TURBOMOLE

Program Package for ab initio

Electronic Structure Calculations

USER'S MANUAL

TURBOMOLE Version 4
Documentation 10/1997

Contents

Chapter 1

Preface

1.1 Contributions and Acknowledgements

TURBOMOLE has been designed by the Quantum Chemistry Group, University of Karlsruhe, Germany, since 1988. The following members of the group have made contributions:

Reinhart Ahlrichs, Michael Bär, Hans–Peter Baron, Rüdiger Bauernschmitt, Stephan Böcker, Michael Ehrig, Karin Eichkorn, Simon Elliott, Frank Haase, Marco Häser, Hans Horn, Christian Huber, Uwe Huniar, Marco Kattannek, Christoph Kölmel, Markus Kollwitz, Christian Ochsenfeld, Holger Öhm, Ansgar Schäfer, Uwe Schneider, Oliver Treutler, Malte von Arnim, Florian Weigend, Patrick Weis, Horst Weiss

Contact address:

Institut für Physikalische Chemie und Elektrochemie
Lehrstuhl für Theoretische Chemie
Universität Karlsruhe
Kaiserstr. 12
D–76128 Karlsruhe
E–mail: ramail@tchibm3.chemie.uni-karlsruhe.de

We acknowledge help from

- Michael Dolg, University of Stuttgart
- Jürgen Gauss, University of Mainz
- Christoph van Wüllen, University of Bochum
- Stefan Brode, BASF AG, Ludwigshafen
- Heinz Schiffer, HOECHST AG, Frankfurt

and financial support by the University of Karlsruhe, BASF AG, BAYER AG, HOECHST AG, the DFG, and the "Fonds der Chemischen Industrie".

The following publications describe details of the methodology implemented in **TURBOMOLE**:

1. Improvements on the Direct SCF Method. M. Häser and R. Ahlrichs; J. Comput. Chem. **10**, 104 (1989).
2. Electronic Structure Calculations on Workstation Computers: The Program System **TURBOMOLE**. R. Ahlrichs, M. Bär, M. Häser, H. Horn, and C. Kölmel; Chem. Phys. Letters **162**, 165 (1989).
3. Prescreening of Two–Electron Integral Derivatives in SCF Gradient and Hessian Calculations. H. Horn, H. Weiss, M. Häser, M. Ehrig, and R. Ahlrichs; J. Comput. Chem. **12**, 1058 (1991).
4. Exploiting Non–abelian Point Group Symmetry in Direct Two–electron Integral Transformations. M. Häser, J. Almlöf, and M. W. Feyereisen; Theor. Chim. Acta **79**, 115 (1991).
5. Fully Optimized Contracted Gaussian Basis Sets for Atoms Li to Kr. A. Schäfer, H. Horn, and R. Ahlrichs; J. Chem. Phys. **97**, 2571 (1992).
6. Direct Computation of Second–order SCF properties of Large Molecules on Workstations with an Application to Large Carbon Clusters. M. Häser, R. Ahlrichs, H. P. Baron, P. Weis, and H. Horn; Theor. Chim. Acta **83**, 455 (1992).
7. A Direct Algorithm for Self–consistent–field Linear Response Theory and Application to C_{60}: Excitation Energies, Oscillator Strengths, and Frequency–dependent Polarizabilities. H. Weiss, R. Ahlrichs, and M. Häser; J. Chem. Phys. **99**, 1262 (1993).
8. Semidirect MP2 Gradient Evaluation on Workstation Computers: The **MPGRAD** Program. F. Haase and R. Ahlrichs; J. Comp. Chem. **14**, 907 (1993).
9. Parallel Direct SCF and Gradient Program for Workstation Clusters. S. Brode, H. Horn, M. Ehrig, D. Moldrup, J. .E. Rice, and R. Ahlrichs; J. Comp. Chem. **14**, 1142 (1993).
10. Fully Optimized Contracted Gaussian Basis Sets of Triple Zeta Valence Quality for Atoms Li to Kr. A. Schäfer, C. Huber, and R. Ahlrichs; J. Chem. Phys. **100**, 5829 (1994).
11. Efficient Molecular Numerical Integration Schemes. O. Treutler and R. Ahlrichs; J. Chem. Phys. **102**, 346 (1995).
12. Auxiliary Basis Sets to Approximate Coulomb Potentials (Chem. Phys. Letters **240**, 283 (1995)). K. Eichkorn, O. Treutler, H. Öhm, M. Häser, and R. Ahlrichs; Chem. Phys. Letters **242**, 652 (1995).

13. **Turbomole**, Parallel Implementation of SCF, Density Functional, and Chemical ift Modules. R. Ahlrichs and M. v. Arnim, in: "Methods and Techniques in Computational Chemistry: METECC–95", E. Clementi, G. Corongiu, eds.

14. Stability Analysis for Solutions of the Closed Shell Kohn–Sham Equation. R. Bauernschmitt and R. Ahlrichs; J. Chem. Phys. **104**, 9047 (1996).

15. Treatment of Electronic Excitations within the Adiabatic Approximation of Time Dependent Density Functional Theory. R. Bauernschmitt and R. Ahlrichs; Chem. Phys. Letters **256**, 454 (1996).

16. Calculation of excitation energies within time-dependent density functional theory using auxiliary basis set expansions. R. Bauernschmitt, M. Häser, O. Treutler and R. Ahlrichs; Chem. Phys. Letters **264**, 573 (1997).

17. Auxiliary basis sets for main row atoms and transition metals and their use to approximate Coulomb potentials. K. Eichkorn, F. Weigend, O. Treutler and R. Ahlrichs; Theor. Chem. Acc. **97**, 119 (1997).

18. RI-MP2: first derivatives and global consistency. F. Weigend and M.Häser; Theor. Chem. Acc. **97**, 331 (1997).

19. A direct implementation of the GIAO-MBPT(2) method for calculating NMR chemical shifts. Application to the naphthalenium and and anthracenium ions. M. Kollwitz and J. Gauss; Chem. Phys. Letters **260**, 639 (1996).

1.2 Features of Turbomole

Turbomole has been specially designed for UNIX workstations and PCs and efficiently exploits the capabilities of this type of hardware. **Turbomole** consists of a series of modules; their use is facilitated by various tools.

Outstanding features of **Turbomole** are

- semi–direct algorithms with adjustable main memory and disk space requirements
- full use of all point groups
- efficient integral evaluation
- stable and accurate grids for numerical integration
- low memory and disk space requirements

1.3 Modules and Their Functionality

Define interactive input generator which creates the input file *control*. **Define** supports most basis sets in use, especially the only fully atom optimized consistent basis sets of SVP and TZV quality (**Turbomole** publications 5, 10 and 17) available for the atoms H–Rn, excluding lanthanides. **Define** also determines internal coordinates allowing efficient geometry optimization (fused rings may cause problems). **Define** sets the keywords necessary for single point calculations and geometry optimizations within a variety of methods. There are also many features to manipulate geometries of molecules: just try and see how it works.

Dscf for semi–direct SCF and DFT calculations (see keywords for functionals supported). **Dscf** supports restricted closed shell (RHF), spin restricted ROHF as well as UHF runs. **Dscf** includes an in–core version for small molecules.

Grad requires a successful **Dscf** run and calculates the gradient of the energy with respect to nuclear coordinates for all cases treated by **Dscf**.

Ridft and **Rdgrad** perform DFT calculations — as **Dscf** and **Grad** — with the additional approximation that the total density is approximated by a sum of atom centered s, p, d ... functions — the auxiliary (or fitting) basis. This allows a very efficient treatment of Coulomb interactions. The following functionals are supported: S–VWN, B–P, B–LYP. (See publications 12, 17).

Mpgrad requires a well converged SCF run — by **Dscf**, see keywords — and performs RHF closed shell or UHF calculations yielding single point MP2 energies and, if desired, the corresponding gradient.

Relax requires a gradient run — by **Grad**, **Rdgrad** or **Mpgrad**— and proposes a new structure based on the gradient and the approximated force constants. The approximated force constants will be updated.

Frog executes one molecular dynamics (MD) step. Like **Relax**, it follows a gradient run: these gradients are used as classical Newtonian forces to alter the velocities and coordinates of the nuclei.

Aoforce and **Uhfforce** require a well converged SCF run — by **Dscf**, see keywords — and perform analytic RHF closed shell (**Aoforce**) or UHF (**Uhfforce**) calculations of force constants, vibrational frequencies, IR intensities and Raman cross sections (Raman cross sections only for SCF closed shells). Note that DFT is not supported at present. The numerical calculation of force constants is also possible (see tool **numforce** in section 1.4).

Escf requires a well converged SCF or DFT run and calculates time dependent and dielectric properties:

– static and frequency–dependent polarizabilities within the SCF approximation, RHF and UHF

– static and frequency–dependent polarizabilities within the closed shell DFT formalism, including hybrid functionals such as B3LYP

– electronic excitations within the RHF and UHF CI(S) restricted CI method

– electronic excitations within the so–called SCF–RPA approximation (poles of the frequency dependent polarizibility)

– electronic excitations within the time dependent DFT formalism (adiabatic approximation), for closed shells only. It can be very efficient to use the RI approximation here, provided that the functional is of non–hybrid type: we recommend B–P86 (but slightly better results are obtained for the hybrid functional B3–LYP). (publication 15).

–stability analysis of single–determinant wave functions (second derivative of energy with respect to orbital rotations) (publication 14).

Mpshift requires a converged SCF run for closed shells. **Mpshift** computes NMR chemical shieldings for all atoms of the molecule at the MP2 level within the GIAO ansatz and the (CPHF) SCF approximation. From this one gets the NMR chemical shifts by comparison with the shieldings for the standard compound usually employed for this purpose, e. g. TMS for carbon shifts. Note that NMR shielding typically requires better basis sets than necessary for geometries or energies. ECP's are not supported in **Mpshift**Furthermore, NMR shieldings are very sensitive to correlation effects. A large "paramagnetic contribution", e. g. of the order of "diamagnetic contribution" is a serious reason to worry about the accuracy. (See publication 19).

Moloch computes a variety of first–order properties and analyses of the wavefunction as can be seen from the keywords.

Rimp2 calculates MP2 energies for RHF wavefunctions, using the RI technique (publication 18).

1.4 Tools

This is a brief description of additional **Turbomole** tools. Further information will be available by running the programs with the argument `-help`.

actual please use: `actual -help`

bend example: `bend 1 2 3`
displays the bending angle of three atoms specified by their number

from the *control* file. Note that unlike in the **TURBOMOLE** definition of internal coordinates the apex atom is the second!

