4 x, x, x

icode=5 $0, x, x$

icode=6 $0, -x, x$

icode=7 $x, 2x, y$

Next, *mult* lines follow in `FORMAT(3(F9.6,3F4.0))`, each containing 12 numbers $t_1, s_{11}, s_{12}, s_{13}, t_2, s_{21}, s_{22}, s_{23}, t_3, s_{31}, s_{32}, s_{33}$. These numbers define the rotation matrix and the translation vector of the symmetry used to generate a new atom from the one given in the first line of the current block. The meaning of these numbers is the same one as for the command 'S' (see page 6).

Example: Rutile crystallizes in space group $P4_2/mnm$ (136) with two sites in the asymmetric unit:

```
Ti  1     0.000000  0.000000  0.000000   2 a   0,0,0
O   1     0.304910  0.304910  0.000000   4 f   x,x,0
```

Using the command 'GASP', the following file is produced:

```
Rutile
   4.59370   4.59370   2.95870   90.0000   90.0000   90.0000     2   136
Ti  1        0.00000   0.00000   0.00000
   0   2   1   1   1   1   0   0   0
 0.000000  0.  0.  0. 0.000000  0.  0.  0. 0.000000  0.  0.  0.
 0.500000  0.  0.  0. 0.500000  0.  0.  0. 0.500000  0.  0.  0.
O   1        0.30491   0.30491   0.00000
   1   4   2   0   0   1   0   0   0
 0.000000  1.  0.  0. 0.000000  1.  0.  0. 0.000000  0.  0.  0.
 0.000000 -1.  0.  0. 0.000000 -1.  0.  0. 0.000000  0.  0.  0.
 0.500000 -1.  0.  0. 0.500000  1.  0.  0. 0.500000  0.  0.  0.
 0.500000  1.  0.  0. 0.500000 -1.  0.  0. 0.500000  0.  0.  0.
```

Generate atom instructions for the NRC system

NRCA n_1, n_2 ;

The atoms in the parameter list in the given range (defaults: 3 and last atom) are written to the file with the logical number *ntpch* as atom instructions for the program NRC. The atom names are created from the names and extensions in the parameter list, eliminating all blanks in the process. If e.g. the program MISSYM is to be run, proceed as follows:

- Create a .CD file using the program CDFILE;
- write the atom parameters to some file using 'NRCA';
- import this file to the .CD file, using the program CDEDIT;
- now you are ready to run the program MISSYM.

Generate MISSYM input

MSIN n_1, n_2 ;

As an extension to the command 'NRCA', with this command four files are created:

1. CDFILE.DAT Besides the dialog expected by the program CDFILE the cell constants and the atomic symbols together with their number in the unit cell are stored. The Hermann-Mauguin symbol is reset to P 1. If this is changed later, the lattice constants may have to be adapted, e.g. in case of an orthorhombic cell the input for the angles should be removed. The .CD file then simply will be generated by the DOS command:
   `CDFILE <CDFILE.DAT`
2. CDEDIT.DAT Corresponding to the given range the dialogue input for CDEDIT is generated. Input:
   `CDEDIT <CDEDIT.DAT`
3. KPATMS.DAT This file contains the atoms to be imported.
4. MISSYM.DAT Here the dialog for the program MISSYM is stored. Input:
   `MISSYM <MISSYM.DAT`

Of course, these three DOS commands can be collected in a .BAT file. Note that the file TEMP.CD which is used by default has to be deleted at the beginning.

Read CRYSTIN formatted file

LCFF *opt* ;

A file generated by the retrieval system CRYSTIN (should have extension .CRY) is read by KPLOT, and all relevant data are extracted. If $opt = 0$ (default), all arrays are cleared before reading and two dummy atoms ORGN and NULL are stored in the parameter list. If $opt \neq 0$, the CRYSTIN information will be appended to the current parameter list.

Save CRYSTIN formatted file

SCFF *name* ;

A file with the given *name* containing the relevant crystallographic data is written in CRYSTIN format.

Read CIF-File

LCIF (*name*) ;

A CIF file generated in the crystin format is read, and the data needed for KPLOT are extracted. If the file *name* [no default] contains the data of several structures, as is usually the case, one can read the next entry with the command 'LCIF', without having to enter the name of the next entry. *Note:* if the command 'LCIF' is used within a macro, it can happen that the read-process is terminated, e.g. if a space group symbol is not recognized. In such a case, one can enter 'LCIF' and continue reading the remainder of the file. However, one needs to deal with the problem caused by the missing/wrong entry "in person" afterwards.

Generate ORTEP input (format free)

ORTF opt_1, opt_2,*pkz, type* ;

To the unit with the logical number *ntpch* (see 'EAE') a file is written, which can be processed by the program ORTEP. From the information available, instructions are generated to ensure that the set or parameters expected by ORTEP is complete. The following lines are written:

1. title;
2. unit cell;
3. symmetries;
4. atom parameters; if $opt_1 = 1$, anisotropic temperature factors are written (if available); if $opt_1 = 0$ (default), radii are used instead (ORTEP type 7);
5. instruction 201 starting the plotter;
6. instruction 301 defining the drawing area;

7. one or more instructions 401 containing all codes present;

8. instruction 502 to obtain the current orientation;

9. instruction 604 containing the current scale factor in brackets if desired to change to 601;

10. instructions 700 - 800 corresponding to PK, VB, VBS, VBR, VD, ... The default value 714 may be changed by input of *pkz*, e.g. 711;

11. the instruction 503 (stereo rotation) and instruction 1211 (execute plot commands) if $opt_2 = 1$ (default: 0);

12. if labeling commands are present (BA, BT), instructions 1400 or 900 are generated, if *type*=0 (default) or *type*=1, respectively;

    The following instructions (14) and (15) are issued only if $opt_2 = 1$ (default: 0).

13. Instruction 202 to shift drawing field;

14. instructions (11) - (12);

15. instruction -1 to terminate the job.

Generate THEO input

THEO $s_1, n_1, s_2, n_2, \ldots$;

The program THEO implemented in the program package of STOE can calculate powder patterns from hypothetical structures. In order to generate an input for this program, enter the symbols s_j together with the numbers n_j, how often an atom of type j appears in the unit cell. Default for all n_j is 1. *Example:* `THEO Na 4 Cl 4`. The program searches in the parameter list for the atom symbols and writes them to file. The site occupation is calculated automatically from the symmetries present. For the rest of the input file required by THEO, default values are used which may be changed by the following command.

Set THEO parameters

THPR $w_1, w_2, Sym, U(iso), Calc(6), Prof(3)$;

Parameters used with 'THEO' can be redefined using 'THPR'. w_1 and w_2 are the specifications on the WAVE instruction (defaults: `CU A1`); Sym is the Hermann-Mauguin symbol (in quotes, e.g. `'P m 3 m'`), default is the KPLOT HMS symbol if available (c.f. table given with command 'HMS'); $U(iso)$ is the isotropic temperature factor; $Calc(6)$ are the six values for the CALC instruction: 2θ min, max, step, mode (1 = transmission, 2 = reflection, 3 = Debye-Scherrer), monochromator (1 Graphite, 2 Germanium), $\mu \cdot t$ (absorption factor); $Prof(3)$ are the three values of the PROF line: *nprof*, h_1 and h_2. The meaning of these symbols is explained in the user manual Stadi P in chapter 18.

Input file for Lazy Pulverix

LZIN *name* ;

A file suitable as input for the program "Lazy Pulverix" is generated with the name *name* [`FOR002.DAT`]. This program computes theoretical powder diffractograms.

Generate SCHAKAL input

SHKL n ;

To the unit with the logical number *ntpch* (see 'EAE') a file is written containing data for the program SCHAKAL. This program plots models as shaded calottes. Two versions of SCHAKAL exist, which require different inputs; with n one chooses between the versions: if $n = 0$ (default), a file for SCHAKAL 66 is generated, otherwise for SCHAKAL 88. The atoms are taken from the code list. The coordinates are calculated from their triclinic coordinates with respect to the point currently in the point register, i.e. the triclinic coordinates are subtracted from the triclinic coordinates of the codes. In order to avoid problems positioning the model, calculate via 'SPC' the center of gravity before using 'SHKL'.

Generate DLS input

MDLS n_1, n_2, opt, s, n ;

The program DLS (Ch. Baerlocher, A. Hepp and W.M. Meier 1977) is used to refine crystal structures by geometric restraints. The generation of an input file is supported by KPLOT. The following lines are written to the file *ntpch*:

1. Title line
2. DLS-76 - line with standard settings
3. lattice constants
4. Hermann-Mauguin symbol, if known
5. atom parameters,
6. DISTAN instructions
7. END and FINISH

The atom parameters are taken from the code list (!). Atoms are selected having atom numbers (in the parameter list) n between n_1 and n_2. If the code is $n55501$, the program checks, whether the atom occupies a special position. In this case, the site occupation is calculated and written additionally on the same line. This item (see example) must be replaced later by a pseudo symmetry, e.g. `X,Y,0`. If the code is not $n55501$, a SYMEQ instruction is created which already contains the pseudo symmetry.

If $opt = 0$ is entered (default), the SYMEQ instructions contain the same name and extension as the original atoms. This has to be changed later by hand (using a text editor). If $opt = 1$ is entered, the SYMEQ atoms receive the same name as the original atoms, but the extension is concatenated from symbol s (default: X) and a number which begins with n (default: 1) and is incremented with each new SYMEQ atom. These atoms are stored as new atoms in the parameter list.

Example: The molecule C28 was constructed and (based on its symmetries) transformed to space group P$\bar{4}$3m (No. 215). The KPLOT input file is:

```
NDLG
T 'C28  14.11.1996'
Z   10
HMS 'P-43M   '
AE 2
ATOM C 1 -0.035888 -0.035888 -0.226020
ATOM C 2 -0.163446 -0.163446  0.005040
ATOM C 3  0.133084  0.133084  0.133084
EAE 5 ; DLG
```

We have a molecule which may be simply generated by 'FM':

```
FM;
MDLS;
```

Besides the atoms originally entered, the parameter list now contains the following additional atoms:

```
( 6) C X1  0.035888  0.035888 -0.226020
( 7) C X2 -0.163446  0.005040 -0.163446
( 8) C X3  0.005040 -0.163446 -0.163446
( 9) C X4 -0.035888 -0.226020 -0.035888
(10) C X5 -0.226020 -0.035888 -0.035888
(11) C X6 -0.133084 -0.133084  0.133084
(12) C X7  0.005040  0.163446  0.163446
(13) C X8  0.163446  0.005040  0.163446
(14) C X9  0.163446  0.163446  0.005040
```

The DLS file written so far (somewhat condensed) looks as follows:

```
TITLE  C28  14.11.1996
DLS-76        10             0
CELL   TRIC       10.0000 10.0000 10.0000 90.00 90.00 90.00
HMS    P-43M    (NO.   215)
ATOM   C1    -0.03589-0.03589-0.22602      Special pos. ( 12)
ATOM   C2    -0.16345-0.16345 0.00504      Special pos. ( 12)
ATOM   C3     0.13308 0.13308 0.13308      Special pos. (  4)
SYMEQ  C1     CX1    -X,-Y,Z
SYMEQ  C2     CX2    Y,Z,X
SYMEQ  C2     CX3    Z,X,Y
SYMEQ  C1     CX4    Y,Z,X
SYMEQ  C1     CX5    Z,X,Y
SYMEQ  C3     CX6    -X,-Y,Z
SYMEQ  C2     CX7    Z,-X,-Y
SYMEQ  C2     CX8    -Y,Z,-X
SYMEQ  C2     CX9    -X,-Y,Z
DISTAN C1     CX3       1.01507   1.00000           2
DISTAN C1     CX4       1.47857   1.00000           2
DISTAN CX3    CX4       2.11190   1.00000           3
DISTAN CX4    CX5       2.38275   1.00000           3
DISTAN C2     CX9       1.35052   1.00000           2
DISTAN CX1    CX2       2.68887   1.00000           3
DISTAN CX1    CX9       2.15953   1.00000           3
DISTAN CX6    CX7       2.24020   1.00000           3
END
FINISH
```

Following the International Tables one sees that atoms C1 and C2 are on the sites 12 m, and C3 is on site 4 e. The items 'Special pos. ...' must be replaced by symmetry entries:

```
ATOM   C1    -0.03589-0.03589-0.22602      X,X,Z
ATOM   C2    -0.16345-0.16345 0.00504      X,X,Z
ATOM   C3     0.13308 0.13308 0.13308      X,X,X
SYMEQ  C1     CX1    -X,-Y,Z
...
SYMEQ  C2     CX9    -X,-Y,Z
```

After the 'SYMEQ' instructions there follow the desired distances (2 in column 54, this column is ignored by DLS) and angles. Distances are described by (1,2) while angles are described by (1,3) distances with DLS. Atoms are treated to be connected by bonds if, when drawn a bond is produced (in the graphic). Only those are selected where at least one atom appears in an 'ATOM' instruction. Atoms are treated to be connected by (1,3) bonds if bonded to the same atom but having no bond to each other (in the graphic). The values for the distances are calculated using the current coordinates where symmetrically equivalent DISTAN instructions are suppressed. These values must be replaced by idealized ones:

```
DISTAN C1     CX3       1.39000   1.00000           2
DISTAN C1     CX4       1.39000   1.00000           2
DISTAN CX3    CX4       2.24000   1.00000           3
DISTAN CX4    CX5       2.40700   1.00000           3
DISTAN C2     CX9       1.39000   1.00000           2
DISTAN CX1    CX2       2.40700   1.00000           3
DISTAN CX1    CX9       2.24000   1.00000           3
DISTAN CX6    CX7       2.24000   1.00000           3
```

In this case the value of 1.39 is taken for C-C distance, 2.24 for the (1,3) distance in a pentagon, and 2.407 in a hexagon. The DLS job produces the following result:

```
ATOM C 1       -0.052533   -0.052533   -0.223566
ATOM C 2       -0.165359   -0.165359    0.005093
ATOM C 3        0.145498    0.145498    0.145498
```

There is another way to make specifications for idealized distances and angles using the commands 'AR' and 'WR'.

