

CLUSTERGEN program manual (05/05/2013)

+++++

This manual treats only on how to use the *CLUSTERGEN* code itself which is written in Fortran programming language. It briefly explains options present in the input file and other minor issues. Description of algorithms and Graphical User Interface (GUI) is presented elsewhere.

+++++

First of all be sure you have the newest version of the compiler. Be aware that the code may not work properly with rather old compiler versions. The best is to use *gfortran* but *ifort* also works well (and it is even a bit faster). In order to perform typical fully optimized compilation:

```
gfortran -O3 -o clustergen *.f90
ifort -O3 -o clustergen *.f90
```

You may also add option `-static` to make it transferable between different computers. If something goes wrong during execution of the program (and for sure there are no errors in the input file) try:

```
gfortran -O0 -o clustergen-win32 -g -Wall -pedantic -fbounds-check *.f90
```

This is only for diagnostics (optimization is completely switched off) and, while running, the program will write the line of code where it crashed.

+++++

To run the program with a given input file just write for example (under Linux):

```
./clustergen input-file-name
```

+++++

The input file is in a free format. It consists of block of commands, which can be in any order. Each specific keyword given below (in green colour) is associated with a block. Every keyword consists of four capital letters. It is possible to place comments after an exclamation mark (!). Such lines are omitted, but only between different blocks, not within them. Variables in brackets are optional in some cases. Examples are in blue colour, followed by brief description of each case.

+++++

CLUS kbuild

kbuild – [integer {1,2,3}] – cluster building method: 1 – distance from molecular geometrical centre, 2 – distance from molecular geometrical centre + distance from geometrical centre to the most distant atom, 3 – anisotropic method based on the molecular shape.

CLUS 1 {the 1st cluster building model will be used}

+++++

FRGM kfragsft kfragbld kfragmentize

- 1) kfragsft – [integer {0,1}] – shifting of fragments to the basic unit cell (= 0 no, = 1 no).
- 2) kfragbld – [integer {0,1}] – completing fragments (= 0 no, = 1 no).
- 3) kfragmentize – [integer {0,1}] – control of automatic fragment search. If = 0 then no such search is invoked. If = 1 then fragments are assigned automatically. 1st atom on the list belongs always to the 1st fragment. If atoms are pre-assigned to fragments manually this will override those.

FRGM 1 1 1 {shifting and completing of fragments, automatic fragmentizing}

+++++

TEST ktestmol

ktestmol – [integer {0,1}] – control of writing of test geometry. If = 0 switched off. If = 1 then program writes *testmol.xyz* file with a final geometry and stops.

TEST 0 {no *testmol.xyz* file written}

+++++

XYZF kxyz

kxyz – [integer {0,1}] – control of XYZ file printing. If = 0 then no such file is created. If = 1 then XYZ file is printed.

XYZF 1 {write XYZ file}

+++++

VERB kver

kver – [integer {0,1,2}] – verbose variable. If = 0 then no additional information is written. If = 1, 2 some additional info is written into the output file.

VERB 0 {no additional information in output file}

+++++

CHRG kcharge

kcharge – [integer {0,1}] – control of charge reading. If = 0 no additional atom information is read. If = 1 then additional column after fractional coordinates is read and this information is written into the ADF input file (only if kadf = 1).

CHRG 0 {do not read charges}

+++++

ADFF kadf

kadf – [integer {0,1}] – control of ADF input file printing. If = 0 then no such file is created. If = 1 then ADF input file is printed.

ADFF 1 {write ADF input file}

+++++

SHLX kshelx

kshelx – [integer {0,1}] – control of SHELX-type RES file printing. If = 0 then no such file is created. If = 1 then RES file is printed. If kfragsft = 0 and kfragmentize = 0 then the RES file will have the symmetry operations written. Always “LATT -1” is printed. If kfragbld = 0 and kfragmentize = 0 then symmetry operations are written into the RES file.

SHLX 1 {write RES file}

+++++

DMAX dmax

dmax – [real] – maximal distance for cluster building in Å.

DMAX 6.0 {maximal distance is equal to 6.0 Å}

+++++

TOLS tol1 tol2 tol3

tol1 – [real] – tolerance for bonds searching in Å.

tol2 – [real] – tolerance for ring planarity in Å.

tol3 – [real] – tolerance for group planarity in Å.

TOLS 0.05 0.10 0.08

{bond tolerances are set to 0.05 Å, ring planarity to 0.10 Å, group planarity to 0.08 Å}

+++++

CELL a b c alpha beta gamma

a, b, c, alpha, beta, gamma – [real] – unit cell parameters in obvious order in Å and °.

CELL 5.678 6.456 9.567 90.000 101.345 90.000 {unit cell parameters}

+++++

GECR kgecr

kgecr – [integer {0,1}] – keyword controlling fixed geometrical centre. If = 1 then one extra line is read with fractional coordinates of the fixed geometrical centre. If = 0 then option is switched off.

GECR 1 {manually set the geometrical centre}

0.5 0.5 0.0 {fractional coordinates of a given point}

+++++

STXH kstxh (nxh)

1) kstxh – [integer {0,1}] – standardization of X–H bonds (= 1 yes, = 0 no).