CONVGREP greps lines for convergence check out of *control* file

DIST example: `dist 1 2`
calculates atomic distances from **TURBOMOLE** input files; `dis -l 4` gives all interatomic distances to 4 a. u. (5 a. u. is the default).

EIGER displays orbital eigenvalues obtained from data group `$scfmo`.

FINIT initialises the force constant matrix for the next relax.

FREEH calculates thermodynamic functions from molecular data in a *control* file; an **AOFORCE** run is a necessary pre–requisite.

HCORE prepares the *control* file for a Hamilton core guess (RHF only).

HOLUMO displays the highest occupied and the lowest unoccupied orbital.

JOBEX usage: see section 3.3
is the **TURBOMOLE** driver for all kinds of optimizations.

KDG example: `kdg scfdiis`
kills a data group (here `$scfdiis`) in the *control* file.

KONTO interface between moloch grid output and gle graphics; perl is required, please adjust the path at the top of the script.

MP2PREP prepares mp2–calculations interactively by adjusting parameters of the *control* file according to your system resources.

MDPREP interactive program to prepare for an MD run, checking in particular the *mdmaster* file (**MDPREP** is actually a FORTRAN program).

LOG2X converts the file logging an MD trajectory into coordinates in frames appropriate for *xmol* animation program.

LOG2EGY extracts the energy data (KE, total energy, PE) from an MD log file.

OUTP displays the angle between an atom and the a plane.

SCREWER distorts a molecule along a vibrational mode; IBM only.

SDG shows data group from *control* file:
for example `sdg energy` shows the list of calculated energies.

SYSNAME necessary for **TURBOMOLE** installation.

STATI prepares the *control* file for a statistics–run.

TORS is a script to query a dihedral angle in a molecular structure:
e.g. `tors 1 2 3 4` gives the torsional angle of atom 4 out of the plane of atoms 1, 2 and 3.

Chapter 2

Installation of Turbomole

Installation requires familiarity with some simple UNIX commands. The Turbomole package is generally shipped as one *tar* file. This has to be uncompressed

```
gunzip turbomole.tar.gz
```

and unpacked

```
tar -xvf turbomole
```

to produce the whole directory structure. Executable modules are in the `bin/`[arch] directory (for example, IBM modules are in `bin/ibm`). Tools (including Jobex) are in `scripts` and basis sets are kept in the directories `basen`, `jbasen` and `cbasen`. Coordinates for some common chemical fragments are supplied in `structures`.

The environmental variable `$TURBODIR` must be set to the directory where Turbomole has been unpacked, for example:

```
TURBODIR=/my_disk/my_name/my_turbo
```

Check that the sysname tool works on your computer:

```
$TURBODIR/scripts/sysname
```

should return the name of your system and this should match a `bin/`[arch] subdirectory.

You can call Turbomole executables and tools easily from anywhere if you add the corresponding directories to your path (korn shell):

```
PATH=$PATH:$TURBODIR/bin/[arch]:$TURBODIR/scripts
```

Now the Turbomole executables can be called from a directory with the required input files. For example to call **Dscf** and save the output:

```
$TURBODIR/bin/dscf > dscf.out
```

or if the path is OK, simply

```
dscf > dscf.out
```

In addition, some sample calculations are supplied in `TURBOTEST` so that the modules can be tested. Just run `TEST` from this directory to get help on how this works.

Chapter 3

How to Run TURBOMOLE

All **TURBOMOLE** modules need the *control* file as input file. The *control* file provides directly or by cross references the information necessary for all kinds of runs and tasks (see section 4). The *control* file is usually generated by **DEFINE**. We recommend generating a set of cartesian coordinates for the desired molecule using special molecular design software and converting this set into **TURBOMOLE** format (see section 4.3.2) as input for **DEFINE**.

3.1 DEFINE

DEFINE is the general interactive input generator of **TURBOMOLE**. During a session with **DEFINE**, you will create the *control* file which controls the actions of all other **TURBOMOLE** programs. During your **DEFINE** session you will be guided through four main menus:

1. **The geometry main menu**: This first menu allows you to build your molecule, define internal coordinates for geometry optimizations, determine the point group symmetry of the molecule, adjust internal coordinates to the desired values and related operations. After leaving this menu, your molecule to be calculated should be fully specified.
2. **The atomic attributes menu**: Here you will have to assign basis sets and/or effective core potentials to all atoms. The SV(P) basis is assigned as the default.
3. **The occupation numbers and start vectors menu**: In this menu you should chose `eht` to start from Extended Hückel MO vectors. Then you have to define the number of occupied orbitals in each irreducible representation.
4. **The general menu**: The last menu manages a lot of control parameters for all **TURBOMOLE** programs.

Most of the menu commands are self–explanatory and will only be discussed briefly. Typing `*` terminates the current menu and leads to the next while typing `&` goes back to the previous menu.

3.1.1 The Geometry Main Menu

After some preliminaries providing the title etc. you reach the geometry main menu:

```
SPECIFICATION OF MOLECULAR GEOMETRY ( #ATOMS=0     SYMMETRY=c1  )
YOU MAY USE ONE OF THE FOLLOWING COMMANDS :
sy        : DEFINE MOLECULAR SYMMETRY
desy <eps>: DETERMINE MOLECULAR SYMMETRY AND ADJUST
            COORDINATES (default for eps=1d-6)
susy      : ADJUST COORDINATES FOR SUBGROUPS
ai        : ADD ATOMIC COORDINATES INTERACTIVELY
a <file>  : ADD ATOMIC COORDINATES FROM FILE <file>
aa <file> : ADD ATOMIC COORDINATES IN ANGSTROEM UNITS FROM FILE <file>
sub       : SUBSTITUTE AN ATOM BY A GROUP OF ATOMS
i         : INTERNAL COORDINATE MENU
m         : MANIPULATE GEOMETRY
w <file>  : WRITE MOLECULAR COORDINATES TO FILE <file>
name      : CHANGE ATOMIC IDENTIFIERS
del       : DELETE ATOMS
dis       : DISPLAY MOLECULAR GEOMETRY
banal     : CARRY OUT BOND ANALYSIS
*         : TERMINATE MOLECULAR GEOMETRY SPECIFICATION
            AND WRITE GEOMETRY DATA TO CONTROL FILE

IF YOU APPEND A QUESTION MARK TO ANY COMMAND AN EXPLANATION
OF THAT COMMAND MAY BE GIVEN
```

This menu allows you to build your molecule by defining the cartesian coordinates interactively (`ai`) or by reading the coordinates from an external file (`a`,`aa`). The structure can be manipulated by the commands `sub`, `m`, `name` and `del`. The command `sy` allows you to define the molecular symmetry while `desy` determines automatically the symmetry group of a given molecule. For the definition of internal coordinates enter `i` to obtain the internal sub–menu. There exists a structure library which contains the cartesian coordinates of selected molecules e. g. CH_4. These data can be obtained by typing for example `a ! ch4` or `a ! methane`. The data files are to be found in the directory '$TURBO/structures'. The library can be extended.

3.1.2 The Atomic Attributes Menu

```
ATOMIC ATTRIBUTE DEFINITION MENU  ( #atoms=5     #bas=5     #ecp=0     )

b    : ASSIGN ATOMIC BASIS SETS
bb   : b RESTRICTED TO BASIS SET LIBRARY
bl   : LIST ATOMIC BASIS SETS ASSIGNED
bm   : MODIFY DEFINITION OF ATOMIC BASIS SET
bp   : SWITCH BETWEEN 5d/7f AND 6d/10f
```

```
lib  : SELECT BASIS SET LIBRARY
ecp  : ASSIGN EFFECTIVE CORE POTENTIALS
ecpb : ecp RESTRICTED TO BASIS SET LIBRARY
ecpi : GENERAL INFORMATION ABOUT EFFECTIVE CORE POTENTIALS
ecpl : LIST EFFECTIVE CORE POTENTIALS ASSIGNED
ecprm: REMOVE EFFECTIVE CORE POTENTIAL(S)
c    : ASSIGN NUCLEAR CHARGES (IF DIFFERENT FROM DEFAULTS)
cem  : ASSIGN NUCLEAR CHARGES FOR EMBEDDING
m    : ASSIGN ATOMIC MASSES (IF DIFFERENT FROM DEFAULTS)
dis  : DISPLAY MOLECULAR GEOMETRY
dat  : DISPLAY ATOMIC ATTRIBUTES YET ESTABLISHED
h    : EXPLANATION OF ATTRIBUTE DEFINITION SYNTAX
*    : TERMINATE THIS SECTION AND WRITE DATA OR DATA REFERENCES TO control
GOBACK=& (TO GEOMETRY MENU !)
```

Unless your *control* file already contains basis set specifications, **Define** will automatically assign standard basis sets SV(P) and appropriate ECPs for atoms heavier than Kr (listing by `bl` and `ecpl`). SV(P) means split valence for all atoms except H, where SV is used. SV(P) is comparable to 6–31g*. If other basis sets and/or ECPs are desired, use `b`, e. g. `b all def--TZVP` and/or `ecp`. You also are able to assign non–default masses or nuclear charges to the atoms in your molecule (`c`, `m`) as well as nuclear charges for embedding (`cem`).

3.1.3 The MO Start Vectors Menu

```
OCCUPATION NUMBER & MOLECULAR ORBITAL DEFINITION MENU

CHOOSE COMMAND
infsao     : OUTPUT SAO INFORMATION
eht        : PROVIDE MOS && OCCUPATION NUMBERS FROM EXTENDED HUECKEL GUESS
use <file> : SUPPLY MO INFORMATION USING DATA FROM <file>
man        : MANUAL SPECIFICATION OF OCCUPATION NUMBERS
hcore      : HAMILTON CORE GUESS FOR MOS
&          : MOVE BACK TO THE ATOMIC ATTRIBUTES MENU
THE COMMANDS  use  OR  eht  OR  *  OR q(uit) TERMINATE THIS MENU !!!
FOR EXPLANATIONS APPEND A QUESTION MARK (?) TO ANY COMMAND
```

This menu generates start vectors for SCF and DFT calculations. They may be constructed from earlier SCF calculations (perhaps employing another basis set, type `use`), by Hamilton core guess (`hcore`), or by an extended Hückel calculation which can be performed automatically (`eht`). An occupation of the start orbitals will be proposed and can be modified if desired.