Distance restraint

AR s_1, e_1, s_2, e_2, d ;

While generating DISTAN instructions a lookup is performed in a table generated with 'AR' commands. If the symbols s_i are found, the calculated distances will be replaced by the stored ones. If for e_1 and/or e_2 blanks are entered (default), the specification is used for all symbols. Therefore the special cases have to be given first. Default for d: value calculated last by 'L'. If $d = 0$ is given, the last value will be copied.

Angle restraint

WR $s_1, e_1, s_z, e_z, s_2, e_2, d$;

As explained at 'AR' for the (1,2)-distances, one may specify (1,3) distances with 'WR'. Note that not the angle, but the distance has to be given.

This means for the example above: All angles at the atoms C1 ... C3 should correspond to ideal pentagons except CX4 - C1 - CX5 and CX1 - C2 - CX2. For efficiency, 'AR'- and 'WR'-commands should be given in the order shown below.

```
AR  C *  C *  1.39
WR  C X4 C 1  C X5 2.407
WR  C X1 C 2  C X2 0        (Use last distance)
WR  C *  C *  C *  2.24     (General case)
```

Print AR- and WR-lists

OR ;

The distance and angle restraints stored will be printed.

End of AR/WR-list

ARE bzw. WRE n ;

One can set the number n of the end of the lists arbitrarily. The purpose is to make corrections in the list. How to perform such a task is described in detail at 'AE'.

Restart the program

RSTR *opt* ;

KPLOT will be restarted. If *opt* is given as zero (default), all open files are closed. Else, all files remain open.

Initialize

INIT *spg* ;

There are situations where the details of the unit cell are essentially irrelevant. Nevertheless some basic definitions may be needed to be able to do certain calculations. Using 'INIT' one can conveniently perform the following initialisation:

- All lists are cleared (RSTR).
- The unit cell is defined (Z 10 10 10 90 90 90).
- The space group *spg* is introduced (given as number or symbol, default: P1). If it is a hexagonal (or trigonal) space group, the angle $\gamma = 120°$ and $c = 11.547$ will be set (in order to preserve $V = 1000$ Å^3).
- Two dummy atoms (ORGN and NULL) will be placed into positions 1 and 2 of the parameter list.

Terminate program

END resp. QUIT ;

The program is terminated.

Macros

In order to simplify sequences of commands frequently entered, one may collect them into macros. A macro is defined by the command MACR n, where n is a number between 1 and 9 (no default for n). The line 'ENDM' indicates the end of a macro.

Example:

```
MACR 2
DK 2 3
BF
EPU
ENDM
```

The macro is executed, if the number n is given as command. In the example above, entering a '2' on the command line of KPLOT will result in the three commands stored in this macro being executed.

In order to gain more flexibility, parameters may be passed to the macro. If in the example above the rotation angle is supposed not to be a fixed value (3), but is to be set when calling the macro, the 'DK' command may be coded as follows:

```
DK 2 %1
```

The expression "%1" refers to argument no. 1 of the macro. The macro (named 2) is now called by the following command, where 3 is the value of argument no. 1:

```
2 3
```

Up to eight parameters may be given as input for a macro. One parameter may consist of up to 12 characters. Note that the arguments are passed to a macro as text, and that their initial values are saved. Thus they can serve as default values of the arguments when the macro is called a second time. However, one can also generate default values for the arguments by using the command:

DFMA arg_1 arg_2 ....

This feature may be useful when starting KPLOT e.g. in the setup file KPLOT.STP.

Macros may also contain loops. For this purpose there exist several commands that can be used to control the sequence of execution of commands within a macro. Note that these commands are only valid within a macro. For this purpose, the lines in a macro are numbered (automatically) by KPLOT. The initial line "MACR n" is assigned the number 0.

Example: The macro above is supposed to be executed ten times.

```
MACR 2
SETC 10       (Set count to 10)
DK 2 3        (line 2)
BF
EPU
DMNZ 2        (Decrement count; if count)
ENDM          (larger zero, move to line 2)
```

In connection with group commands, e.g. 'VG', 'DG' etc., the following command is useful:

`MLEG dist line1 line2 line3 tol` (Move if Less or Equal or Greater than *dist*)

The goal of this command is to provide a "case"-type command, where depending on the value of a variable compared to *dist*, one can move to three possible lines within a macro. This is particularly useful for terminating a macro, if unreasonable distances occur.

For example, each calculation of distances by 'ATAB' deposits the shortest distance encountered, and similarly the commands 'L', 'AL' etc. deliver their results. This value is now compared with *dist*. First it is checked, whether the deposited result is equal to *dist* within the tolerance *tol* (Default: 0.000001). If so, the macro will be continued at *line2*; else, if the result is less or greater than *dist*, one continues on line *line1* or line *line3*, respectively. A zero may be specified for *line1*, *line2*, or *line3*. This means that no action is taken, and the next command in the macro will be executed.

There are situations where floating point operations are needed inside a macro. Two commands are provided: 'FADD' and 'FMUL' for addition and multiplication, respectively. Both commands operate on one macro argument *Arg-No.* which is replaced by the result: *Arg-No.* $\rightarrow$ *Arg-No.* + *number* and *Arg-No.* $\rightarrow$ *Arg-No.* $\times$ *number*, respectively.

`FADD` *Arg-No. number*
`FMUL` *Arg-No. number*

The following example will illustrate the commands 'FMUL' and 'MLEG':

Suppose that there are two points, (3) at (.5,.15,0) and (4) at (.1,.4,0), which are to be shifted by equal amounts parallel to the a- and the b-axis, respectively, until a prescribed distance (1.39) is obtained between (3) and (4). The macro is started from an initial guess for the stepsize, e.g. 0.5.

```
ATOM (   3) x 1 0.500000  0.150000  0.000000 (define positions of atoms)
ATOM (   4) x 2 0.100000  0.400000  0.000000

MACR 2 (define macro 2)
VG 3 3 2 245501 %1 R  (Direction -a)
VG 4 4 2 254501 %1 R  (Direction -b)
L 3 4
ENDM

MACR 3 (define macro 3)
2; (call macro 2)
MLEG 1.39 0 8 1 (If distance = 1.39, go to line 8;
if distance < 1.39, go to next line;
if distance > 1.39, go to line 1)
FMUL 1 -.5  (Reverse direction and halve stepsize)
2;
MLEG 1.39 4 8 0 (If distance = 1.39,
go to line 8; if distance < 1.39, go to line 4;
if distance > 1.39, go next line)
FMUL 1 -.5
MVTO 1
ENDM

3 0.5 (activate macro 3, with initial value for variable 1 equaling 0.5)
```

Finally there is an unconditional branch command:

`MVTO line` (MoVe TO line)

As usual, care has to be taken, if macros call other macros. Usually one line of a macro should contain only one command. When calling other macros this is *mandatory*. Note that arguments are strings of characters. If they contain special characters, e.g. ";", they have to be enclosed in quotes.

There exists another way to build loops, using the command 'WAIT'. When reaching 'WAIT', the program stops and waits for input. If only the ENTER (RETURN) key is pressed, execution is continued. All other input, e.g. typing a letter, terminates the macro.

A special meaning have the symbols $C, $P or $P-$x$, where x is an integer number. $C is substituted by the number of codes in the codelist. $P is substituted by a number corresponding to the first free position in the parameter list. Consequently $P-1 is the number of the last parameter. The following example illustrates this technique by constructing a triphenylmethane molecule at the current position of the free coordinate system.

```
! Generates a triphenylmethane molecule. First parameter:
! Rotation angle of the phenyl ring around the ''bond''; second
! parameter: deviation from the plane, e.g. 109.5 tetrahedral
! angle.
MACR 1
P 0 0 1
AA H 0
P 0 0 0
AA C 0
P 1.39 0 0
GP 6 3 C 1
GREL $P-6 $P-1 2.93 0 0 1 R
DG   $P-6 $P-1 1 %1 R
DG   $P-6 $P-1 2 %2 R
DG   $P-6 $P-1 3 120 A
DG   $P-6 $P-1 3 120 A
ENDM
```

The counter, which may be set by 'SETC' and decremented by 'DMNZ', may also be incremented using the command 'INC'. To obtain the value of this counter the letter & is used. Let us assume that on position 2 in the parameter list there is a point with the coordinates (0,0,0), and there is no other point in the structure with these coordinates. Now the following macro defines every vector (2) $\rightarrow$ (%1) as direction towards the viewer.

```
MACR 3
SETC %1
K 2 2 &
WK
BF
EPU
WAIT
INC
MVTO 2
ENDM
```

Beside the 'INC' command there is the 'IMLE' (Increment and Move if Less or Equal) command. The syntax is: IMLE $n, line$. The current counter is compared with n, and if it is less or equal the execution of the macro is continued at the *line* specified.

Sometimes it is desirable to issue messages to the screen. This is possible using the command 'ECHO'. *Example:*

```
ECHO 'Target atom now:  &'
```

Macros may be shown on the screen by OMAC n, where n is the number of the macro. QMAC n writes macro n to file.

There are some restrictions when using macros due to the way KPLOT interprets the commands: The lines stored in a macro are copied to the input buffer, and a special command is appended to return to the macro interpreter after executing the input buffer. As a consequence, only columns 1-75 can be used on the screen or in a macro file. Lines within the macro must not be continued on the next line using the equal sign as is possible in dialogue mode, and they must not contain comments beginning with the exclamation mark "!" . In these cases the end of the line is not read and hence not interpreted.

The storage for the macros is 5000 characters. If there is not enough storage for a macro, this error condition is handled by reading all lines until 'ENDM' is reached. The macro itself is not available.

Macros may also be removed, i.e. all macros will be deleted if one enters `EMAC 1 9` .

Polyhedra analysis

PLDA tol_s,tol_w ;

Default values: $tol_s = 0.25$, $tol_w = 85$. If a structure consists of linked polyhedra, it can be analyzed using 'PLDA'. First, the polyhedra around all atoms in the asymmetric unit of the structure are determined (see below at 'PLDO'). Next it is checked, whether these polyhedra are of the same type. Here, it is of particular interest to find out, whether similar polyhedra which are not already equivalent by definition due to the space group symmetry, do agree within the tolerance tol_w. Polyhedra are of the same type and match, if they (a) have the same number of atoms, (b) agree with respect to additional properties set by 'PLDO', and (c) they can be shifted and rotated such that all atoms of one polyhedron can be matched to (corresponding) atoms on the second polyhedron such that the distance between atoms in a pair does not exceed tol_s. If this holds, the polyhedra are essentially equivalent.