2) (nxh) – [integer {...,-1,0,1,...}] – if ksrxh = 1 then this keyword is read in. If ksrxh = 0 then this keyword is ignored. This is the number of X–H bonds to be shifted. If = 0 then automatic shifting is invoked. If > 0 then nxh lines needs to be proceeding, each with numbers of X and H atoms (integers), respectively, and target distance (real) in

Å (numbers of atoms as in input file). If < 0 then automatic shifting will be performed followed by additional manual shifting (works in the same way as if nxh > 0, numbers of atoms from updated molecular table for all fragments)

```
STHX 1 -2 {perform both automatic and manual standardization for 2 bonds}
1 2 0.89 {bond between atoms 1 and 2 will be set to 0.89 Å}
5 6 1.34
```

+++++

```
SYMM nsym ksymmf
```

- 1) nsym – [integer {1,...}] – number of symmetry cards. For each symmetry operator this has to be proceeded with two lines (*i.e.* 2 × nsym lines in total) consisting of rotational and translational parts of the operators (all real numbers) or CIF-like symmetry format. You need to give all symmetry operators for a given space group for the program to work properly.
- 2) ksymmf – [integer {1,2}] – keyword controlling the format of symmetry operator. If = 1 then traditional matrix form is to be read. If = 2 then the CIF-like format should be provided. Letter size or spaces do not matter.

```
SYMM 2 2 {read 2 symmetry operators in a string format}
x, y, z {example operator}
-x, -y, -z+1/2
```

+++++

```
NFRG nat nfrag
```

- 1) nat – [integer {1,...}] – total number of atoms.
- 2) nfrag – [integer {1,...}] – number of fragments. This has to be proceeded with “FRAG fno natf” for each fragment. The latter one has to be proceeded with element type (must be capital), atom type, 3 fractional coordinates and optionally an atomic charge (when kcharge = 1).
- 3) fno – [integer {1,...}] – fragment number.
- 4) natf – [integer {1,...}] – number of atoms in the given fragment.

```
NFRG 3 2 {total of 3 atoms in 2 fragments will be given}
FRAG 1 2 {fragment number 1 with 2 atoms}
O O1 0.585183 0.556084 0.650164 {element, label and fractional coordinates}
C C1 0.455100 0.441286 0.551998
FRAG 2 1 {fragment number 2 with 1 atom}
H H2 0.578320 0.303230 0.887310
```

+++++

EXBD kexbd (nex)

- 1) kexbd – [integer {0,1}] – exclude atom pairs to be bonded. If = 0 then no such thing is done. If = 1 then the program will exclude those bonds which are specified during automatic fragment search (*i.e.* when kfragmentize = 1 and nfrag = 1). This will also work while automatic fragment completing routine (*i.e.* kfrgbld = 1)
- 2) (nex) – [integer {1,...}] – if kexbd = 1 then read this as a number of atom pairs to be excluded as bonded. This should be proceeded by nex lines specifying atom pairs (*e.g.* N O or C H etc.). There is no need to permute atoms in pairs. Letters must be capital.

EXBD 1 1 {exclude 1 bond}

K O {K-O bonds will be excluded}

+++++

CRYF kcrystal

kcrystal – [integer {0,1}] – control of CRYSTAL input file printing. If = 0 then no such file is created. If = 1 then CRYSTAL input file is printed.

CRYF 1 {write CRYSTAL input file}

+++++

RING kring (nrings) (...)

- 1) kring – [integer {0,1}] – control of ring specification. If =0 then no any other ring instruction will be read. If = 1 then it is assumed that nrings is given and this together with additional instructions are given properly.
- 2) (nrings) – [integer {1,...}] – if kring = 1 then read this as a number rings. This has to follow by the number of atoms in each rings. This should be proceeded by nrings lines specifying ring atoms.

RING 1 2 6 4 {assign two ring with 6 and 4 atoms, respectively}

3 4 5 6 8 9 {atom numbers forming ring no. 1 of size 6}

23 12 34 56 {atom numbers forming ring no. 2 of size 4}

+++++

NICS knics dnics nnics

- 1) knics – [integer {0,1}] – control of NICS points calculation. If = 0 then no such points are calculated. If = 1 then NICS points are automatically calculated and written in XYZ or GAUSSIAN files (if those are specified). This command is switched off if kring = 0.
- 2) dnics – [real] – maximal distance (in once direction) for NICS points generation in Å.
- 3) nnics – [integer {1,...}] – number of NICS points in one direction.

NICS 1 1.0 2 {calculate two NICS point positions with the maximal distance of 1.0 Å}

+++++

GAUF kxyz

kgaussian – [integer {0,1}] – control of GAUSSIAN file printing. If = 0 then no such file is created. If = 1 then GAUSSIAN file is printed.

GAUF 1 {write GAUSSIAN file}

+++++

DIMR kdimer

kdimer – [integer {0,1}] – control of dimer computation. If = 0 then no files are created. If = 1 then a number of XYZ files with dimer geometries are produced.

DIMR 1 {write XYZ files with dimer geometries}

+++++

Due to some issues with a line reading the last line needs to be blank (*i.e.* for example with one space).

+++++

Default options for the program may be absent in the input file. They are as follows:

TOLS 0.05 0.10 0.08

CLUS 1 FRGM 1 0 1 TEST 0 XYZF 1

VERB 0 CHRG 0 ADFF 0 SHLX 0

CRYF 0 GECR 0 STXH 1 0 EXBD 0

RING 0 NICS 0 GAUF 1

+++++

DMAX, CELL and SYMM commands and NFRG block are mandatory and there are no any default options for them.

+++++