3.1.4 The General Menu

```
GENERAL MENU : SELECT YOUR TOPIC
```

```
scf    : SELECT NON-DEFAULT SCF PARAMETER
prop   : SELECT TOOLS FOR SCF-ORBITAL ANALYSIS
drv    : SELECT NON-DEFAULT INPUT PARAMETER FOR EVALUATION
         OF ANALYTICAL ENERGY DERIVATIVES
         (GRADIENTS, FORCE CONSTANTS)
rex    : SELECT NON-DEFAULT OPTIONS FOR GEOMETRY UPDATES
e      : DEFINE EXTERNAL ELECTROSTATIC FIELD
misc   : PROVIDE SPECIAL STUFF ($lock, $statistics, ...)
para   : PROVIDE STUFF FOR PARALLELIZED DSCF/GRADIENT RUNS
dft    : DFT Parameters
ri     : RI Parameters
dis    : DISPLAY MOLECULAR GEOMETRY
list   : LIST OF CONTROL FILE
&      : GO BACK TO OCCUPATION/ORBITAL ASSIGNMENT MENU
* or q : END OF DEFINE SESSION
```

The last menu provides a lot of control parameters for all **Turbomole** programs. Note that extra commands for **Aoforce** do not appear here and so have to be set 'by hand'. Some of the commands in this menu will lead you to extensive submenus. Note for example that `dft` provides the necessary input for density functional calculations and that `ri` governs DFT calculations using the fast Coulomb approximation (publications 12 and 17).

3.2 Single Point Calculations

3.2.1 Running Turbomole Modules

All calculations are carried out in a similar way. First you have to run **Define** to obtain the *control* file or to add/change the keywords you need for your purpose. This can also be done manually with an editor. Given a korn shell and a path to `$TURBODIR/bin/`[arch] (see installation, section 2) you call the appropriate module in the following way (e. g. module **Dscf**):

```
nohup dscf > dscf.out &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. The output will be written to the file *dscf.out*. Several modules write some additional output to the *control* file. For the required keywords see section 4.2. The features of **Turbomole** will be described in the following section.

3.2.2 Energy and Gradient Calculations

Energy calculations may be carried out at different levels of theory.

- Hartree–Fock–SCF: use modules **DSCF** and **GRAD** to obtain the energy and gradient. The energy can be calculated after a **DEFINE** run without any further keywords or previous runs. The gradient calculation however requires a converged **DSCF** run.
- Density functional theory: DFT calculations are carried out in exactly the same way as Hartree–Fock calculations except for the additional keyword `$dft`. For DFT calculations with the fast Coulomb approximation you have to use the modules **RIDFT** and **RDGRAD** instead of **DSCF** and **GRAD**.Be careful: **DSCF** and **GRAD** ignore all RI flags and will try to do a normal calculation.
- MP2: the module **MPGRAD** calculates the MP2 energy as well as the energy gradient. If only the energy is desired use the keyword `$mp2energy`. MP2 calculations need well converged SCF runs (the SCF run has to be done with at least the density convergence `$denconv 1.d-7`, and `$scfconv 6` as described in section 4.2). For all further preparations run the tool **MP2PREP**.

For a semidirect **DSCF** calculation (Hartree–Fock or DFT) you first have to perform a statistics run. If you type

```
stati dscf
nohup dscf > dscf.stat &
```

the disk space requirement (MB) of your current `$thime` and `$thize` combination will be computed and written to the data group `$scfintunit size=<integer>` (see section 4.2.3). The requirement of other combinations will be computed as well and be written to the output file *dscf.stat*.

3.2.3 Calculation of Molecular Properties

See section 1.3 for the functionality and section 4.2 for the required keywords of the modules **AOFORCE**, **UHFFORCE**, **ESCF**, **MPSHIFT**, and **MOLOCH**.

3.3 Calculation of Molecular Structure: Shell Script JOBEX

In its normal mode of operation, the shell script **JOBEX** controls and executes automatic optimizations of molecular geometry parameters. It will cycle through the direct SCF, gradient and force relaxation programs and stop if either the maximum number of cycles is reached or the convergence criteria (change in the total energy, maximum norm of the gradient) are fulfilled. By default, the executable programs are taken from the load modules library within the **TURBOMOLE** directory.

3.3.1 Options

Given a korn shell the usage is:

```
nohup jobex -gcart 4 &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. **Jobex** accepts the following arguments (for example `-gcart 4` as above):

`-energy` <integer> converge total energy up to 10**(-<integer>) Hartree (default : `-energy 6`)

`-gcart` <integer> converge maximum norm of cartesian gradient up to 10**(-<integer>) atomic units (default : 3)

`-c` <integer> perform up to <integer> cycles (default : 20)

`-dscf` begin with a direct SCF step

`-grad` begin with a gradient step

`-relax` begin with a force relaxation step

`-level` <level> define the optimization level, <level> = `scf` (=default) or `mp2`

`-ri` use RI modules **Ridft** and **Rdgrad** (fast Coulomb approximation) instead of **Dscf** and **Grad** (DFT calculations only!)

`-l` <path> employ programs from directory <path>

`-md` a molecular dynamics (MD) run (using **Frog** instead of **Relax**).

`-mdfile` <filename> commands for MD run are contained in this file (default : `mdmaster`).

`-mdscript` <filename> option to execute a shell script before the **Frog** step.

3.3.2 Output

There will be an output written to file *job.start* which informs you about the current options. The convergence is signalled by means of the file *converged*; otherwise, you should find the file *not.converged* within your working directory. If **Jobex** finds a file named *stop* or *STOP* in the working directory, **Jobex** will stop after the present step has terminated. You create *stop* by `touch stop`.

The output of the last complete cycle is written to file *job.last*, while the output of the running cycle is collected within the file *job.*<cycle>, where <cycle> corresponds to the index of the current cycle.

In order to allow convergence control by **Jobex** and subsidiary tools some data groups are output to *control* by programs **Dscf** or **Mpgrad**, depending on the optimization subject and **Relax**:

Dscf → `$last SCF energy change = {real}`

Mpgrad → `$last MP2 energy change = {real}`

Relax → `$maximum norm of cartesian gradient = {real}`

Relax → `$maximum norm of internal gradient = {real}`

Relax → `$maximum norm of basis set gradient = {real}`

3.3.3 Molecular Dynamics

Ab initio molecular dynamics (MD) can be carried out on the ground state Born-Oppenheimer potential hypersurface. At the start of an MD run the user must specify the initial atomic positions and velocities and give some general instructions for the run. This is managed by running the interactive program MDPREP and generating the command file *mdmaster*. If this is successful, the MD run itself may be started: `jobex -md`. Time is then advanced in steps. The electronic potential energy and its gradients are calculated quantum mechanically at the required coordinates each timestep (as detailed above, e.g. **Dscf** and **Grad**). The MD program **Frog** uses the Leapfrog Verlet algorithm to turn the gradients into new atomic positions and velocities. The atoms thus undergo classical Newtonian dynamics on the ab initio potential hypersurface. Trajectory information is recorded in a log file (*mdlog*). It is possible to instruct **Frog** to heat or cool the system, conserve total energy or read in new positions or velocities: the appropriate keywords are described in section 4.2.10 below.

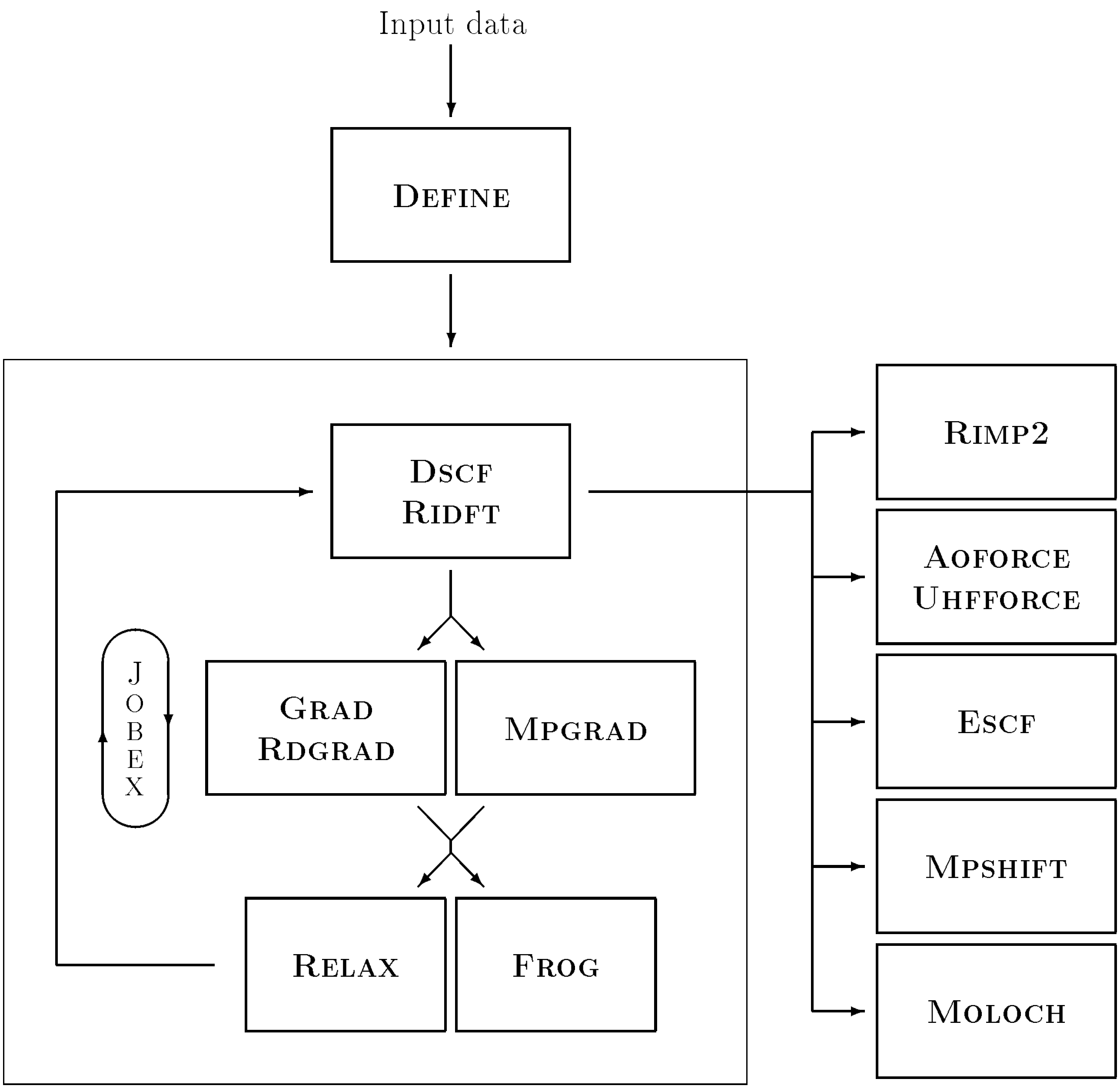

Figure 3.1: The modules of Turbomole and the main data flow between them

Chapter 4

The *control* file

4.1 Introduction

The file *control* is the input file for **Turbomole** which directly or by cross references provides the information necessary for all kinds of runs and tasks. *control* is usually generated by **Define**, the input generator. The following demo–file provides a short–hand documentation: a list of the most important key words, the possible parameters for each keyword, default values, and a brief explanation.