For each comparison between a pair of polyhedra, a line is printed containing the information which polyhedra were being compared followed by a "+" or a "*" if they could be matched successfully. The star indicates that one of the two polyhedra had to be inverted to achieve a match, i.e. it had been mapped at a centre before shifting and rotating. If the two polyhedra do not agree, a "-" sign is printed together with a number indicating the reason why the match failed:

1	The names of the central atoms of the polyhedra do not agree
2	The number of ligand atoms is different
3	The geometry and/or the names of the ligands do not match

After this comparison the connectivity of the polyhedra is analyzed. For each polyhedron it is calculated, which polyhedra are linked to it, and what kind of connection is present (vertex, edge, face).

Note that in order to work with the polyhedra, it must be possible to define a 'handle'. This handle consists of the central atom and two ligand atoms which may not lie on a single line. The program selects from among the atoms surrounding a central atom two atoms (=ligands) that have an angle as close as possible to 90 degree to find a suitable handle. The parameter tol_w is the maximum allowed deviation of the angle ligand-central-ligand (in degree) from 90°.

Polyhedra options

PLDO n_1, n_2, n_3;

Defaults: $n_1 = 0$, $n_2 = 1$, $n_3 = 0$. Using this command, options for the polyhedra analysis may be set.
n_1 selects the strategy. $n_1 = 0$: First, all atoms that lie within a sphere with a radius which equals the distance to identical atoms are determined, and then those are removed which have the same name as the central atom. The remaining atoms define the polyhedron. This approach will work best if the standard deviation of the distances between the central atom and the ligands around the average central-ligand distance is small. $n_1 = 1$: A table of predefined distances is used to identify the ligand atoms belonging to the polyhedron. It is the same table that is used by 'FM', and that may be enlarged or modified using the command 'FMDP' (see p. 23). Using this option, one can even deal with intermetallics. $n_1 = 3$: Two polyhedra are compared which are stored in the mouse list. Typically, one would use e.g. the command 'M' (see p. 49) to add the atoms belonging to the two polyhedra to the mouse list, starting in each case with the central atom. Besides allowing the user to interactively compare polyhedra in a given structure and making the definition of irregular polyhedra possible, this strategy can be very useful if one wants to include e.g. lone electron pairs as vertices of the polyhedron because here the generation of the polyhedron is not restricted to regular atoms.

$n_2 \neq 0$: Comparisons will also be performed after inverting one of the two polyhedra. $n_2 = 0$: Only rotating and shifting is allowed for the comparison.

$n_3 \neq 0$: The names of the atoms belonging to the two polyhedra under consideration have to agree in addition to their geometry. Here we refer to the "working names" (see p. 77). $n_3 = 1$: Only the central atoms have to agree. $n_3 = 2$: Both central and ligand atoms must match.

KPLOT commands - assigned to topics

Commands assigned to topics

Name	Page	Abbreviated Description
		Distances / angles ..
AL	37	Change length
ATAB	18	Generate distance table
AS	18	Distance table using symbols
AW	38	Change angle
EW	38	Calculate angle between planes
FL	54	Force length
L	37	Length (distance)
LEP	71	Distance of a point from a plane
LSWS	56	Calculate edge of a triangle
MDM	18	Mean distance in molecule
MDMR	18	Mean distance in molecule + reduction
TW	38	Torsion angle
W	37	Angle
WEE	72	Angle between two planes
WTAB	18	Distance und angle table
WVE	38	Angle beween vector and plane
WVP	37	Angle by four points
		Output to screen / file
O	25	Print orientation
OA	12	Print atom parameters
OAS	17	Print atom parameters separately
OATF	14	Print atom color control blocks
OAWY	12	Print atom parameters + Wyckoff sites
OC	22	Print codes
OCG	23	Print codes in grid coordinates
OFMD	23	Print FM/FZ distance table
OFN	47	Show current file name
OGC	50	Print group of codes (short form)
OGCF	50	Print group of codes (full form)
OH	27	Print viewing distance
OMAC	96	Print macro
OMAT	26	Print orientation matrix
OOPT	39	Show options in effect
OPK	33	Print plot commands
OPTR	39	Print pointer
OS	10	Print symmetries
OSA	78	Print symmetry-axes

Commands assigned to topics

Name	Page	Abbreviated Description
OSAR	81	Print list of special directions
OSF	32	Show data control blocks for shading
OSKG	29	Show DCB's for shading balls
OSY	10	Print symmetries symbolically
OT	6	Print title
OVDD	57	Print 'VDG' data
OZ	6	Print cell constants
OZAB	74	Print cells a and b
OZB	28	Print plotting conditions
Labeling		
BA	32	Label atoms
BAM	32	Label atoms using the mouse
BSF	29	Labeling flag for atoms
BSN	29	Size of label
BST	32	Labeling type for atoms
BT	33	Label with text
BTM	33	Label with text using the mouse
BVLM	33	Label vector with length using mouse
EGCB	33	Remove codes from label. plot commands
PCDS	33	Label atoms with codes
PN	33	Label atoms with numbers
PSN	33	Label atoms with numbers
VSFT	33	Shift label
Code list		
ACAF	17	Add foreign codes to parameter list
ACAL	17	Add codes as atoms
ACI	19	Add codes immediately
ACIM	19	Like "ACI", but multiple execution
ACIZ	19	Add codes for cell outlines
ACT	20	Add codes by translation
AGCC	49	Add group of codes as codes
AGLP	19	All equal positions allowed
AKB	20	Add Cartesian box
AKS	20	Add sphere
AMC	17	Add multiple codes
AO	19	Give output on adding codes
ATB	20	Add triclinic box
AU	21	Add surrounding
AUF	21	Add surroundings to foreign atoms
AUS	21	Add surrounding acc. to symbols
AUW	21	Like "AU" with repetition
AUZ	21	Like "AU" with a number of cycles
AZIN	24	Add content of one unit cell
CC	19	Clear codes
CE	24	Set number of codes
CUTT	24	Cut triclinic box
DCN	20	Delete code(s) acc. to atom numbers
DCS	20	Delete code(s) acc. to atom symbol
DGCC	21	Delete group of codes
EFTC	67	Plane fit through codes
EFTS	67	Plane fit through symbol(s)
EN	20	Remove codes by numbers in code list
FM	22	Find molecule
FMA	22	Find molecule and add it
FZ	22	Find cell content
LGCC	22	Replace code list by mouse list

Commands assigned to topics

Name	Page	Abbreviated Description
SCI	20	Subtract codes immediately
SU	21	Subtract surrounding
UC	23	Show overview of code list
		Files
CLSE	47	Close file
EAE	46	I/O units
HEND	37	End of generation of HPGL code
HPGL	37	Generate HPGL code
IMP	46	Import file
LCFF	89	Load CRYSTIN formatted file
LCIF	89	Read CIF-File
LOG	46	Write a logfile
LZIN	90	Input file for Lazy Pulverix
MDLS	91	Generate DLS input file
MSIN	89	Generate MISSYM input
NRCA	88	Generate parameter file for NRC-System
OFN	47	Show current file name
OOPN	47	Show numbers of files currently open
OPEN	46	Open a file
OPTR	39	Print pointer
ORTF	89	Generate ORTEP input file
PROT	5	Write commands given to file
PUT	46	Create KPLOT input file
PUTC	46	Create KPLOT input file and close it
PXLE	37	Pixel graphic end
PXLG	37	Pixel graphic start
QA	47	Atoms to Q-file
QAE	48	Set nummer of Q-file
QBS	48	Backspace Q-file
QC	47	ACIM commands to Q-file
QFK	48	Free coordinates to Q-file
QK	47	Free text to Q-file
QMAC	96	Macro to Q-file
QORT	47	DK commands to Q-file
QPK	47	Plot commands to Q-file
QRW	48	Rewind Q-file
QSF	47	DCB's to Q-file
QSY	47	Symmetries to Q-file
QT	47	Title to Q-file
QUP	47	Origin of coordinate system to Q-file
QZ	47	Unit cell to Q-file
QZB	47	Plotting parameters to Q-file
RW	47	Rewind unit
SCFF	89	Write CRYSTIN formatted file
SHKL	90	Generate SCHAKAL input file
THEO	90	Create THEO file
		DLS
AR	93	Distance restraint
ARE	93	End of AR list
MDLS	91	Generate DLS input file
OR	93	Print restraint list
WR	93	Angle restraint
WRE	93	End of WR list
		Planes
CVNK	72	Convert normal vector of plane to Cartesian
CVNT	72	Convert normal vector of plane to triclinic

Commands assigned to topics

Name	Page	Abbreviated Description
EFTC	67	Plane fit through codes
EFTG	67	Plane fit trhough codes in mouse list
EFTP	67	Plane fit through parameters
EFTS	67	Plane fit through symbol(s)
EL	72	Load plane (normal vector to point reg.)
EML	68	Plane by Miller indices and d
EMP	68	Plane by Miller indices and a point
ENP	68	Plane by a vector and a point of intersection
EP	68	Plane by three points
ESP	72	Store plane
EW	38	Calculate angle between planes
FACE	68	Plane by Miller indices and d
FACF	68	Factor for command FACE
GESP	69	Generate intersecting points of planes
GPOL	69	Generate polyhedron
LEP	71	Distance of a point from a plane
MFCE	71	Generate FACE instructions
MILL	72	Miller indices of a plane
MLLE	72	Miller indices of a plane
OFCE	72	Print FACE"-line
PGCE	70	Project group of codes onto plane
PVOL	70	Calculate volume of a polyhedron
SKE	72	Store coordinate system as plane
WEE	72	Angle between two planes
WSU	69	Wigner-Seitz Surrounding
Color pointer		
CA	14	Color atom
FGC	49	Assign color pointer to group of codes
FRBN	14	Color pointer for numbers
FRBS	14	Color pointer for symbols
RFRB	17	Change radii of atoms with color pointer
SFRB	17	Assign symbol to atoms having colorpnt
SRTF	68	Sort according to color pointer
Idealisation		
AFG	60	Affine group
AFGC	60	Affine group completely
CMPO	76	Comparison of two cells - setting of options
CMPZ	76	Compare two cells
CRRI	63	Replace parameters conditionally only
DTG	66	Define and transform group
ID	62	Idealize
IDF	77	Idealize fragment
IDG	62	Idealize group
IDL	62	Idealize using list
IPSY	45	Idealize according to symmetries
ISST	81	Idealize structure according symmetry type
KIG	63	Keep ideal group
NRRI	63	Don't replace parameters wh. idealising
OUM	65	Print overlay matrix
RRI	63	Replace real parameters by ideal ones
UFFL	64	Map onto fragment and fill if empty
UFR	63	Overlay fragment
UFRA	64	Overlay fragment automatically
Coordinate system		
DK	24	Rotate coordinate system
DKOS	51	Rotate coordinate s. f. ORTEP st. plot

Commands assigned to topics

Name	Page	Abbreviated Description
DKSV	26	Rotate coordinats. f. vertical vector
DKV	24	Rotate coordinate system about a vector
GNKO	40	Generate coordinate cross
IMAT	25	Input orientation matrix
K	24	Coordinate system by points
KEE	71	Coordinate system by planes
KML	71	Coordinate system by Mill. indices
KOC	25	Coordinate system via code
KS	25	Reset coordinate system to stand. defin.
KUP	25	Org. of coord. system from point reg.
KUSP	25	Origin of coord. syst. over barycentre
MILL	72	Miller indices of a plane
MINO	39	Minimize overlap
MNOV	39	Minimize overlap (vector)
RK	25	Restore coordinate system
SK	25	Save coordinate system
SKE	72	Store coordinate system as plane
WK	25	Rotate coordinate system
XK	25	Interchange coordinate systems
ZK	38	Plot a coordinate system
Macros		
DFMA	94	Defaults for macro arguments
DMNZ	94	Decrement and move if not zero
EMAC	96	Remove macro(s)
ENDM	94	End of macro
FADD	95	Floating add (within macros)
FMUL	95	Floating multiply (within macros)
IMLE	96	Increment (within a macro)
INC	96	Increment (within a macro)
MACR	94	Define a macro
MLEG	95	Move if distance less or equal or greater
MVTO	95	Move to line
MSAV	48	Save macros
SETC	94	Set count (within a macro)
Mouse / Mouse list		
AGC	49	Add group of codes to mouse list
AGCA	49	Add group of codes as parameters
AGCC	49	Add group of codes as codes
AGCM	49	Add group of codes using the mouse
AGCT	49	Add to group list of codes a tree
BAM	32	Label atoms using the mouse
BTM	33	Label with text using the mouse
CTGC	49	Store codes in mouselist
DGC	58	Rotate group of codes (mouse list)
DGCC	21	Delete group of codes
DGCV	58	Rotate group of codes about vector
EFTG	67	Plane fit through codes in mouse list
FDGC	58	Apply factor to group of codes
FGC	49	Assign color pointer to group of codes
FLGC	58	Force lengths for group of codes
FRGC	58	Apply factor reciprocally to group of codes
LGC	48	Load group of codes (to the mouse list)
LGCC	22	Replace code list by mouse list
M	49	Load group of codes using the mouse
PGCE	70	Project group of codes onto plane
RFRB	17	Change radii of atoms with color pointer

Commands assigned to topics

Name	Page	Abbreviated Description
VDM	30	Connect directly using mouse
Options		
AGLP	19	All equal positions allowed
AO	19	Give ouput on adding codes
CFS	5	Copy foreground structure
CRRI	63	Replace parameters conditionally only
DFTB	39	Definition of color table
DLG	5	Dialog mode
FMDD	23	"FM" default distance
FMDP	23	"FM" distance pair
FMDR	23	"FM" defaults reset
FMMC	22	"FM" max. number of allowed codes
FMOP	23	"FM" option number
GLP	19	Equal points allowed
GOPT	53	Set group (A/R) option
H	27	Viewing distance
IU	17	Always U
IZA	41	Show axes always
KF	27	Factor for the ball size
KIG	63	Keep ideal group
LOG	46	Write a logfile
MORE	5	Control amount of output
MSAV	48	Save macros
NAO	19	No output while filling code list
NDLG	5	No dialogue mode
NGLP	19	Equal points not allowed
NIU	17	Not always U
NOI	63	No output while idealizing
NOKD	5	Don't echo commands
NPCK	16	Don't check parameters
NPEN	28	Define default pen (color)
NPO	35	No plot output
NPR	34	No frame
NRRI	63	Don't replace parameters wh. idealising
NS	5	Number of structure
NZFQ	48	Don't save plotting field on Q-file
OI	63	Print while idealizing
OKD	5	Print commands
PCK	16	Check parameters
PFMT	37	Plot format (landscape/portrait)
PLDO	97	Polyhedra options
PLT	35	Plotting allowed
PO	35	Plot output allowed
PR	34	Plot frame
PROT	5	Write commands given to file
PXCM	28	Definition pixels and cm (gr. term.)
QAE	48	Set number of Q-file
RRI	63	Replace real parameters by ideal ones
SH	28	Size of characters (default value)
SPST	13	Set starting point for parameters
SQLS	42	Define sequence to clear screen
SQTM	42	Define sequence to switch to text mode
STPO	35	Stereo plot option
STRD	28	Thickness of a line (default)
SW	35	Stereo angle
TAPR	30	Tapering factor

Commands assigned to topics

Name	Page	Abbreviated Description
THPR	90	Set parameters for THEO program
VEO	31	Polyhedra option
ZFQ	48	Save plotting area on Q-file
		ORTEP
CVRT	50	Convert radii to temp.-factors
DISP	51	Dispalcement parameter (ORTEP)
DKOS	51	Rotate coordinates. f. ORTEP st. plot
EPAR	50	Ellipsoid parameter
ORTP	50	Switch to ORTEP mode
WFKT	51	Probability factor for ellipsoids
		Parameterlist
AA	53	Add point as atom to parameter list
AAN	77	Change working names
ACAF	17	Add foreign codes to parameter list
ACAL	17	Add codes as atoms
AE	18	Set end of parameter list
AGCA	49	Add group of codes as parameters
AI	63	Add ideal parameter
AKRT	12	Atom parameters with Cartesian coordinates
AMC	17	Add multiple codes
AT	16	Atom
ATED	16	Edit atom
ATF	13	Atom colors control block
ATG	16	Atom (coordinates in grid points)
ATOM	12	Atom (name and parameters)
CA	14	Color atom
CNCL	45	Condense cluster
CPG	56	Copy group
CVNK	72	Convert normal vector of plane to Cartesian
CVNT	72	Convert normal vector of plane to triclinic
DELA	53	Delete atom from parameter list
DG	56	Rotate group
DGC	58	Rotate group of codes (mouse list)
DGCV	58	Rotate group of codes about vector
DGV	56	Rotate group about a vector
EFTP	67	Plane fit through parameters
EGCP	39	Remove param. according to group list of codes
EPL	53	Remove parameter
FDG	59	Apply factor to group
FRG	59	Apply factor reciprocally to group
GDTU	61	Generate 3 points in tetrah. surr.
GESP	69	Generate intersecting points of planes
GETU	61	Generate 1 point in Td environment
GP	60	Generate points
GPHU	61	Generate phenyl surrounding
GREL	62	Group relative
GRZ	41	Generate reciprocal cell
GT	16	Define a grid
GZTU	61	Generate 2 pts. in Td environment
HNH	68	Bring H atoms to the end of the list
KTG	59	Displace group using coordinate system
MVP	17	Move parameter (re-order)
N	17	Name of an atom
NPZ	16	Normalize parameters acc. cell
PGCE	70	Project group of codes onto plane
QDG	59	Apply quotient directly to group

Commands assigned to topics

Name	Page	Abbreviated Description
R	16	Radius resp. radii of atoms
RAN	77	Reset working names
RFRB	17	Change radii of atoms with color pointer
RPSY	45	Reduce parameters acc. to symmetries
RS	16	Radii of atoms by symbol
SFRB	17	Assign symbol to atoms having colorpnt
SPG	59	Mirror group
SRGS	12	Symmetries to file f or RGS
U	13	Show overview of atoms
VDG	56	Shift and rotate group
VDGC	58	Shift and rotate group of codes
VDGD	57	Data for 'VDG'
VG	56	Shift group (in Angstrom)
VGC	57	Shift group of codes (in A)
VGCT	57	Shift group of codes triclinic
VGDP	59	Shift group like three points
VGV	56	Shift group (in units of a vector)
VM	58	Shift group of codes using mouse
VZG	59	Distort group
XAT	17	Interchange atoms
XP	17	Interchange parameters
ZG	61	Centric group
Plot commands		
AFZ	34	Change color pointer
APBF	34	Change plot command (type)
APK	34	Modify plot command (parameters)
CVDP	70	Change VD-parameter
DFTB	39	Definition of color table
EPK	34	Remove plot command(s)
EXP	35	Execute plot commands
EXPU	35	Execute plot commands r. hidden lines
FLM	29	Pattern for filling
FLMB	29	Pattern for filling given by bytes
MVPK	34	Move plot commands (re-order list)
NP	29	Prohibit plotting of sel. atoms/bonds
NPM	29	Don't plot (selectively, mouse)
NPN	29	Don't plot (selectively, numbers)
PF	31	Plot plane (polygon)
PFD	31	Plot plane directly
PFU	31	Plot plane opaquely
PK	28	Plot spheres
PKS	28	Plot spheres by symbol
VB	30	Connect
VBR	30	Connect with sticks
VBS	30	Connect with lines
VD	30	Connect directly
VDM	30	Connect directly using mouse
VDR	30	Connect with sticks directly
VDS	30	Connect with lines directly
VE	30	Plot polyhedron
VEH	30	Plot polyhedron half opaque
VS	30	Connect symbols
VSR	30	Connect symbols with sticks
VSS	30	Connect symbols with lines
ZK	38	Plot a coordinate system
Point register		

Commands assigned to topics

Name	Page	Abbreviated Description
AA	53	Add point as atom to parameter list
DP	54	Rotate point
DPV	54	Rotate a point around a vector
EL	72	Load plane (normal vector to point reg.)
FL	54	Force length
OP	54	Print point register
P	53	Point in free coordinate system
PAW	55	Point by distance and two angles
PDA	55	Point by three distances
PEEE	69	Point by intersecting of three planes
PEZA	55	Point in plane by two distances
PGM	56	Point on line by mouse
PL	53	Load point register with code
PM	56	Point by mouse
PP	38	Plot point (point register)
PREL	53	Point relative
PT	53	Point in triclinic coordinates
PTTR	54	Transform point to reciprocal system
PVA	55	Point by vector and distance
PVE	71	Point by vector and plane
PZA	55	Point by two distances
PZAA	55	Point by two distances (other side)
SP	54	Center of gravity
SPC	54	Center of gravity via codes
SPGC	54	Center of gravity via group of codes
SVE	56	Point of intersection vector - plane
SVV	55	Point of intersection vector - vector
VP	54	Shift point
VPV	54	Shift point (vector units)
WA	40	Which atom?
Polyhedra		
PLDA	97	Polyhedra analysis
PLDO	97	Polyhedra options
Reflections		
EAR	18	Remove systematically absent reflect.
TR	18	Test reflex
SHELX		
GNSX	87	Generate SHELX atom parameters
GSXF	87	Flag for GSXJ
GSXJ	87	Generate SHELXL job
OTM	43	Print transformations up to now
SXAT	87	SHELX atom lines
SXL	86	Load SHELX instruction file (.ins)
Scaling		
AF	27	Change factor
BF	27	Best fit (scaling factor)
F	27	Define scaling factor
UR	27	Define origin on plotting area
Miscellaneous		
AIDZ	39	How many atoms do we have in the cell?
DKML	71	Definition of a KML field
DRMO	40	Ball- and stick model (plastic and wire)
ECHO	96	Echo text on screen
FK	26	Calculate free coordinates
GASP	87	Generate atoms including special site parameters
GOMX	41	Generate orientation matrix

Commands assigned to topics

Name	Page	Abbreviated Description
GT	16	Define a grid
INIT	93	Initialize
IOMX	41	Import orientation matrix
MVOL	41	Calculate volume of a molecule
NKML	71	Next KML
SYS	40	System call
SYSW	40	SYS wait
T	6	Title
VNRM	37	Norm vector
WTB	24	Which triclinic box?
Sorting		
HNH	68	Bring H atoms to the end of the list
SRT	67	Sort according to free coordinates
SRTC	67	Sort codes
SRTF	68	Sort according to color pointer
SRTL	67	Sort parameters acc. to list
SRTN	67	Sort parameters acc. to names
SRTP	67	Sort according to points
SRTT	67	Sort acc. to triclinic coordinates
Symmetries		
C	7	Centering
CNCL	45	Condense cluster
CNRM	84	Copy normalizers
EDS	11	Remove double symmetries
ES	11	Remove symmetries
EHMS	10	Extra Hermann-Mauguin symbol
GRTS	11	Group test
GTY	7	Lattice type
HMS	8	Load space group acc. H.M. symbol
HMSO	10	HMS-option (setting)
IDA	82	Idealize axis
IGFA	78	Ignore foreign atoms
IPSY	45	Idealize according to symmetries
ISST	81	Idealize structure according type of symmetry
ITS	44	Import translational symmetries
MSIN	89	Generate MISSYM input
MTRI	45	Reduce symmetry to P1 (make triclinic)
ONRM	84	List normalizers
OS	10	Print symmetries
OSA	78	Print symmetry-axes
OSAR	81	Print list of special directions
RG	10	Load space group with number
RGS	82	Search a space group
RPSY	45	Reduce parameters acc. to symmetries
S	6	Symmetry as matrix
SAR	81	Search only in the special directions
SDEL	81	Delete symmetry direction(s)
SE	11	Set number of symmetries
SEL	12	Select symmetries
SFND	78	Find symmetries
SGOG	44	Search for common supergroups (aristotype)
SLAB	78	Find symmetries in a slab
SPUG	43	Search path to subgroup
SM	82	Search mirror planes in molecule
SPG	59	Mirror group
SPUG	43	Search path to subgroup

Commands assigned to topics

Name	Page	Abbreviated Description
SSI	80	Search symmetry and idealize
ST	11	Symmetry by type
STA	11	Option: Symmetry by type (ST) also add
SY	7	Enter symmetry acc. to symbol
SYML	87	Generate file for the program SYMMOL
TGL	78	Test for equal sites
TIM	43	Transform isomorphically
TNRM	84	Transform according to a normalizer symmetry
TRSY	45	Transform Symmetries
TSY	83	Test symmetry
TSYI	83	Test symmetry after idealisation
TSZ	82	Search translation symmetry in cell
TUG	43	Transform to subgroup
WRGN	44	Which space group number?
ZFND	83	Find cell
ZIDL	79	Idealize cell
ZZ	82	Search for centers of symmetries
		Temperature factors
GTF	18	Generate temperature factor
LTF	15	Clear temperature factors
OTF	16	Print temperature factors
TF	14	Temperature factor
TFL	15	Mode of writing temperature factors
TL	15	Temperatur factor of last atom
		Terminating
CLSE	47	Close file
END	94	Terminate program
ESC	40	Set escape sequence
HEND	37	End of generation of HPGL code
PXLE	37	Pixel graphic end
QUIT	94	Terminate program
RSTR	93	Restart
		Valence sums
OOX	86	Print list with oxidation stages
OX	86	Definition of an oxidation stage
OXL	86	Delete list with oxidation stages
ORIJ	86	Print list with Rij-values
RIJ	86	Define Rij-value
RIJB	86	Calculate Rij-value
RIJL	86	Delete Rij-list
RIJZ	86	Print Rij-value
VSUM	85	Compute valence sums
		Plotting
AINI	36	Initialize animation
BUFN	36	Set buffer numbers view/write
D	26	Rotate and redraw
DF	26	Duplicate drawing area
EPU	36	Single plot respecting hidden lines
FO	27	Origin on drawing field
G	36	Graphic (=BF;EPU or BF;STPU)
GG	36	Graphic (=BF;STPU)
GNZL	22	Generate cell outlines
H	27	Viewing distance
HEND	37	End of generation of HPGL code
HF	26	Bisect drawing area
HPGL	37	Generate HPGL code