4.2 Keywords and Comments

Input is keyword–directed, keywords start with a `$`, e.g. `$title`. Comments may be given after `$dummy`, or by a line starting with `#`, these lines are ignored by **Turbomole**. Blank lines are also ignored.

Sample inputs given below should help to get an idea how the keywords are to be used.

4.2.1 General Keywords

```
# Keywords may be in any order except stated otherwise.
# The control file must end with: '$end' , see last line.

$operating system unix
$path
$lock off
$suspend off

# these 4 keywords are set by define but not necessary.

$statistics dscf
```

```
# or
$statistics mpgrad

# only a statistics run will be performed to determine file
# space requirements as specified for DSCF or MPGRAD
# on return the statistics option will be changed to
# '$statistics off'.

$actual step dscf

# means 'current step'. Keyword and data group (as e.g. 'dscf')
# set by every program and removed on successful completion.

$last step relax

# Keyword and data group (as e.g. 'relax')
# set by every program on successful completion.

$dummy -----  General file references  --------------------

$coord         file=coord
$intdef        file=coord
$basis         file=basis
$ecp           file=basis
$jbas          file=auxbasis
$scfmo         file=mos
$uhfmo_alpha   file=alpha
$uhfmo_beta    file=beta
$energy        file=energy
$grad          file=gradient
$forceapprox   file=force
$natural orbitals           file=natural
$natural orbital occupation file=natural


# It is convenient not to include all input in control directly
# and to refer to other files providing the corresponding
# information.
# The following cross references are default settings from define,
# you may use other file names. Define will create most of these files.
# Examples of these files are given below in the samples.

# coord (and intdef):
#      contains atom specification - type and location - and the
#      internal coordinates convenient for geometry optimizations.
```

```
# basis:
#      specification of basis sets.

# ecp:
#      specification of effective core potentials.

# jbas:
#      auxiliary (fitting) basis for the Coulomb terms in RIDFT.

# scfmo, uhfmo_alpha, uhfmo_beta:
#      MO vectors of SCF or DFT calculations for RHF or UHF runs.

# energy, grad:
#      energies and gradients of all runs, e. g. for documentation
#      in a geometry optimizations.

# forceapprox:
#      approximate force constant for geometry optimizations.

# natural orbitals, natural orbital occupation
#      keywords and data groups set by unrestricted DSCF or RIDFT
#      runs. Contain natural MO vector and orbital occupation.
```

4.2.2 Keywords for System Specification

```
# General information defining the molecular system:
# nuclear coordinates, symmetry, basis functions,
# number of occupied MOs, etc which are required
# by every module.

$title

# give title of run or project here.

$symmetry d4h

# Schoenflies symbol of the point group.
# all point groups are supported with the exception of
# NMR shielding calculations ....
# which do not work for groups with complex irreps
# (C3, C3h, T, etc).
# Use a lower symmetry group in this case.

$atoms
cu 1-4                                                          \
```

```
   basis =cu ecp-18 arep                                          \
   jbas  =cu ecp-18                                               \
   ecp   =cu ecp-18 arep
se 5-6                                                            \
   basis =se ecp-28 arep dzp                                      \
   jbas  =se ecp-28                                               \
   ecp   =se arep

# note the backslash '\' : this is necessary.
# for each type of atom, e.g. the Cu atoms with running numbers
# 1-4, one has to specify
# the basis set,
# the ECP if this is used,
# and the auxiliary (fitting) basis for RIDFT calculations.
# the files 'basis', 'ecp' and 'jbas'
# must provide the necessary information under the labels
# specified in '$atoms'.

$pople AO

# only proper (real) spherical harmonics are used as basis
# i.e. 5d, 7f, 9g, etc.
# if full cartesian sets are wanted - which is a bad habit
# and not recommended - put '$pople CAO'. This option
# is not supported by define!

$dummy -----   RHF    ------------------------------------

$closed shells

# Specification of MO occupation for RHF, e.g.
 a1g     1-4                                   ( 2 )
 a2g     1                                     ( 2 )

$open shells type=1

# MO occupation of open shells and number of open shells.
# 'type=1' here means that there is only a single open shell
# consisting e.g. of two MOs:
 b2g     1                                     ( 1 )
 b3g     1                                     ( 1 )

$roothaan         1
      a = 1      b = 2

# Roothaan paremeters for the open shell, here a triplet case.
```

```
# DEFINE recognises most cases and suggests good Roothaan parameters.

$dummy -----   UHF    ------------------------------------

$uhf

# directs the program to carry out a UHF run.

$alpha shells
 a1g     1-4                                   ( 1 )
 a2g     1                                     ( 1 )
$beta shells
 a1g     1-4                                   ( 1 )
 a2g     1                                     ( 1 )

# specification of MO occupation for UHF,
# $uhf overwrites closed shell occupation specification.
```

4.2.3 Keywords for Modules DSCF and RIDFT

```
$dummy ------   SCF-run specifications  ------------------

# If the SCF iterations should start by a core guess put

$scfmo none file=mos

# and analogously for alpha and beta MOs in UHF.
# The converged MOs are then written on the file specified (mos)
# and 'none' is deleted from the $scfmo line.

$scfiterlimit       30

#      maximum number of iterations

$scfconv        7

# convergence requirement: converge E to 1.d-7 au
# and one-electron energy to 1.d-3.
# you may use 6 (instead of 7) which usually suffices
# or 8 (9) for better converged iterations.
# Gradients will only be evaluated if scfconv > 6 !!!!!

$denconv 1.d-7

#      enforces the density matrix to be converged to 1.d-7
```

```
$thize      .10000000E-04
$thime        5

# thize and thime determine which batches of integrals are written
# on file in semi-direct runs: the smaller these numbers the more
# integrals will be saved on disk.

$scfdamp   start= .500  step= .050  min= .100

# for damping oscillations in SCF iterations, the old
# Fock-operator is added to the current one with weight
# 0.5 as start; if convergence is good, this weight is then
# reduced by the step 0.05 in each successive iteration
# until the minimum of 0.1 is reached.
# (These are the default settings of define for closed shell RHF).
# DSCF automatically tries to adjust the weight to optimize
# convergence but in difficult cases it is recommended to
# start with a large weight, e.g. 1.5, and to set the minimum
# to a larger value, e.g. 0.5.

$scforbitalshift automatic 0.1

# to assist convergence, the energies of unoccupied MOs are shifted
# to have a HOMO-LUMO gap of at least 0.1.
# this is default - you can delete the whole line! -
# but a larger shift may help to get better convergence.
# another possibility, for open shell cases, is to shift
# the energies of closed shell MOs down by 0.3 which is
# the default setting by define:
# $scforbitalshift  closedshell=.3

$scfdenapproxl 20

# To update the fock matrix each SCF iteration, we need to know how the
# density has changed over the last iteration. We can then save time by
# neglecting those integrals with a small density difference. Thus it can be
# advantageous to use a linear combination of densities from previous
# iterations, chosen so that the differential density is a minimum. The
# integer number (which is 20 here) is the number of terms in this combination.
# By using 1, the density difference used is the simply the density for
# last iteration minus that for the one before. The default 20 causes up to 20
# previous densities to be retained in the combination.

$incore 10
```

```
# By using this option the two-electron integrals are kept in RAM ;
# the number following the keyword specifies how many megabytes
# should be allocated (in this example, 10 MB).
# If the integrals exceed
# the RAM allocated the program reverts to the standard mode.
# Supports all methods which process two-electron integrals, ie:
#  SCF and DFT (including hybrid functionals);
#  RHF and UHF
# The following conditions must be met:
#  $scfdenapproxl 1
#  $symmetry c1
#   rhfshells 1 or 2
# It is advisable to set thize as small as possible
# (eg $thize 0.1d-08) and to remove the keyword $scfdump.

$scfdump

# This option causes the MOs to be saved each iteration in formatted
# form on file mos.

$scfdiis   start=0.5

#      default setting of define for the onset of DIIS
#      acceleration of SCF convergence.

$scfintunit
 unit=30       size=10       file=twoint

# scratch file for two-electron integrals in semi-direct
# calculations: size in MB. Make sure the file is local,
# otherwise integrals are transmitted over the network.
# size=0 leads to a fully direct run.
# size is set by a statistics run, see '$statistics'.
# DSCF switches to direct mode if the file space is exhausted.

$rundimensions
   dim(fock,dens)=6072
   natoms=6
   nshell=34
   nbf(CAO)=108
   nbf(AO)=98
   dim(trafo[SAO<-->AO/CAO])=256
   rhfshells=1

# rundimensions in SCF runs as provided here - set by define -
# must be grater or equal to those actually required, i. e.
```

```
# you can delete basis functions and keep rundimensions.
# This keyword is not necessary for small cases.

$fldopt
 1st derivative off
 2nd derivative off
 edelt =    .5000E-02
 fields on
 geofield off

# Specification of options related to external electrostatic fields.
# A missing suboption or a suboption followed by the flag 'off' will
# not be considered. The flag 'on' may be omitted.

# if 'field' is active you need

$electrostatic field
 .0 .0 .125

# which is the specification of the external electric field vector in a.u.

# For structure optimizations you additionally need
# the flag geofield 'on'.
# '1st derivative on' and '2nd derivative on' causes the computation
# of the numerical 1st and 2nd derivative of the energy d(E)/d(Field) and
# d^2(E)/d(Field^2) where edelt is the increment for the numerical
# differentiation.
# CAUTION: don't use the RI approximation for all these calculations
# since this will lead to non-negligible errors !!

$point_charges
0.2 0.2 0.4 2

establishes a point charge of +2 at the point 0.2 0.2 0.4.

$dummy -----   DFT    ------------------------------------

$dft
 functional b-p
 gridsize m3

# for DFT calculations one has to specify the functional
# and the grid (for the quadrature of the exchange correlation part).
# The settings above are default: both lines can be
# left out if the B-P86 functional and grid m3 are required.
# Other useful functionals supported are:
```

```
# b-lyp
# b3-lyp
# bh-lyp
# s-vwn
# Possible grids are 1 - 5 and m3 - m5
# where grid 1 is coarse (least accurate) and 5 most dense.
# We recommend however the use of so-called multiple grids m3 - m5:
# SCF iterations with grid 1 - 3, final energy and gradient with grid 3 - 5.
# Usually m3 is fine: for large or delicate systems, try m4.
# For a reference calculation with a very fine grid and very tight
# thresholds use 'reference' as grid specification instead of
# 'gridsize xy'.

$dummy -----    RI    ------------------------------------

# DSCF ignores RI flags: you must call the RI modules
# RIDFT and RDGRAD for energy and gradient calculations.

$ridft

# enforces a ridft calculation if module RIDFT is used

$ricore 180

# choose the memory core available (MB) for special arrays in the RI
# calculation (the less memory you give the more integrals are
# treated directly, i.e. recomputed on the fly in every iteration).

$jbas        file=auxbasis

# cross reference for the file specifying the auxiliary basis
# as referenced in '$atoms'.
# we strongly recommend using auxbasis sets optimized for the
# MO basis sets, e.g. use SVP (or TZVP)
# for the basis and the auxbasis set in define.

$ripop

# Calculation of atomic charges according to the s partial wave
# and atomic dipole moments according to the p partial wave
# as resulting from the aux-basis representation of the density.
```

4.2.4 Keywords for Modules Grad and Rdgrad

```
$drvopt
```

```
# This keyword and corresponding options are required in gradient
# calculations only in special circumstances.
# Just '$drvopt' is fine, no options needed
# to compute derivatives of the energy with respect
# to nuclear coodinates within the method specified:
# SCF, DFT, RIDFT.
```

4.2.5 Keywords for Modules Aoforce and Uhfforce

```
$forcepath path

# directs all scratch files to the directory 'path'. Alternatively,

$scratchpath
df          path2
DSD         path3
CPDRGMN     path4
GAI         path5
rhs         path6
ri          path7
vi          path8
dri         path9

# distributes different scratch files to different directories.

# directs all scratch files to the directories 'path', 'path2' and so on
# which should be on local disk.

$maxcor 50

# fixes the RAM memory to be used by the run (here 50 MB),
# about 70% of available memory should be fine.

# If Raman intensities are to be computed (only for closed shell)
# include
# raman  on
# in '$drvopt'.
```

4.2.6 Keywords for Module Escf

```
$dummy            escf-calculations

# to perform an escf calculation converged molecular orbitals from
# a HF, DFT or RIDFT calculation are needed. The HF, DFT or RIDFT
```

```
# method is chosen according to the $dft or $ridft keywords,
# specified above.

# Before starting escf, scratch files nfile* from previous escf
# runs should be removed.

# In an escf run one of the following properties can be calculated:
# (please note the 'or' in the text, do only one thing at a time.)

# 1. eigenvalues of singlet or triplet (HF+DFT,RHF)
#    or nonreal stability matrices (HF,RHF):

$scfinstab singlet 2
#or
$scfinstab triplet 2
#or
$scfinstab non-real 2
# the two smallest eigenvalues are calculated in this example.


# 2. CI singles excitation energies (HF,RHF+UHF):

#    singlet
$scfinstab ciss 3
#    triplet
$scfinstab cist 3
# the three smallest excitation energies are calculated
# in this example.

# 3. RPA excitation energies (HF,RHF+UHF):

#    singlet
$scfinstab rpas 2
#    triplet
$scfinstab rpat 2

# 4. excitation energies from time dependent DFT (adiabatic approx.)
#    (DFT+RIDFT,RKS)(same as preceding example, $dft or $ridft has to
#    be specified):

#    singlet
$scfinstab rpas 1
#    triplet
$scfinstab rpat 1

# 5. static polarizability tensors (HF+DFT,RHF).
```

```
$scfinstab polly

# 6. dynamic polarizability tensors at up to 10 different
#    frequencies w_i [in au] (HF+DFT,RHF, addiab. approx. for DFT).

$scfinstab dynpol 0.0 0.1 0.21345

#general keywords:

$soes b1
#      the irrep of the excited states to be calculated is b1.
#      Default: totally symmetric irrep.

$maxroot 1.145
#      escf terminates normally if the last converged excitation
#      energy is above 1.145 au.

$rpaconv 6
#      determines the residual vector (10^-6)
#      necessary for convergence. Larger values lead to higher
#      accuracy (default is 5).

$davidson
#      The Davidson algorithm will be used for HF-RPA calculation
#      instead of the modified Davidson algorithm.
#      We recommend the modified Davidson algorithm (the default for HF-RPA),
#      since it is faster and leads to an improved convergence,
#      but is more memory intensive.

$escfspace 4
#      before the actual Davidson algorithm starts, a vector space
#      consisting of the four (in this case) unit-vectors with the smallest
#      particle-hole energy differences will be constructed. Please note
#      that this feature is not yet implemented for UHF calculations
#      and HF-RPA calculations using the modified Davidson algorithm
#      (the default, see $davidson). The dimension of the start
#      vector space is restricted to 40. It is recommended to set its value
#      to a number slightly larger than the number of roots to be determined.

$escf_overview 30
#      An "overview calculation" is performed. This means that the
#      eigenvalues and oscillator strengths are calculated for the
#      30 (in this case) eigenvectors obtained in a sub space
#      constructed from the 30 unit-vectors with smallest particle-hole
#      energy difference. Please note that this feature is not yet implemented
```

```
#      for UHF calculations and HF-RPA calculations using the modified
#      Davidson algorithm (the default, see $davidson). The dimension
#      of the sub space is restricted to 40.

$upcrit 1000
#      Only for DEBUG purposes (experts only): changes the criterion for
#      an update within the algorithm of Olsen et al. (modified Davidson)
#      (is normally set to 7000).

$update
#      Only for experts: If a file update is found in the same directory
#      where the program ESCF is running, an UPDATE will be performed
#      in the next step (for DEBUG purposes sometimes important).
```

4.2.7 Keywords for Modules Mpgrad and Rimp2

```
$dummy -----  MP2 run intended? --------------------------
# If an MP2 run is to be performed after the SCF run:

# 1) The SCF run has to be done with at least the
#    density convergence '$denconv 1.d-7', and
# 2) '$scfconv 6' as described above.

$dummy -----  MPGRAD: how to perform ----------------------

# For an MP2 run perform the following 3 steps which are best
# done automatically in using the tool 'mp2prep'.

# 1) Insert at least the data groups $mointunit, $traloop and
#    $statistics mpgrad into the control file.
# 2) Call mpgrad: it will perform a statistics run. It will
#    - tell you in the output how much disk space will be needed.
#    - set the maximum disk space for each file in the data group
#      $mointunit (values in MB).
# 3) If the estimated disk space is not available: Increase the
#    value in the data group $traloop by one, reset the data group
#    "$statistics" to "$statistics mpgrad" and go to step 3.

#    Two possibilities: Either
#    - Call mpgrad: it will perform an MP2 energy or MP2 gradient
#      calculation or
#    - Call "jobex -level mp2" (other keywords possible): it will
#      perform an MP2 structure optimization.
```

```
$dummy -----  MPGRAD: important data groups ----------------------

$statistics mpgrad
#      Only a statistics run for the adjustment of the
#      file sizes will be performed.

$traloop        1
#      traloop specifies the number of loops (or 'passes') over
#      occupied orbitals performed in the semi-direct MP2 run:
#      the more passes the smaller file space requirements - but
#      CPU time will go slightly up.

$mointunit
 type=intermed unit=61      size=0        file=halfint
 type=1111     unit=62      size=0        file=moint#0
 type=1112     unit=63      size=0        file=moint#1
 type=1122     unit=64      size=0        file=moint#j
 type=1212     unit=65      size=0        file=moint#k
 type=1212a    unit=70      size=0        file=moint#a
 type=gamma#1  unit=71      size=0        file=gamma#1
 type=gamma#2  unit=72      size=0        file=gamma#2
 type=1212u    unit=73      size=0        file=moint#u
 type=1112u    unit=74      size=0        file=moint#v
 type=gamma#1u unit=75      size=0        file=gamma#1u

# The data group $mointunit specifies:
# - which scratch files are needed,
# - where they are located ( -> path name! ) and
# - (after a statistics run) how big the file sizes probably will be

$maxcor 50

#      The data group $maxcor adjusts the maximum size of core
#      memory which will be allocated during the MP2 run.
#      Recommendation: 2/3 of the actual main memory.

$freeze
 a1      1-2
 b1      1

# The data group $freeze specifies the frozen (core) orbitals.

$dummy ----- MPGRAD: optional data groups -----------------------

$mp2energy
```

```
# necessary if only the MP2 energy is to be calculated (no MP2 gradient!)

$mp2pair

# MP2 pair correlation energies will be calculated

$mp2fcm

# MP2 gradient and MP2 dipole moment
# will be dumped to the files GRD and DIPOL respectively.
# (reason: numerical MP2 force calculation needs these infos).

$dummy ----- RIMP2: additional keywords -----------------------------------

# rimp2 is only available for c1 symmetry and energy calculations
# auxiliary basis are given from the file 'rimp2basen'
# to be used in connection with SVP LCAO basis sets
# for elements h, li, be, b, c, n, o, f, p, s, cl.
#
# add in $atoms for each atomic type

 frococoauxbas =el TZVPPP              (el = element symbol)

# and specify

$frococoauxbas file=<rimp2basen>
$mp2energy
```

4.2.8 Keywords for Module Relax

```
$interconversion  off
   qconv=1.d-10
   maxiter=25

$optimize
   internal on
   cartesian  off

# These are defaults and thus not needed.
# For an optimization in cartesian coordinates put
#    'internal off' and 'cartesian  on'.

$coordinateupdate
   dqmax=0.3
```

```
   interpolate  on
   statistics 5

# these are the default settings of define which control the
# coordinate update and printout (and have proved reasonable):

# dqmax :    largest allowed change of an internal in au or degrees
# interpolate  on : perform interpolation of internals
#                   between 2 iterations if reasonable
# statistics 5 : print-out for the last 5 cycles.


$forceupdate
  ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.3
  threig=0.001  reseig=0.001  thrbig=1000.0 scale=1.00  damping=0.0

# Default settings of define for the update of the approximate
# Hessian required in geometry optimizations. numgeo grows with the number
# of relax runs.
# Although DEFINE sets 'dynamic fail' to 0.3 as above, convergence
# is usually better with a value of 0.5 to 0.8.

$forceinit on
   diag=default

# Initialize the Hessian to reasonable values. On execution
# this will be changed to '$forceinit off'. You should reset
# forceinit (to 'on') if the Hessian has been lost or appears
# to have been updated in an unreasonable way.
```