Commands assigned to topics

Name	Page	Abbreviated Description
KF	27	Factor for the ball size
LAF	38	Clear outside of window
LNP	30	Clear non-plot list
LPK	34	Clear plot commands
LS	35	Clear screen (graphic)
MBUF	36	Switch to multi buffering
MPK	34	Mask plot command(s)
PCDS	33	Label atoms with codes
PN	33	Label atoms with numbers
PP	38	Plot point (point register)
PR	34	Plot frame
PSCD	33	Label atoms with codes
PSN	33	Label atoms with numbers
PST	34	Start plotter (IBM)
PXCM	28	Definition pixels and cm (gr. term.)
PXLE	37	Pixel graphic end
PXLG	37	Pixel graphic start
SF	32	Hatch planes
SKG	29	Hatch spheres
SNS	39	Set Null black (PC's only)
SNW	39	Set Null white (PC's only)
SQLS	42	Define sequence to clear screen
STP	35	Generate stereo plot
STPO	35	Stereo plot option
STPU	35	Stereo plot respecting hidden lines
STRD	28	Thickness of a line (default)
UDST	40	Parameter for thin band
UMPK	34	Unmask plot command(s)
VEU	30	Plot polyhedron opaque
VF	27	Shift plot field
VMOD	40	Set video mode (PC)
VSFT	33	Shift label
WPBF	37	Write plot buffer
XC	39	Exchange color numbers
XZF	26	Interchange parameters for drawing field
ZF	26	Define plotting area
ZK	38	Plot a coordinate system
Cell		
DZA	73	Define cell a
DZB	73	Define cell b
DZV	65	Rotate cell around a vector
GRZ	41	Generate reciprocal cell
IZA	41	Show axes always
LZ	74	Load cell
OTM	43	Print transformations up to now
OZ	6	Print cell constants
OZAB	74	Print cells a and b
RDZ	44	Reduce cell
RTHO	44	Rhombohedral to hexagonal obverse
RZ	44	Recall cell
SZ	74	Search cell
SZA	75	Search cell automatically
SZAT	75	Tolerances for SZA
SZXZ	75	Interchange cell with search-cell
TIM	43	Transform isomorphically

Commands assigned to topics

Name	Page	Abbreviated Description
TZ	42	Transform cell
TZC	42	Transform cell and insert centerings
TZP	42	Transform cell using parameters
TZUP	43	Transform cell origin us. parameters
TZUR	42	Transform origin of cell
VDZ	65	Move and rotate cell (macro)
VZ	65	Move cell by a vector
VZAB	73	Compare cell a with cell b
VZDP	77	Move cell according to three points
VZV	65	Move cell by a vector
XZ	6	Interchange direct with reciprocal cell
Z	6	Cell constants
ZA	41	Show axes
ZIDL	79	Idealize cell
ZBAS	6	Lattice constants via basis vectors
ZTAK	43	Cell and transformation of atom coord.
Two structures		
AAN	77	Change working names
ACAF	17	Add foreign codes to parameter list
AO	19	Give ouput on adding codes
AUF	21	Add surroundings to foreign atoms
CCL	64	Compare clusters
CMPO	76	Comparison of two cells - setting of options
CMPZ	76	Compare two cells
DZV	65	Rotate cell around a vector
IDF	77	Idealize fragment
NAO	19	No output while filling code list
NS	5	Number of structure
RAN	77	Reset working names
UFFL	64	Map onto fragment and fill if empty
UFR	63	Overlay fragment
UFRA	64	Overlay fragment automatically
VDZ	65	Move and rotate cell (macro)
VZDP	77	Move cell according to three points
VZ	65	Move cell by a vector
VZV	65	Move cell by a vector
ZBAS	6	Lattice constants via basis vectors

Commands in alphabetical order

Name	Page	Abbreviated Description
AA	53	Add point as atom to parameter list
AAN	77	Change working names
ACAF	17	Add foreign codes to parameter list
ACAL	17	Add codes as atoms
ACI	19	Add codes immediately
ACIM	19	Like “ACI”, but multiple execution
ACIZ	19	Add codes for cell outlines
ACT	20	Add codes by translation
AE	18	Set end of parameter list
AF	27	Change factor
AFG	60	Affine group
AFGC	60	Affine group completely
AFG	60	Affine group
AFZ	34	Change color pointer
AGCA	49	Add group of codes as parameters
AGCC	49	Add group of codes as codes
AGCM	49	Add group of codes using the mouse
AGCT	49	Add to group list of codes a tree
AGLP	19	All equal positions allowed
AI	63	Add ideal parameter
AIDZ	39	How many atoms do we have in the cell?
AINI	36	Initialize animation
AKB	20	Add Cartesinan box
AKRT	12	Atom parameters with Cartesian coordinates
AKS	20	Add sphere
AL	37	Change length
AMC	17	Add multiple codes
AO	19	Give ouput on adding codes
APBF	34	Change plot command (type)
APK	34	Modify plot command (parameters)
AR	93	Distance restraint
ARE	93	End of AR list
AS	18	Distance table using symbols
AT	16	Atom
ATAB	18	Generate distance table
ATB	20	Add triclinic box
ATED	16	Edit atom
ATF	13	Atom colors control block
ATG	16	Atom (coordinates in grid points)
ATOM	12	Atom (name and parameters)
AU	21	Add surrounding
AUF	21	Add surroundings to foreign atoms
AUS	21	Add surrounding acc. to symbols
AUW	21	Like “AU” with repetition
AUZ	21	Like “AU” with a number of cycles
AW	38	Change angle
AZIN	24	Add content of one unit cell
BA	32	Label atoms
BAM	32	Label atoms using the mouse
BF	27	Best fit (scaling factor)
BSF	29	Labeling flag for atoms
BSN	29	Size of label
BST	32	Labeling type for atoms
BT	33	Label with text
BTM	33	Label with text using the mouse

Commands in alphabetical order

Name	Page	Abbreviated Description
BUFN	36	Set buffer numbers view/write
BVLM	33	Label vector with length using mouse
C	7	Centering
CA	14	Color atom
CC	19	Clear codes
CCL	64	Compare clusters
CE	24	Set number of codes
CFS	5	Copy foreground structure
CLSE	47	Close file
CMPO	76	Comparison of two cells - setting of options
CMPZ	76	Compare two cells
CNCL	45	Condense cluster
CNRM	84	Copy normalizers
CPG	56	Copy group
CRRI	63	Replace parameters conditionally only
CTGC	49	Store codes in mouselist
CUTT	24	Cut triclinic box
CVDP	70	Change VD-parameter
CVRT	50	Convert radii to temp.-factors
D	26	Rotate and redraw
DCN	20	Delete code(s) acc. to atom numbers
DCS	20	Delete code(s) acc. to atom symbol
DELA	53	Delete atom from parameter list
DF	26	Duplicate drawing area
DFMA	94	Defaults for macro arguments
DFTB	39	Definition of color table
DG	56	Rotate group
DGC	58	Rotate group of codes (mouse list)
DGCC	21	Delete group of codes
DGCV	58	Rotate group of codes about vector
DGV	56	Rotate group about a vector
DISP	51	Displcement parameter (ORTEP)
DK	24	Rotate coordinate system
DKML	71	Definition of a KML field
DKOS	51	Rotate coordinates. f. ORTEP st. plot
DKSV	26	Rotate coordinates. f. vertical vector
DKV	24	Rotate coordinate system about a vector
DLG	5	Dialog mode
DMNZ	94	Decrement and move if not zero
DP	54	Rotate point
DPV	54	Rotate a point around a vector
DRMO	40	Ball- and stick model (plastic and wire)
DTG	66	Define and transform group
DZA	73	Define cell a
DZB	73	Define cell b
DZV	65	Rotate cell around a vector
EAE	46	I/O units
EAR	18	Remove systematically absent reflect.
ECHO	96	Echo text on screen
EDS	11	Remove double symmetries
EFTC	67	Plane fit through codes
EFTG	67	Plane fit trhough codes in mouse list
EFTP	67	Plane fit through parameters
EFTS	67	Plane fit through symbol(s)
EGCB	33	Remove codes from label. plot commands

Commands in alphabetical order

Name	Page	Abbreviated Description
EGCP	39	Remove param. according to group list of codes
EHMS	10	Extra Hermann Mauguin symbol
EL	72	Load plane (normal vector to point reg.)
EMAC	96	Remove macro(s)
EML	68	Plane by Miller indices and d
EMP	68	Plane by Miller indices and a point
EN	20	Remove codes by numbers in code list
END	94	Terminate program
ENDM	94	End of macro
ENP	68	Plane by a vector and a point of intersection
EP	68	Plane by three points
EPAR	50	Ellipsoid parameter
EPK	34	Remove plot command(s)
EPL	53	Remove parameter
EPU	36	Single plot respecting hidden lines
ES	11	Remove symmetries
ESC	40	Set escape sequence
ESP	72	Store plane
EW	38	Calculate angle between planes
EXP	35	Execute plot commands
EXPU	35	Execute plot commands r. hidden lines
F	27	Define scaling factor
FACE	68	Plane by Miller indices and d
FACF	68	Factor for command FACE
FADD	95	Floating add (within macros)
FDG	59	Apply factor to group
FDGC	58	Apply factor to group of codes
FGC	49	Assign color pointer to group of codes
FK	26	Calculate free coordinates
FL	54	Force length
FLGC	58	Force lengths for group of codes
FLM	29	Pattern for filling
FLMB	29	Pattern for filling given by bytes
FM	22	Find molecule
FMA	22	Find molecule and add it
FMDD	23	“FM” default distance
FMDP	23	“FM” distance pair
FMDR	23	“FM” defaults reset
FMMC	22	“FM” max. number of allowed codes
FMOP	23	“FM” option number
FMUL	95	Floating multiply (within macros)
FO	27	Origin on drawing field
FRBN	14	Color pointer for numbers
FRBS	14	Color pointer for symbols
FRG	59	Apply factor reciprocally on group
FRGC	58	Apply factor reciprocally to group of codes
FZ	22	Find cell content
G	36	Graphic (=BF;EPU or BF;STPU)
GASP	87	Generate atoms including special site parameters
GDTU	61	Generate 3 points in tetrah. surr.
GESP	69	Generate intersecting points of planes
GET	46	Read file
GETU	61	Generate 1 point in Td environment
GG	36	Graphic (=BF;STPU)
GLP	19	Equal points allowed

Commands in alphabetical order

Name	Page	Abbreviated Description
GNKO	40	Generate coordinate cross
GNSX	87	Generate SHELX atom parameters
GNZL	22	Generate cell outlines
GOMX	41	Generate orientation matrix
GOPT	53	Set group (A/R) option
GP	60	Generate points
GPHU	61	Generate phenyl surrounding
GPOL	69	Generate polyhedron
GREL	62	Group relative
GRTS	11	Group test
GRZ	41	Generate reciprocal cell
GSXF	87	Flag for GSXJ
GSXJ	87	Generate SHELXL job
GT	16	Define a grid
GTF	18	Generate temperature factor
GTY	7	Lattice type
GZTU	61	Generate 2 pts. in Td environment
H	27	Viewing distance
HEND	37	End of generation of HPGL code
HF	26	Bisect drawing area
HMS	8	Load space group acc. H.M. symbol
HMSO	10	HMS-option (setting)
HNH	68	Bring H atoms to the end of the list
HPGL	37	Generate HPGL code
INIT	93	Initialize
ID	62	Idealize
IDA	82	Idealize axis
IDF	77	Idealize fragment
IDG	62	Idealize group
IDL	62	Idealize using list
IGFA	78	Ignore foreign atoms
IMAT	25	Input orientation matrix
IMLE	96	Increment (within a macro)
IMP	46	Import file
INC	96	Increment (within a macro)
IOMX	41	Import orientation matrix
IPSY	45	Idealize according to symmetries
ISST	81	Idealize structure according symmetry type
ITS	44	Import translational symmetries
IU	17	Always U
IZA	41	Show axes always
K	24	Coordinate system by points
KEE	71	Coordinate system by planes
KF	27	Factor for the ball size
KIG	63	Keep ideal group
KML	71	Coordinate system by Mill. indices
KOC	25	Coordinate system over code
KS	25	Reset coordinate system to stand. defin.
KTG	59	Displace group using coordinate system
KUP	25	Org. of coord. system from point reg.
KUSP	25	Origin of coord. syst. over barycentre
L	37	Length (distance)
LAF	38	Clear outside of window
LCFF	89	Load CRYSTIN formatted file
LCIF	89	Read CIF-File