4.2.9 Keywords for Module Moloch

```
$properties
    trace                              off
    moments                            active
    potential                          off
    cowan-griffin                      off
    localization                       off
    population analyses                off
    plot                               off
    firstorder                         off
    fit                                off

# a missing suboption or a suboption followed by the flag 'off' will
# not be taken into account. The flag 'active' may be omitted.
```

```
# if 'moment' is active you need

$moments
  0th 1st 2nd 3rd
  point .0 .0 .0

# to compute the 0th, 1st, 2nd and 3rd moment at the reference point
# 0 0 0.

# if 'potential' is active you need

$points #1
  pot fld fldgrd shld
  point .0 .0 .0

# to compute the electrostatic potential (pot) and/or
# electrostatic field (fld) and/or electrostatic field gradient
# (fldgrd) and/or the zeroth order contribution to
# the diamagnetic shielding (shld) at reference point  0 0 0 .

# if 'population analyses' is active you need

$mulliken
 spdf molap netto irpspd irpmol mommul

# to perform a mulliken population analysis (the suboptions specify
# the output data)

# or

$paboon
 momao maodump maofile=mao all

# to perform a population analysis based on occupation numbers
# (the suboptions produce some output data concerning the modified
# atomic orbitals). To obtain shared electron numbers you further need

$shared electron numbers
 irreps
 orbitals
 2-center  threshold =    .0100
 3-center  threshold =    .0100
 4-center  threshold =    .0100

# to compute multi-center shared electron numbers .
```

```
# The suboptions irreps and orbitals produce some more output
# data.

# if 'plot' is active you need

$grid     #1
 mo 4a1g
  origin      .000000      .000000      .000000
 vector1     1.000000      .000000      .000000
 vector2      .000000     1.000000      .000000
 grid1 range    -5.000000     5.000000 points   100
 grid2 range    -5.000000     5.000000 points   100
 outfile = 4a1g

# to obtain two-dimensional plot data of mo 4 a1g
# (the plane is specified by origin and
# two vectors with grid range and number of grid points) which is written
# to file 4a1g. Several plots may be obtained (#1, #2 etc.) at the same
# time. Use tool 'konto' to visualize the plot.
```

4.2.10 Keywords for Module Frog

```
# The molecular dynamics (MD) program FROG needs a command file
# named mdmaster.

# The interactive MDPREP program manages the generation of
# mdmaster and associated files.  It is always a good idea to
# let MDPREP check over mdmaster before starting an MD run.

# ----------------------
#
# The following keywords are vital for FROG:

$nsteps  75
#                        Number of MD timesteps to be carried out
#                        $nsteps is decreased by 1 every time FROG is run.
#                        JOBEX -md stops when $nsteps reaches 0.
$natoms  9
#                        Number of atoms in system.
$current file=mdlog.aa
#                        The file containing the current position, velocity,
#                        time and timestep, that is, the input configuration.
#                        During an MD run the $log file is generally
#                        also the $current file.
```

```
$log file=mdlog.ZZ
#                        The file to which the trajectory should be logged.
#                        (position, velocity, gradient,
#                        time, timestep, energies).  This file can be
#                        manipulated with LOG2? tools afterwards.
#                        The extension (here ZZ) can be changed
#                        during a run with $label below.
$turbomole file=control
#                        Where to look for TURBOMOLE keywords $grad etc.
$end

# ----------------------
#
# These are optional keywords:

$seed  -123
#                        Integer random number seed
$title
na7o2
#                        Arbitrary title
$md file=control
#                        This command in mdmaster indicates that all
#                        other MD commands are to be found in an alternative
#                        file (in this example, in the control file)
$stack file=mdstack
#                        The kinetic energy (KE) control routines keep a record
#                        of recent KE and total energy in a "stack"
#                        in this file.  The instructions $temperature,
#                        $total_energy and $kinetic_energy (below) require
#                        a stack.  However $stack can also be activated
#                        by the user to prepare a stack before it is needed.
$comments
#                        Keeps comments (#) apart from keywords.

# ----------------------
#
# The following instructions tell FROG to change some aspect
# of the MD run at some point in time (t).  Times must be in ascending order.
# In all cases, the instruction list follows the keyword on the next line.

# Here are all the possible instructions:
$position
   file=? [filename]    t=?
$temperature
   constant
   ? <real>             t=?
```

```
$velocity
   random              t=?
   zero                t=?
   file=? [filename]   t=?
$total_energy
   constant
   ? <real>            t=?
$kinetic_energy
   constant
   ? <real>            t=?
$label
   ? <string>          t=?

# ----------------------

# The simplest instructions assign a new value to a variable at the
# specified time:
$temperature
   2400.0              t=10.0
   5000.0              t=210.0
   273.0               t=900.0
# Here the temperature is set to 2400K at time 10au, raised to 5000K
# 200au later and finally cooled to 273K at 900au.

# A new log filename can be specified with $label
$label
   a1                  t=1000
   a2                  t=2000
# This will create (and output to) mdlog.a1 at t=1000au and
# mdlog.a2 from t=2000au.

# As vectors, coordinates and velocities can not be given as a single value or
# string.  Therefore FROG expects a file from which the new vector can be read.
# The format should match that of the log files and atomic units are used.
$position
   file=coord1      t=50.0
   file=coord2      t=100.0
$velocity
   file=vel1        t=50.0

# Finally, there are some special instructions:

$velocity
   random              t=1000
   zero                t=2000
# At 1000au assign a random velocity, of magnitude roughly matching the old KE.
```

```
# Set the velocity to zero at 2000au.

$total_energy
   constant
# By using the energy stack, try to conserve total energy (fluctuations
# in total energy are an unavoidable consequence of the Leapfrog algorithm).
# Note that this is carried out every timestep (No t= is necessary).
```

4.3 NH_3 input for a RHF calculation

4.3.1 Main File *control*

```
$title
NH3 c3v SVP
$operating system unix
$symmetry c3v
$coord    file=coord
$intdef    file=coord
$atoms
n  1                                                                          \
   basis =n def-SVP
h  2-4                                                                        \
   basis =h def-SVP
$pople   AO
$basis    file=basis
$rundimensions
   dim(fock,dens)=495
   natoms=4
   nshell=15
   nbf(CAO)=30
   nbf(AO)=29
   dim(trafo[SAO<-->AO/CAO])=69
   rhfshells=1
$scfmo    file=mos
$closed shells
 a1      1-3                                ( 2 )
 e       1                                  ( 2 )
$scfiterlimit       30
$scfconv        7
$thize      .10000000E-04
$thime        5
$scfdamp   start= .500  step= .050  min= .100
$scfdump
$scfintunit
```

```
 unit=30       size=0        file=twoint
$scfdiis   start=0.5
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$last step     define
$end
```

4.3.2 File *coord*

```
$coord
     .00000000000000       .00000000000000       .54561506935122      n
    -.87806233111566      1.52084856970468      -.18187168978374      h
    -.87806233111566     -1.52084856970468      -.18187168978374      h
    1.75612466223131       .00000000000000      -.18187168978374      h
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    4    1            val=   1.90084
   2 k  1.0000000000000 bend    4    3    1       val= 106.27756
        1.0000000000000 bend    3    2    1
```

```
        1.0000000000000  bend     2    4    1
$end
```

4.3.3 File *basis*

```
$basis
*
n def-SVP
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
  1712.8415853     -.53934125305E-02
  257.64812677     -.40221581118E-01
  58.458245853     -.17931144990
  16.198367905     -.46376317823
  5.0052600809     -.44171422662
   1  s
  .58731856571      1.0000000000
   1  s
  .18764592253      1.0000000000
   3  p
  13.571470233     -.40072398852E-01
  2.9257372874     -.21807045028
  .79927750754     -.51294466049
   1  p
  .21954348034      1.0000000000
   1  d
  1.0000000000      1.0000000000
*
h def-SVP
# h     (7s) / [3s]     {511}
*
   3  s
  13.010701000      .19682158000E-01
  1.9622572000      .13796524000
  .44453796000      .47831935000
   1  s
  .12194962000      1.0000000000
   1  p
  .80000000000      1.0000000000
*
$end
```

4.3.4 File *mos*

```
$scfmo    expanded   format(4d20.14)
     1  a1      eigenvalue=-.15633041862301D+02   nsaos=10
 .98699003163455D+00-.47221435341751D-01 .55873125006179D-02-.48016374887169D-02
 .26746008768233D-02 .20823779196149D-03 .14270460008808D-01 .90849517503597D-02
 .58676121352806D-03 .29091871198884D-03
     2  a1      eigenvalue=-.99896275238736D+00   nsaos=10
 .26412162337482D+00 .51846472345768D+00 .37623729061179D+00-.77139882704089D-02
-.47252329287316D-02-.21494050853221D-02 .11795673774658D+00 .83316086019184D-01
-.11229203933488D-01-.27038186251429D-02
     3  a1      eigenvalue=-.57101279949392D+00   nsaos=10
-.35584199011701D-01-.96938258881594D-01-.70254605702716D-01 .65569041318341D+00
-.44746149963029D+00 .40094287741992D-03 .51691151834284D-01 .47722350097160D-01
 .19189122068531D-02 .56638497851180D-03
     1  e       eigenvalue=-.64374209294851D+00   nsaos=9
-.49313475446075D+00 .33757893447603D+00-.76142296567409D-04-.74524664248740D-04
-.26407572210452D+00-.22619038902975D+00-.50035170531670D-05-.12199166245418D-03
 .63021657663245D-04
$end
```

4.4 NO_2 input for an unrestricted DFT calculation

4.4.1 Main File *control*

```
$title
NO2 c2v UKS SVP
$operating system unix
$symmetry c2v
$coord    file=coord
$intdef    file=coord
$atoms
n  1                                                                            \
   basis =n def-SVP
o  2-3                                                                          \
   basis =o def-SVP
$pople   AO
$basis    file=basis
$rundimensions
   dim(fock,dens)=1098
   natoms=3
   nshell=18
   nbf(CAO)=45
   nbf(AO)=42
   dim(trafo[SAO<-->AO/CAO])=85
```

```
   rhfshells=2
$uhfmo_alpha none   file=alpha
$uhfmo_beta none  file=beta
# none : hamilton core guess will be made
# files alpha and beta will be generated by the program
$uhf
$alpha shells
 a1      1-6                                ( 1 )
 a2      1                                  ( 1 )
 b1      1-4                                ( 1 )
 b2      1                                  ( 1 )
$beta shells
 a1      1-5                                ( 1 )
 a2      1                                  ( 1 )
 b1      1-4                                ( 1 )
 b2      1                                  ( 1 )
$scfiterlimit       30
$scfconv        7
$thize      .10000000E-04
$thime        5
$scfdamp   start=1.500  step= .050  min= .100
$scfdump
$scfintunit
 unit=30       size=2        file=/work/user/twoint
$scfdiis   start=0.5
$scforbitalshift  closedshell=.3
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
```

```
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$dft
 functional b-p
 gridsize m3
$last step     define
$end
```

4.4.2 File *coord*

```
$coord
     .00000000000000       .00000000000000     -1.00494155217173      n
    1.85766051386774       .00000000000000       .50247077608587      o
   -1.85766051386774       .00000000000000       .50247077608587      o
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    2    1           val=   2.39232
   2 d  1.0000000000000 stre    3    1           val=   2.39232
   3 k  1.0000000000000 bend    2    3    1      val= 101.88429
$end
```

4.4.3 File *basis*

```
$basis
*
n def-SVP
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
  1712.8415853     -.53934125305E-02
  257.64812677     -.40221581118E-01
  58.458245853     -.17931144990
  16.198367905     -.46376317823
  5.0052600809     -.44171422662
   1  s
  .58731856571      1.0000000000
   1  s
  .18764592253      1.0000000000
   3  p
```

```
  13.571470233     -.40072398852E-01
  2.9257372874     -.21807045028
  .79927750754     -.51294466049
   1  p
  .21954348034      1.0000000000
   1  d
  1.0000000000      1.0000000000
*
o def-SVP
# o     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
  2266.1767785     -.53431809926E-02
  340.87010191     -.39890039230E-01
  77.363135167     -.17853911985
  21.479644940     -.46427684959
  6.6589433124     -.44309745172
   1  s
  .80975975668      1.0000000000
   1  s
  .25530772234      1.0000000000
   3  p
  17.721504317      .43394573193E-01
  3.8635505440      .23094120765
  1.0480920883      .51375311064
   1  p
  .27641544411      1.0000000000
   1  d
  1.2000000000      1.0000000000
*
$end
```

4.5 $TaCl_5$ input for an RIDFT calculation with ECPs

4.5.1 Main File *control*

```
$title
$operating system unix
$symmetry d3h
$coord    file=coord
$intdef    file=coord
$atoms
ta 1                                                                  \
   jbas=ta def-SVP                                                    \
   basis =ta def-SVP                                                  \
```

```
   ecp   =ta def-ecp
cl 2-6                                                                       \
   jbas=cl def-SVP                                                           \
   basis =cl def-SVP
$pople   AO
$basis    file=basis
$ecp    file=basis
$rundimensions
   dim(fock,dens)=7662
   natoms=6
   nshell=51
   nbf(CAO)=122
   nbf(AO)=115
   dim(trafo[SAO<-->AO/CAO])=346
$scfmo   none   file=mos
# none : hamilton core guess will be made
# file mos will be generated by the program
$scfiterlimit       30
$scfconv        6
$thize      .10000000E-04
$thime        5
$scfdamp   start= .900  step= .050  min= .100
$scfdump
$scfintunit
 unit=30       size=0        file=twoint
$scfdiis   start=0.5
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
```

```
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$dft
   functional b-p
   gridsize   m3
$last step     define
$ricore    20
$ridft
$jbas file=auxbasis
$closed shells
 a1'     1-11                                 ( 2 )
 a2'     1-2                                  ( 2 )
 e'      1-10                                 ( 2 )
 a2"     1-8                                  ( 2 )
 e"      1-4                                  ( 2 )
$end
```

4.5.2 File *coord*

```
$coord
     .00000000000000      .00000000000000      .00000000000000      ta
    2.19392179448315    -3.79998401587749      .00000000000000      cl
    2.19392179448315     3.79998401587749      .00000000000000      cl
   -4.38784358896629      .00000000000000      .00000000000000      cl
     .00000000000000      .00000000000000     4.46615918865523      cl
     .00000000000000      .00000000000000    -4.46615918865523      cl
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    1    2           val=   4.38784
   2 k  1.0000000000000 stre    1    5           val=   4.46616
$end
```

4.5.3 File *basis*

```
$basis
*
ta def-SVP
# ta    (7s6p5d) / [6s3p2d]     {211111/411/41}
*
```

```
   2  s
  14.400000000      .99343296745
  12.000000000     -1.6510077975
   1  s
  5.0701477302      1.0000000000
   1  s
  .86033356487      1.0000000000
   1  s
  .37158938894      1.0000000000
   1  s
  .10745336254      1.0000000000
   1  s
  .39142776556E-01  1.0000000000
   4  p
  7.4188720000      .26979695152
  5.6984100000     -.46968874449
  1.1777211960      .50905100155
  .54478533555      .52298161137
   1  p
  .22309270117      1.0000000000
   1  p
  .43100000000E-01  1.0000000000
   4  d
  3.9738796278     -.52799310714E-01
  1.4528884813      .18558319471
  .61042908544      .42959071631
  .24216276510      .43497228232
   1  d
  .87909318337E-01  1.0000000000
*
cl def-SVP
# cl    (7s5p) / [6s2p]     {211111/41}
*
   5  s
  10449.827566      .19708362484E-02
  1571.7365221      .14754727977E-01
  357.12065523      .66679112875E-01
  100.25185935      .17228924084
  30.812727554      .15883786100
   3  s
  51.923789434     -.10009298909
  5.7045760975      .60841752753
  2.3508376809      .54352153355
   1  s
  .44605124672      1.0000000000
   1  s
```

```
  .16848856190      1.0000000000
   5  p
  307.66790569     -.87801484118E-02
  72.102015515     -.63563355471E-01
  22.532680262     -.24016428276
  7.8991765444     -.47798866557
  2.8767268321     -.38515850005
   1  p
  .77459363955      1.0000000000
   1  p
  .21037699698      1.0000000000
   1  d
  .65000000000      1.0000000000
*
$ecp
*
ta def-ecp
*
  ncore =    60           lmax =     3
#        coefficient   r^n          exponent
f
         12.0179609    2           2.0178811
s-f
       1345.8806470    2          14.5464077
         36.7668062    2           7.2732038
        -12.0179609    2           2.0178811
p-f
        378.4253015    2           9.9355653
         22.2930909    2           4.9677824
        -12.0179609    2           2.0178811
d-f
        104.8839557    2           6.3473769
          8.7558481    2           3.1736885
        -12.0179609    2           2.0178811
*
$end
```

4.5.4 File *auxbasis*

```
$jbas
*
ta def-SVP
*
    3  s
     15.521335           -.493702989D+00
```

```
     7.555743             .259256574D+01
     3.699576            -.523168657D+01
    1  s
     1.820141             .262393615D+01
    1  s
     0.898838             .157711902D+01
    1  s
     0.445062             .200789711D+00
    1  s
     0.220729             .185974307D+00
    1  s
     0.109530             .765184411D-01
    1  p
     1.5024958           1.0
    1  p
     0.5629855           1.0
    1  p
     0.2281880           1.0
    1  p
     0.09507835          1.0
    2  d
     1.337006             .190072032D-01
     0.599535            -.155214344D-01
    1  d
     0.280427            -.138946250D-01
    1  d
     0.133078            -.895263676D-02
    1  f
     1.1428211           1.0
    1  f
     0.4395465           1.0
    1  f
     0.1758186           1.0
    3  g
     1.630421             .100251139D+00
     0.747093             .737448223D-01
     0.349040             .276219913D-01
    1  g
     0.164143             .546316580D-02
*
*
cl def-SVP
*
    8  s
     4097.080409            .198054511D+01
     1203.083193            .530973450D+01
```

```
     386.280948               .132352655D+02
     135.337690               .107149960D+02
     51.567046               -.132565114D+01
     21.261034                .271180364D+01
     9.420135                 .754640511D+01
     4.445228                 .173603618D+01
    1   s
     2.209399                -.140197496D+01
    1   s
     1.141575                 .982719736D+00
    1   s
     0.604182                 .464178589D+00
    1   s
     0.322378                 .369336889D+00
    4   p
     51.8499902611            .359335506D-01
     17.5847835188           -.869599318D-01
     6.49227239618            .721211200D-01
     2.55889114714           -.634201864D-01
    1   p
     1.05118767781            .264152293D-01
    1   p
     .437994865757           -.197670692D-01
    4   d
     34.705550               -.548703710D-01
     10.704427               -.619019402D-02
     3.568067                 .337450480D-01
     1.249848                -.905232209D-01
    1   d
     0.445360                 .418680075D-01
    1   f
     1.1872146118            1.0000000
    1   g
     1.30000000              1.0000000
*
$end
```

Appendix A

Allowed Keywords

In this section we give an alphabetical list of all keywords which may appear in *control* (or other files, e. g. libraries or files referenced in *control*) along with a short description. If suboptions are described, they may appear in the same line as the keyword as well as in the next line (except where indicated otherwise). For most keywords it is possible to write the corresponding data to a separate file and and append the reference `file=...` to the keyword. For example,

```
$scfmo  file=mo-vector
```

directs all programs to read (and write) the SCF MOs from file *mo–vector*.
Most keywords are only briefly described in this list; for further advice see the *control* file chapter (section 4). The references to the sections of the chapter are given in the following way: '→ general keywords' and '→ system specifications' refer to the sections 4.2.1 and 4.2.2 respectively, whereas references of modules as '→ **Dscf**' refer to the sections that describe the keywords of the appropriate module. For example '→ **Dscf**' and '→ **Ridft**' refer to the same section: 'Keywords for Modules **Dscf** and **Ridft**' (4.2.3). `<integer>` means an integer value, `<real>` a real number `<string>` a string argument following a keyword. Possible suboptions are not indicated explicitly.