Commands in alphabetical order

Commands in alphabetical order

Name	Page	Abbreviated Description
OATF	14	Print atom color control blocks
OAWY	12	Print atom parameters + Wyckoff sites
OC	22	Print codes
OCG	23	Print codes in grid coordinates
OFCE	72	Print FACE"-line
OFMD	23	Print FM/FZ distance table
OFN	47	Show current file name
OGC	50	Print group of codes (short form)
OGCF	50	Print group of codes (full form)
OH	27	Print viewing distance
OI	63	Print while idealizing
OKD	5	Print commands
OMAC	96	Print macro
OMAT	26	Print orientation matrix
ONRM	84	List normalizers
OOPN	47	Show numbers of files currently open
OOPT	39	Show options in effect
OOX	86	Print list with oxidation stages
OP	54	Print point register
OPEN	46	Open a file
OPK	33	Print plot commands
OPTR	39	Print pointer
OR	93	Print restraint list
ORIJ	86	Print list with Rij-values
ORTF	89	Generate ORTEP input file
ORTP	50	Switch to ORTEP mode
OS	10	Print symmetries
OSA	78	Print symmetry-axes
OSAR	81	Print list of special directions
OSF	32	Show data control blocks for shading
OSKG	29	Show DCB's for shading balls
OSY	10	Print symmetries symbolically
OT	6	Print title
OTF	16	Print temperature factors
OTM	43	Print transformations up to now
OUM	65	Print overlay matrix
OVDD	57	Print 'VDG' data
OX	86	Definition of an oxidation stage
OXL	86	Delete list with oxidation stages
OZ	6	Print cell constants
OZAB	74	Print cells a and b
OZB	28	Print plotting conditions
P	53	Point in free coordinate system
PAW	55	Point by distance and two angles
PCDS	33	Label atoms with codes
PCK	16	Check parameters
PDA	55	Point by three distances
PEEE	69	Point by intersecting of three planes
PEZA	55	Point in plane by two distances
PF	31	Plot plane (polygon)
PFD	31	Plot plane directly
PFMT	37	Plot format (landscape/portrait)
PFU	31	Plot plane opaquely
PGCE	70	Project group of codes onto plane
PGM	56	Point on line by mouse
PK	28	Plot spheres

Commands in alphabetical order

Commands in alphabetical order

Name	Page	Abbreviated Description
RIJL	86	Delete Rij-list
RIJZ	86	Print Rij-value
RK	25	Restore coordinate system
RPSY	45	Reduce parameters acc. to symmetries
RRI	63	Replace real parameters by ideal ones
RS	16	Radii of atoms by symbol
RSTR	93	Restart
RTHO	44	Rhombohedral to hexagonal obverse
RW	47	Rewind unit
RZ	44	Recall cell
S	6	Symmetry as matrix
SAR	81	Search only in the special directions
SCFF	89	Write CRYSTIN formatted file
SCI	20	Subtract codes immediately
SDEL	81	Delete symmetry direction(s)
SE	11	Set number of symmetries
SEL	12	Select symmetries
SETC	94	Set count (within a macro)
SF	32	Hatch planes
SFND	78	Find symmetries
SFRB	17	Assign symbol to atoms having colorpnt
SGOG	44	Search for common supergroups (aristotype)
SH	28	Size of characters (default value)
SHKL	90	Generate SCHAKAL input file
SK	25	Save coordinate system
SKE	72	Store coordinate system as plane
SKG	29	Hatch spheres
SLAB	78	Find symmetries in a slab
SM	82	Search mirror planes in molecule
SNS	39	Set Null black (PC's only)
SNW	39	Set Null white (PC's only)
SP	54	Center of gravity
SPC	54	Center of gravity via codes
SPG	59	Mirror group
SPGC	54	Center of gravity via group of codes
SPST	13	Set starting point for parameters
SPUG	43	Search path to subgroup
SQLS	42	Define sequence to clear screen
SQTM	42	Define sequence to switch to text mode
SRGS	12	Symmetries to file for RGS
SRT	67	Sort according to free coordinates
SRTC	67	Sort codes
SRTF	68	Sort according to color pointer
SRTL	67	Sort parameters acc. to list
SRTN	67	Sort parameters acc. to names
SRTP	67	Sort according to points
SRTT	67	Sort acc. to triclinic coordinates
SSI	80	Search symmetry and idealize
ST	11	Symmetry by type
STA	11	Option: Symmetry by type (ST) also add
STP	35	Generate stereo plot
STPO	35	Stereo plot option
STPU	35	Stereo plot respecting hidden lines
STRD	28	Thickness of a line (default)
SU	21	Subtract surrounding
SVE	56	Point of intersection vector - plane

Commands in alphabetical order

Name	Page	Abbreviated Description
SVV	55	Point of intersection vector - vector
SW	35	Stereo angle
SXAT	87	SHELX atom lines
SXL	86	Load SHELX instruction file (.ins)
SY	7	Enter symmetry acc. to symbol
SYML	87	Generate file for the program SYMMOL
SYS	40	System call
SYSW	40	SYS wait
SZ	74	Search cell
SZA	75	Search cell automatically
SZAT	75	Tolerances for SZ resp. SZA
SZXZ	75	Interchange cell with search-cell
T	6	Title
TAPR	30	Tapering factor
TF	14	Temperature factor
TFL	15	Mode of writing temperature factors
TGL	78	Test for equal sites
THEO	90	Create THEO file
THPR	90	Set parameters for THEO program
TIM	43	Transform isomorphically
TL	15	Temperatur factor of last atom
TR	18	Test reflex
TNRM	84	Transform according to a normalizer symmetry
TRSY	45	Transform symmetries
TSY	83	Test symmetry
TSYI	83	Test symmetry after idealisation
TSZ	82	Search translation symmetry in cell
TUG	43	Transform to subgroup
TW	38	Torsion angle
TZ	42	Transform cell
TZC	42	Transform cell and insert centerings
TZP	42	Transform cell using parameters
TZUP	43	Transform cell origin us. parameters
TZUR	42	Transform origin of cell
U	13	Show overview of atoms
UC	23	Show overview of code list
UDST	40	Parameter for thin band
UFFL	64	Map onto fragment and fill if empty
UFR	63	Overlay fragment
UFRA	64	Overlay fragment automatically
UMPK	34	Unmask plot command(s)
UR	27	Define origin on plotting area
VB	30	Connect
VBR	30	Connect with sticks
VBS	30	Connect with lines
VD	30	Connect directly
VDG	56	Shift and rotate group
VDGC	58	Shift and rotate group of codes
VDGD	57	Data for 'VDG'
VDM	30	Connect directly using mouse
VDR	30	Connect with sticks directly
VDS	30	Connect with lines directly
VDZ	65	Move and rotate cell (macro)
VE	30	Plot polyhedron
VEH	30	Plot polyhedron half opaque

Commands in alphabetical order

Name	Page	Abbreviated Description
VEO	31	Polyhedra option
VEU	30	Plot polyhedron opaque
VF	27	Shift plot field
VG	56	Shift group (in Angstrom)
VGC	57	Shift group of codes (in A)
VGCT	57	Shift group of codes triclinic
VGDP	59	Shift group like three points
VGV	56	Shift group (in units of a vector)
VM	58	Shift group of codes using mouse
VMOD	40	Set video mode (PC)
VNRM	37	Norm vector
VP	54	Shift point
VPV	54	Shift point (vector units)
VS	30	Connect symbols
VSFT	33	Shift label
VSR	30	Connect symbols with sticks
VSS	30	Connect symbols with lines
VSUM	85	Compute valence sums
VZ	65	Move cell by a vector
VZAB	73	Compare cell a with cell b
VZDP	77	Move cell according to three points
VZG	59	Distort group
VZV	65	Move cell by a vector
W	37	Angle
WA	40	Which atom?
WEE	72	Angle between two planes
WFKT	51	Probability factor for ellipsoids
WK	25	Rotate coordinate system
WPBF	37	Write plot buffer
WR	93	Angle restraint
WRE	93	End of WR list
WRGN	44	Which space group number?
WSU	69	Wigner-Seitz Surrounding
WTAB	18	Distance und angle table
WTB	24	Which triclinic box?
WVE	38	Angle beween vector and plane
WVP	37	Angle by four points
XAT	17	Interchange atoms
XC	39	Exchange color numbers
XK	25	Interchange coordinate systems
XP	17	Interchange parameters
XZ	6	Interchange direct with reciprocal cell
XZF	26	Interchange parameters for drawing field
Z	6	Cell constants
ZA	41	Show axes
ZBAS	6	Lattice constants via basis vectors
ZF	26	Define plotting area
ZFQ	48	Save plotting area on Q-file
ZG	61	Centric group
ZFND	83	Find cell
ZIDL	79	Idealize cell
ZK	38	Plot a coordinate system
ZTAK	43	Cell and transformation of atom coord.
ZZ	82	Search for centers of symmetries

Appendix:

Documentation and availability

Appendix: Documentation and availability

The program KPLOT (written by Rudolf Hundt, Institute for Inorganic Chemistry, University of Bonn, Germany) is designed to draw, construct and analyze crystal structures. Starting from the first version in 1979 the software has been continuously improved and updated. It provides:

- Various strategies for step-wise construction of crystal structure models
- Facility to work with two crystal structures at the same time
- Tools for the comparison of two crystal structures
- Plotting of ball-and-stick models, thermal ellipsoids and coordination polyhedra
- Symmetry analysis and transformation, space group determination
- Calculation of interatomic distances, angles and torsion angles
- Interfaces to SHELX, ORTEP, SCHAKAL, MISSYM, LAZY PULVERIX
- Easy usability in script environments for automated analysis and visualization

KPLOT is written in Fortran and has been implemented under the Linux and Windows operating systems. It is available free-of-charge and can be downloaded from one of the following links:
http://www.crystalimpact.com/download/kplot.htm
http://el-tim.edu.rs/

For further information about the code, please contact the author directly:

Dr. Rudolf Hundt
Institute for Inorganic Chemistry
Gerhard-Domagk-Str. 1
D-53121 Bonn, Germany
email: hundt@uni-bonn.de

Considering the long life of the KPLOT program and the many new capabilities that have been added during the last four decades, a large documentation regarding the KPLOT program has evolved. Beside the complete KPLOT manual presented here, a KPLOT tutorial has been published. Both books are available free-of-charge and can be downloaded in .pdf format from one of the following links:
http://opentechnicum.com/
http://opentechnicum.com/?page_id=1177

Over the past 40 years, many research papers, book chapters, conference proceedings and other relevant scientific work has been published where the KPLOT program has been employed as part of the research effort. One should point out that during the first 20 years of its existence there were no codes with similar capabilities available, and thus often no explicit reference was made in the publication when KPLOT had been used. Furthermore, the KPLOT program has had a strong influence on other codes and algorithms developed to draw and analyze crystal structures (e.g. the Diamond program).
Thus, we would like to present a few typical applications of KPLOT in the literature, to help the reader to better understand the algorithms used in KPLOT and the type of research problems where KPLOT has been useful in the analysis of structures, as well as some recent applications of the code.

Below, we provide the abstracts of some selected publications, which have been grouped according to the type of application: Description of analysis algorithms contained in the KPLOT program (1), applications in the context of structure prediction (2), applications in the context of energy landscape investigations (3), applications in the context of drawing pictures and visualization (4), applications in crystallographic analysis and structure determination (5), applications in the field of data mining (6), and influence on other analysis programs (7).

1) Description of analysis algorithms contained in the KPLOT program

J. Appl. Cryst. (1998). 31, 922-928
doi: 10.1107/S0021889898008735

A New Algorithm for Space-Group Determination

A. Hannemann, R. Hundt, J. C. Schön and M. Jansen

Institut für Anorganische Chemie Universität Bonn, Gerhard Domagk-Strasse 1, 53121 Bonn, Germany

An important part of the crystallographic description of crystal structures, whether they belong to synthesized compounds or have been generated by computer, is the assignment of the correct space group. Since this task often proves to be highly nontrivial, we have developed an algorithm which determines the space group and the transformation to the standard setting of a given crystal structure, where no restrictions are placed on the original description of the structure.

J. Appl. Cryst. (1999). 32, 413-416
doi: 10.1107/S0021889898015763

Determination of symmetries and idealized cell parameters for simulated structures

R. Hundt,[a] J. Christian Schön,[a] A. Hannemann[b] and M. Jansen[b]

[a]Institut für Anorganische Chemie, Universität Bonn, Gerhard-Domagk-Strasse 1, D-53121 Bonn, Germany
[b]MPI für Festkörperforschung, Postfach 800665, 70506 Stuttgart, Germany

A robust algorithm is presented that determines the symmetries present in an atomic configuration and idealizes the cell parameters according to the crystal system suggested by the symmetries detected. No information besides the coordinates of the atoms within some arbitrary unit cell of the crystal is required.

J. Appl. Cryst. (2006). 39, 6-16
doi: 10.1107/S0021889805032450

CMPZ - an algorithm for the efficient comparison of periodic structures

R. Hundt[a], J. C. Schön[b] and M. Jansen[b]

[a]Institut für Anorganische Chemie Universität Bonn, Gerhard Domagk-Strasse 1, 53121 Bonn, Germany
[b]MPI für Festkörperforschung, Heisenbergstr. 1, D-70569 Stuttgart, Germany

The systematic comparison of the atomic structure of solid compounds has become an important task in crystallography, chemistry, physics and materials science, in particular in the context of structure prediction and structure determination of crystalline solids. In this work, an efficient and robust algorithm for the comparison of periodic structures is presented, which is based on the mapping of the point patterns of the two structures into each other. This algorithm has been implemented as the module *CMPZ* in the structure visualization and analysis program *KPLOT*.

Keywords: computer algorithms; periodic structures; similarity.

J. Appl. Cryst. (2013). 46, 587-593
doi: 10.1107/S0021889813006894

CCL: an algorithm for the efficient comparison of clusters

R. Hundt[a], J. C. Schön[b], S. Neelamraju[b], J. Zagorac[b] and M. Jansen[b]

[a]Institut für Anorganische Chemie, Universität Bonn, Gerhard-Domagk-Strasse 1, D-53121 Bonn, Germany
[b]Max-Planck-Institut für Festkörperforschung, Heisenbergstrasse 1, D-70569 Stuttgart, Germany

The systematic comparison of the atomic structure of solids and clusters has become an important task in crystallography, chemistry, physics and materials science, in particular in the context of structure prediction and structure determination of nanomaterials. In this work, an efficient and robust algorithm for the comparison of cluster structures is presented, which is based on the mapping of the point patterns of the two clusters onto each other. This algorithm has been implemented as the module CCL in the structure visualization and analysis program *KPLOT*.

Keywords: cluster structures; structure prediction; structure determination; nanomaterials; computer programs.

2) Applications in the context of structure prediction

Crystal Growth & Design, 2007, *7* (9), pp 1738–1745
doi: 10.1021/cg060872y

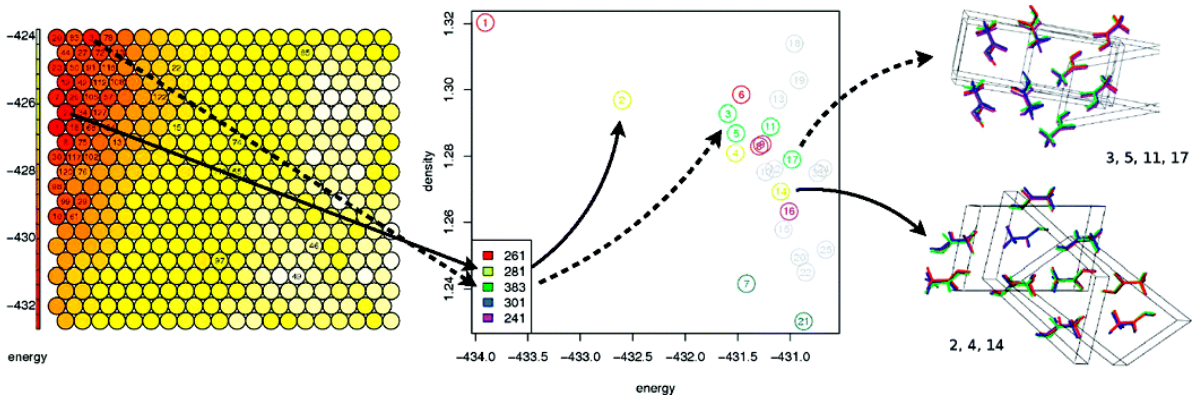
Supervised Self-Organizing Maps in Crystal Property and Structure Prediction

E. L. Willighagen, R. Wehrens, W. Melssen, R. de Gelder, and L. M. C. Buydens

Institute for Molecules and Materials, Radboud University Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands

Synopsis

Supervised self-organizing maps (SOMs) provide a new method to explore large numbers of crystal structures and to visualize structure−property relationships. Examples show how powder diffraction patterns and one or more structural properties determine the positions of crystal structures on the supervised SOMs, and show that the maps allow the classification of structures, prediction of properties, and subset selection in polymorph prediction.



Abstract

This article shows the use of supervised self-organizing maps (SOMs) to explore large numbers of experimental or simulated crystal structures and to visualize structure−property relationships. The examples show how powder diffraction patterns together with one or more structural properties, such as cell volume, space group, and lattice energy, are used to determine the positions of the crystal structures in the maps. The weighted cross-correlation criterion is used as the similarity measure for the diffraction patterns. The results show that supervised SOMs offer a better and more interpretable mapping than unsupervised SOMs, which makes exploration of large sets of structures easier and allows for the classification and prediction of properties. Combining diffraction pattern and lattice energy similarity using a SOM outperforms the separate use of those properties and offers a powerful tool for subset selection in polymorph prediction.

Phys. Rev. B (2014) 89, 075201
doi: 10.1103/PhysRevB.89.075201

Prediction of structure candidates for zinc oxide as a function of pressure and investigation of their electronic properties

D. Zagorac, J. C. Schön, J. Zagorac, and M. Jansen

Max Planck Institute for Solid State Research, Stuttgart, Germany

Abstract

In order to gain new insight in the ZnO system, we performed crystal structure prediction using simulated annealing with an empirical potential and local optimization on *ab initio* level, both at standard and elevated pressure. We have found the experimentally observed structure types [wurtzite (B4), sphalerite (B3), and rock salt (B1)] in agreement with previous research. In addition, many new interesting modifications were found in different regions of the energy landscape, such as the β-BeO type, the GeP type, the NiAs type, and the so-called "5-5" type modification. At extreme conditions (>150 GPa), we observe a CsCl (B2) type of structure, and as a possible intermediate phase along the NaCl (B1) → CsCl (B2) transition route, we suggest the α-WC (B_h) modification. Furthermore, we have investigated the electronic properties of ZnO structures. Our investigations offer new possibilities of tuning the band gap with pure zinc oxide by employing modifications with different structural arrangements.

3) Applications in the context of energy landscape investigations

Phys. Status Solidi B, (2010) 247, No. 1, 23–39
doi: 10.1002/pssb.200945246

Predicting solid compounds via global exploration of the energy landscape of solids on the *ab initio* level without recourse to experimental information

J. Christian Schön, Klaus Doll and Martin Jansen

Max-Planck-Institute for Solid State Research, Heisenbergstr. 1, 70569 Stuttgart, Germany

Abstract

Predicting which crystalline modifications can exist in a chemical system requires the global exploration of its energy landscape. Due to the large computational effort involved, in the past this search for sufficiently stable minima has been performed employing a variety of empirical potentials and cost functions followed by a local optimization on the *ab initio* level. However, this might introduce some bias favoring certain types of chemical bonding and entails the risk of overlooking important modifications that are not modeled accurately using empirical potentials. In order to overcome this critical limitation, it is necessary to employ *ab initio* energy functions during the global optimization phase of the structure prediction. In this paper, we review the current state of the field of structure prediction on the *ab initio* level.

J. Phys. Chem. C, 2012, 116 (31), pp 16726–16739
doi: 10.1021/jp3022375

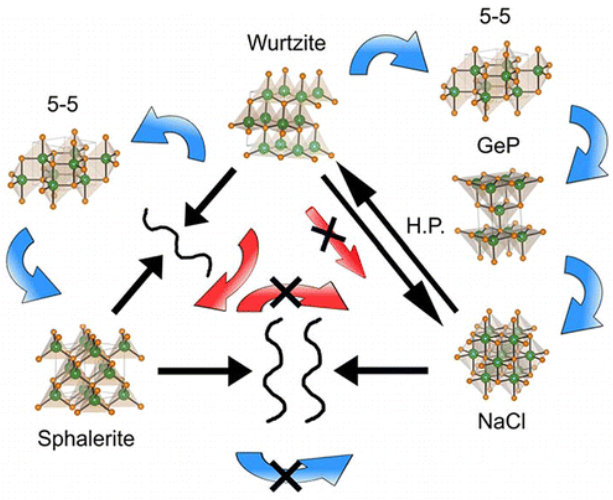
Energy Landscape Investigations Using the Prescribed Path Method in the ZnO System

Dejan Zagorac, J. Christian Schön, and Martin Jansen

Max Planck Institute for Solid State Research, Heisenbergstrasse 1, D-70569 Stuttgart, Germany

Abstract

An important issue in modern solid-state chemistry and nanotechnology is the development of a general methodology to predict possible (meta)stable crystalline and nanocrystalline modifications and to study the possible transition routes among them. To analyze the stability of the various potential modifications and study the possible low energy paths among them, the so-called prescribed path method is employed. This method allows us to explore transition routes and barriers between even distant minima, suggesting possible transition states and specific transition paths for more detailed analysis, as well as to gain more insights into the temperature dependence of the synthesis and transformation processes in the system. In this study, we describe and employ the prescribed path method for the example of the energy landscape of ZnO. The focus is on the influence of the temperature on the transformations along the path and on the stability of the various structures in the ZnO system, such as the wurtzite-, the sphalerite-, and the rock salt-type modifications. The results of our calculations are in good agreement with the experimental data, and we suggest several possible transition states such as the 5-5 and the GeP type along the B4–B1 transition path in agreement with previous calculations. This approach of analyzing transition route bundles among (predicted) modifications could be especially important for growing and eventually controlling synthesis and transformation of metastable nanocrystalline zinc oxide.



4) Applications in the context of drawing pictures and visualization

Proc. Appl. Ceram. 2015 Volume 9, Issue 3, Pages: 157-168
doi: 10.2298/PAC1503157S

Nanomaterials - What energy landscapes can tell us

Johann Christian Schön

Max Planck Institute for Solid State Research, Heisenbergstr. 1, D-70569 Stuttgart, Germany

Abstract

Nanomaterials bridge the gaps between crystalline material is, thin films, and molecules, and are of great importance in the design of new classes of materials, since the existence of many modifications of a nano-object for the same overall composition allows us to tune the proper ties of the nanomaterial. However, the structural analysis of nano-size systems is often difficult and their structural stability is frequently relatively low. Thus, a study of their energy landscape is needed to determine or predict possible structures, and analyse their stability, via the determination of the minima on the landscape and the generalized barriers separating them. In this contribution, we introduce the major concepts of energy landscapes for chemical systems, and present summaries of four applications to nano-materials: a MgO monolayer on a sapphire substrate, possible quasi-two-dimensional carbon-silicon networks, the ab initio energy landscape of Cu4Ag4-clusters, and the possible arrangements of ethane molecules on an ideally smooth substrate.

Keywords:
energy landscapes, MgO, graphene/silicene, nanostructures, intermetallic clusters, thin films

5) Applications in crystallographic analysis and structure determination

J. Phys. Chem. A, 2006, 110 (9), pp 3010–3016
doi: 10.1021/jp054807v

Low-Temperature Phases of Rubidium Silver Iodide: Crystal Structures and Dynamics of the Mobile Silver Ions

Klaus Funke[1], Radha D. Banhatti[1], Dirk Wilmer[1], Robert Dinnebier[2], Andrew Fitch[3] and Martin Jansen[2]

[1] University of Münster, Institute of Physical Chemistry and Sonderforschungsbereich 458, Corrensstrasse 30/36, 48149 Münster, Germany
[2] Max Planck Institute for Solid State Research, Heisenbergstrasse 1, 70569 Stuttgart, Germany
[3] European Synchrotron Radiation Facility, 6 rue Jules Horowitz, BP 220, 38043 Grenoble CEDEX, France

Abstract

Recently, broad-band conductivity spectra have been taken in the low-temperature γ-phase of the archetypal fast ion conductor $RbAg_4I_5$. Attempts to reproduce the experimental data in a simple model calculation have led to the conclusion that strictly localized displacive movements of interacting ionic charge carriers should play an important role in the low-temperature phase. However, with no detailed structural study of γ-$RbAg_4I_5$ available, the relevant processes could not be identified within the crystal structure. This state of affairs has triggered the present investigation of the structures of all three phases of rubidium silver iodide. Powder diffraction data of $RbAg_4I_5$ have been collected at the high-resolution powder diffractometer at ID31 at the European Synchrotron Radiation Facility (ESRF). The structure of the γ-phase has been solved by successive Rietveld refinements in combination with difference Fourier analyses. The same structural principle is found to prevail in all three phases, interconnected distorted RbI_6 octahedra forming a three-dimensional framework, which undergoes only displacive structural changes during the α−β and β−γ phase transitions. With decreasing temperature, the disorder in the silver sublattice is found to decrease, and a clustering of the disordered silver ions is found to develop. In the γ-phase, “pockets” containing partially occupied silver sites have been identified, and it is suggested that the localized displacive motion detected by conductivity spectroscopy is performed by the silver ions located within these pockets.

J. Phys. Chem. C, 2007, 111 (37), pp 13712–13720
doi: 10.1021/jp0735583

New B_2O_3 Crystals Predicted from Concurrent Molecular Dynamics Simulations and First-Principles Calculations

Liping Huang[1,3], Murat Durandurdu[2], and John Kieffer[3]

[1] Department of Materials Science and Engineering, UniVersity of Michigan, Ann Arbor, Michigan 48109-2136
[2] Department of Chemical and Biomolecular Engineering and Department of Physics, North Carolina State UniVersity, Raleigh, North Carolina 27695
[3] Department of Physics, UniVersity of Texas at El Paso, El Paso, Texas 79968

Abstract

Molecular dynamics (MD) simulations, based on a new coordination-dependent charge-transfer potential, were used to study the behavior of crystalline B_2O_3 in response to various thermal and mechanical constraints. This interaction potential allows for the charges on atoms to redistribute upon the formation and rupture of chemical bonds and dynamically adjusts to multiple coordination states for a given species. Our MD simulations predict that upon isotropic expansion and compression, the B_2O_3-I crystal transforms into new low- and high-density B_2O_3 crystals, the stability of which we have further verified using first-principles calculations. The low-density B_2O_3 crystals (B_2O_3-0) provide a key to understanding the anomalous thermomechanical behaviors of vitreous B_2O_3 and the crystallization anomaly of this compound. The high-density B_2O_3 crystal (B_2O_3-III), predicted from concurrent MD simulations and first-principles calculations, is different from the known high-pressure phase of B_2O_3-II crystal, even though the bonding is the same in these two phases. B_2O_3-III is characterized by a higher energy than B_2O_3-II at low pressures, but upon further compression the energies of these two phases become indistinguishable. The transformation from B_2O_3-I to B_2O_3-III appears to be kinetically favored, especially at low temperatures. Our studies indicate that the phase diagram of B_2O_3 is much richer than previously known.

6) Applications in the field of data mining

Chem. Mater., 2010, 22 (12), pp 3762–3767
doi: 10.1021/cm100795d

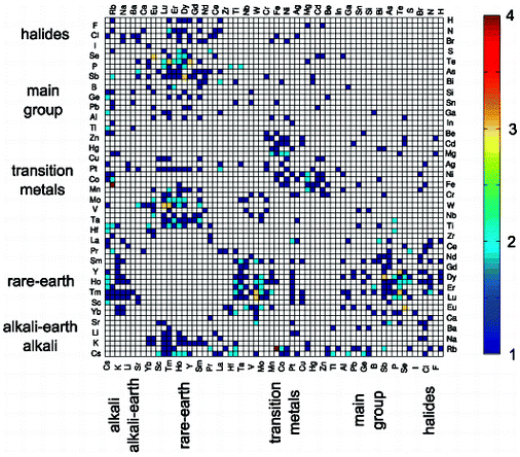
Finding Nature's Missing Ternary Oxide Compounds Using Machine Learning and Density Functional Theory

Geoffroy Hautier, Christopher C. Fischer, Anubhav Jain, Tim Mueller and Gerbrand Ceder

Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Synopsis

Using a combination of machine learning and *ab initio* computations, the ternary oxide chemical space has been searched for potential new compounds. From this search, we predicted 209 new compounds in various oxides chemistries.



Abstract

Finding new compounds and their crystal structures is an essential step to new materials discoveries. We demonstrate how this search can be accelerated using a combination of machine learning techniques and high-throughput *ab initio* computations. Using a probabilistic model built on an experimental crystal structure database, novel compositions that are most likely to form a compound, and their most-probable crystal structures, are identified and tested for stability by *ab initio* computations. We performed such a large-scale search for new ternary oxides, discovering 209 new compounds with a limited computational budget. A list of these predicted compounds is provided, and we discuss the chemistries in which high discovery rates can be expected.

Chem. Mater., 2011, 23 (15), pp 3495–3508
doi: 10.1021/cm200949v

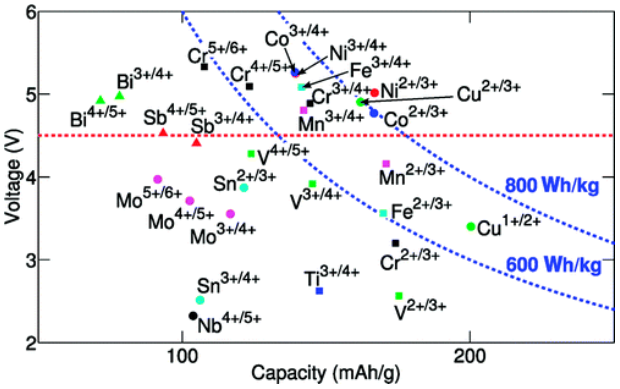
Phosphates as Lithium-Ion Battery Cathodes: An Evaluation Based on High-Throughput *ab Initio* Calculations

Geoffroy Hautier, Anubhav Jain, Shyue Ping Ong, Byoungwoo Kang, Charles Moore, Robert Doe, and Gerbrand Ceder

Massachusetts Institute of Technology, Department of Materials Science and Engineering, 77 Massachusetts Avenue, Cambridge, MA 02139

Synopsis

We use high-throughput *ab initio* computations to evaluate phosphates as Li-ion cathode materials. The limits and opportunities for the phosphate chemistry in terms of voltage, capacity (gravimetric and volumetric), specific energy, energy density, and safety are analyzed and discussed.



Abstract

Phosphate materials are being extensively studied as lithium-ion battery electrodes. In this work, we present a high-throughput *ab initio* analysis of phosphates as cathode materials. Capacity, voltage, specific energy, energy density, and thermal stability are evaluated computationally on thousands of compounds. The limits in terms of gravimetric and volumetric capacity inherent to the phosphate chemistry are determined. Voltage ranges for all redox couples in phosphates are provided, and the structural factors influencing the voltages are analyzed. We reinvestigate whether phosphate materials are inherently safe and find that, for the same oxidation state, oxygen release happens thermodynamically at lower temperature for phosphates than for oxides. These findings are used to recommend specific chemistries within the phosphate class and to show the intrinsic limits of certain materials of current interest (e.g., $LiCoPO_4$ and $LiNiPO_4$).

Keywords: *ab initio*; cathode; DFT; high-throughput; Li-ion battery; phosphates; safety; thermal stability

7) Influence on other analysis programs

J. Appl. Cryst. (2005). 38, 237-238
doi: 10.1107/S0021889804031528

FINDSYM: program for identifying the space-group symmetry of a crystal

H. T. Stokes and D. M. Hatch

Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602, USA

The aim of this program is to identify the space-group symmetry and give the lattice parameters and Wyckoff positions of the atoms in a standard setting, no matter what setting the original information is given in.

Keywords: space group symmetry; standard setting; computer programs; Wyckoff positions

Comp. Phys. Commun. (2012), Vol. 183, Issue 3, Pages 690–697
doi: 10.1016/j.cpc.2011.11.007

Identifying duplicate crystal structures: XtalComp, an open-source solution

David C. Lonie, Eva Zurek

Department of Chemistry, State University of New York at Buffalo, Buffalo, NY 14260-3000, United States

Abstract

We describe the implementation of XtalComp, an efficient, reliable, and open-source library that tests if two crystal descriptions describe the same underlying structure. The algorithm has been tested and found to correctly identify duplicate structures in spite of the "real-world" difficulties that arise from working with numeric crystal representations: degenerate unit cell lattices, numerical noise, periodic boundaries, and the lack of a canonical coordinate origin. The library is portable, open, and not dependent on any external packages. A web interface to the algorithm is publicly accessible at http://xtalopt.openmolecules.net/xtalcomp/xtalcomp.html.

Solution method: The XtalComp algorithm overcomes these issues to detect duplicate structures regardless of differences in representation. It begins by performing a Niggli reduction on the inputs, standardizing the translation vectors and orientations. A transform search is performed to identify candidate sets of rotations, reflections, and translations that potentially map the description of one crystal onto the other, solving the problems of enantiomorphs and rotationally degenerate lattices. The atomic positions resulting from each candidate transform are then compared, using a cell-expansion technique to remove periodic boundary issues. Computational noise is treated by comparing non-integer quantities using a specified tolerance.

Running time: The test run provided takes less than a second to complete

References:

[1] http://opensource.org/licenses/BSD-3-Clause.

Keywords: Duplicate; Structure; Crystal; Crystalline; Computational crystallography; Matching; Similarity

References:

1. Hannemann, A., Hundt, R., Schön, J. C. and Jansen, M. (1998). *J. Appl. Cryst.* **31**, 922-928.
2. Hundt, R., Schön, J. C., Hannemann, A. and Jansen, M. (1999). *J. Appl. Cryst*. **32**, 413-416.
3. Hundt, R., Schön, J. C. and Jansen, M. (2006). *J. Appl. Cryst.* **39**, 6-16.
4. Hundt, R., Schön, J. C., Neelamraju, S., Zagorac J. and Jansen, M., (2013), *J. Appl. Cryst.*, **46**, 587-593.
5. Willighagen, E. L., Wehrens, R., Melssen, W., de Gelder, R. and Buydens, L. M. C., (2007), *Cryst. Growth Des.*, **7**, 1738–1745.
6. Zagorac, D., Schön, J. C., Zagorac, J. and Jansen, M., (2014), *Phys. Rev. B,* **89**, 075201.
7. Schön, J. C., Doll, K. and Jansen, M. (2010), *Phys. Status Solidi B,* **247**, 23–39.
8. Zagorac, D., Schön, J. C., Jansen, M., (2012), *J. Phys. Chem. C*, **116**, 16726-16739.
9. Schön, J. C., (2015), *Proc. Appl. Ceram.,* **9**, 157-168.
10. Funke, K., Banhatti, R. D., Wilmer, D., Dinnebier, R., Fitch, A., Jansen, M., (2006), *J. Phys. Chem. A*, **110**, 3010-3016.
11. Huang, L., Durandurdu, M., Kieffer, J., (2007), *J. Phys. Chem. C*, **111**, 13712–13720.
12. Hautier, G., Fischer, C. C., Jain, A., Mueller, T., Ceder, G., (2010), *Chem. Mater.*, **22**, 3762–3767.
13. Hautier, G., Jain, A., Ong, S. P., Kang, B., Doe, R., Ceder, G., (2011), *Chem. Mater.*, **23**, 3495–3508.
14. Stokes, H. T. and Hatch, D. M. (2005). *J. Appl. Cryst.* **38**, 237-238.
15. Lonie, D. C., Zurek, E., (2012), *Comput. Phys. Commun.*, **183 (3)**, 690-697.