`$actual step <string>`

This data group means current step and is written to your *control* file by every program. After the program has finished properly, this data group will be removed. If a program finds this data group in your *control* file when it starts, it will display a warning to you. If the entry in the data group is from the same program, it will nevertheless work, otherwise it will stop.
→ general keywords, section 4.2.1

`$alpha shells`

Specification of the occupation of alpha UHF MOs.
→ system specifications, section 4.2.2

`$atoms`

Specification of atomic attributes. For each atom type one or more lines should follow which must start with the atomic symbol. After the atomic symbol, the following suboptions may appear:

`<list>`	List of atomic indices like `1,2-4`. If no set of atomic indices is given, all atoms with the specified atomic symbol will be taken.
`basis= <string>`	This subitem specifies the basis set for the atoms specified, for example `basis=c dz`. This option must be specified for all atoms. The basis set nickname given must appear in the data block `$basis`.
`charge= <real>`	With this option you may assign a special nuclear charge to your atom.
`ncore= <integer>`	Number of core electrons which are described by effective core potentials.
`mass= <real>`	With this option you can assign special masses to some atoms (this may be useful for calculations of harmonic frequencies).
`ecp= <string>`	This subitem specifies the ecp set for the atoms specified. The ecp set nickname given must appear in the data block `$basis`.

Lines to be continued in the next line must end with an backslash (\).
→ system specifications, section 4.2.2

`$basis`

Basis set definitions. For the syntax see input examples (e. g. section 4.3.3)
→ general keywords, section 4.2.1

`$beta shells`

Specification of the occupation of beta UHF MOs.
→ system specifications, section 4.2.2

`$closed shells`

Specification of closed shells and their occupation numbers (which may differ from two). Each line consists of the symmetry label followed by a list of orbitals indexed within this symmetry and by the occupation number for this set of orbitals in brackets. For example, the input for H_2O in C_{2v} symmetry would look like

```
$closed shells
a1 1-3     ( 2 )
b1 1       ( 2 )
b2 1       ( 2 )
```

For open shell calculations, a separate data block `$open shells` is necessary (see there).
→ system specifications, section 4.2.2

`$coord`

This keyword specifies the cartesian coordinates of your molecule. For the syntax see input examples (e. g. section 4.3.2)
→ general keywords, section 4.2.1

`$coordinateupdate`

Control variables which deal with the method for updating the coordinates.
→ **Relax**

`$current`

MD data group containing present position, velocity, time and timestep.
→ **Frog**

`$denconv`

Convergence criterion for the root mean square of the density matrix.
→ **Dscf**

`$dft`

Data group for DFT calculations.
→ **Dscf**

`$dipgrad`

This data block contains the dipole moment derivatives as computed by the module **Aoforce**.

`$dipole`

Dipole moment as computed by the module **Aoforce**.

`$drvopt`

Specification of tasks for **Grad** and **Aoforce**.
→ **Grad**

`$ecp`

Specification of effective core potentials. For the syntax see input examples (e. g. section 4.5.3) and basis set library.
→ general keywords, section 4.2.1

`$electronic polarizability`

Electronic contribution to electrostatic polarizability as computed by module **Aoforce**.

`$electrostatic field`

Specification of external electric fields.
→ **Dscf**

`$end`

This keyword must be the last one in every *control* file, i. e. commands following `$end` will be neglected.
→ general keywords, section 4.2.1

`$energy`
: Bookkeeping of calculated energies
→ general keywords, section 4.2.1

`$fldopt`
: Specification of options related to external electrostatic fields.
→ **Dscf**

`$forceapprox`
: Approximate force constant matrix as used for the quasi–Newton methods.
→ general keywords, section 4.2.1

`$forceinit`
: Specifies initialization of the approximate force constant matrix.
→ **Relax**

`$forcepath`
: This keyword directs scratch files to a specified directory.
→ **Aoforce**

`$forceupdate`
: Method for the update of the approximate force constant matrix.
→ **Relax**

`$freeze`
: Specification of frozen core orbitals for an MP2–calculation.
→ **Mpgrad**

`$grad`
: Cartesian coordinates and cartesian gradients during all optimization cycles.
→ general keywords, section 4.2.1

`$grid`
: Data group for MO plots.
→ **Moloch**

`$hessian (projected)`
: Cartesian force constant matrix as computed by the module Aoforce.

`$intdef`
: Definitions of internal coordinates.
→ general keywords, section 4.2.1

`$incore <integer>`
: Specification of the maximum size of core memory which will be allocated for the two electron integrals.
→ **Dscf**

`$interconversion`
: Special options for the transformation of internal into cartesian coordinates

and vice versa.
→ **Relax**

`$jbas`
Auxiliary basis set definitions. For the syntax see input examples (e. g. section 4.5.4).
→ **Ridft**

`$kinetic_energy`
MD instruction requiring stack.
→ **Frog**

`$last MP2 energy change`
This data group is updated by program Mpgrad by taking the difference of the current total energy and the last entry in data group `$energy`.
→ **Jobex**

`$label`
MD instruction to change log filename.
→ **Frog**

`$last SCF energy change`
This data group is updated by program Dscf by taking the difference of the current total energy and the last entry in data group `$energy`.
→ **Jobex**

`$last step`
This data group is written by every program after it has finished properly (and after it has removed the data block `$actual step`).
→ general keywords, section 4.2.1

`$log`
Data group containing full record of MD trajectory.
→ **Frog**

`$maxcor`
This data group fixes the RAM memory to be used by the **Mpgrad** or **Aoforce** run.
→ **Mpgrad**, **Aoforce**

`$maximum norm of cartesian gradient`
Magnitude of the largest component of the cartesian gradient in the last optimization cycle.
→ **Jobex**

`$maximum norm of internal gradient`
Magnitude of the largest component of the internal gradient in the last optimization cycle.
→ **Jobex**

`$maxroot`

Termination criterion for module **Escf**.
→ **Escf**

`$md file=<string>`

Instruction in file *mdmaster* to look for MD commands elsewhere (in given filename).
→ **Frog**

`$mointunit`

Specification of disk space (MO– and half–transformed integrals).
→ **Mpgrad**

`$moments`

Compute electrostatic moments.
→ **Moloch**

`$mp2energy`

Calculate MP2–energy.
→ **Mpgrad**

`$mp2fcm`

MP2 gradient and MP2 dipole moment will be dumped to the files GRD and DIPOL respectively.
→ **Mpgrad**

`$mp2pair`

MP2 pair correlation energies will be calculated and printed.
→ **Mpgrad**

`$mulliken`

A Mulliken population analysis will be carried out.
→ **Moloch**

`$natoms`

Number of atoms for MD run.
→ **Frog**

`$natural orbitals`

Contains spin restricted natural orbitals calculated from UHF. (same syntax as `$scfmo`).
→ **Dscf**

`$natural orbital occupation`

Contains occupation numbers of natural orbitals.
→ general keywords, section 4.2.1

`$nsteps`

Number of MD steps still to be carried out.
→ **Frog**
→ **Jobex**

`$nuclear polarizability`
Nuclear contribution to electrostatic polarizability as computed by module **Aoforce**.

`$off-diagonal lagrange multipliers`
Dscf will write off–diagonal lagrange multipliers (open–shell calculations) to this data block.

`$open shells`
Specification of open shells and their occupation numbers. Same syntax as for `$closed shells`, see there.
→ **Dscf**

`$optimize`
Specification of the kind of non–linear parameters that are to be optimized by program **Relax**.
→ **Relax**

`$paboon`
A population analysis based on occupation numbers will be carried out.
→ **Moloch**

`$point_charges`
Specification of location and size of point charges to be included in a Hamiltonian.
→ **Dscf**

`$points`
Specification of points where electrostatic potential and related properties are computed.
→ **Moloch**

`$pople`
This data group specifies the number of cartesian components of basis functions (i.e. 5d and 7f in AO–Basis, 6d and 10f in CAO–Basis) for which the SCF calculation should be performed.
→ system specifications, section 4.2.2

`$position`
MD instruction to input new coordinates
→ **Frog**

`$properties`
Specification of properties computed by module **Moloch**.
→ **Moloch**

`$raman spectrum`
Scattering cross sections of Raman spectrum.
→ **Aoforce**

`$ricore <integer>`
Core memory given for special arrays in RI calculations of **Ridft** and **Rdgrad**.
→ **Ridft**

`$ridft`
General keyword for DFT calculations using the fast Coulomb approximation (RI) in **Ridft** and **Rdgrad**.
→ **Ridft**

`$ripop`
Calculation of atomic charges and atomic dipole moments based on the RI approximation.
→ **Ridft**

`$roothaan`
Parameters for a restricted–open–shell SCF–calculation.
→ **Dscf**

`$rpaconv`
Convergence criterion for instability calculations.
→ **Escf**

`$rundimensions`
In order to support dynamical memory allocation used by many **Turbomole** modules this data group will be created during a **Define** session.
→ **Dscf**

`$scfconv <integer>`
Convergence criterion for the total SCF–energy will be $10^{-integer}$.
→ **Dscf**

`$scfdamp`
Damping parameter for **Dscf**–calculations.
→ **Dscf**

`$scfdenapproxl <integer>`
The density difference is formed from `<integer>` previous densities plus that for the last iteration.
→ **Dscf**

`$scfdiis`
Parameters for a DIIS acceleration of SCF convergence.
→ **Dscf**

`$scfdump`
This option causes the MOs to be saved each iteration in formatted form on file *mos*.
→ **Dscf**

`$scfinstab`

Enables analysis of HF instabilities.
→ **Escf**

`$scfintunit`

Disc space specification for 2e–integrals.
→ **Dscf**

`$scfiterlimit <integer>`

Maximum number of SCF–iterations.
→ **Dscf**

`$scfmo`

Data group for the specification of SCF orbitals.
→ general keywords, section 4.2.1

`$scforbitalshift`

This option regulates the shifts of orbitals during the SCF–calculation.
→ **Dscf**

`$scratchpath`

This keyword directs scratch files to a specified directory.
→ **Aoforce**

`$seed <integer>`

MD: random number seed
→ **Frog**

`$shared electron numbers`

Specification of the shared electron numbers to be computed.
→ **Moloch**

`$soes`

This data group specifies the irreducible representation of the excited states to be calculated.
→ **Escf**

`$stack file=<string>`

MD energy statistics stack
→ **Frog**

`$statistics`

Several possible suboptions are available:

- `off` Don't calculate disk space requirements.
- `dscf` **Dscf** will calculate its disk space requirements according to `$thime` and `$thize` when called next.
- `mpgrad` **Mpgrad** will calculate its disk space requirements when called next.

After the integral statistics has been performed, the `$statistics` flag will be switched off and the correct file–sizes will be written to *control*.
→ general keywords, section 4.2.1

`$symmetry`
Schoenfliess symbol, e. g. `td`, `ih`, `c2v`.
→ system specifications, section 4.2.2

`$temperature`
MD instruction requiring stack
→ **Frog**

`$thime <integer>`
Integral storage threshold related to integral cost (computational effort versus storage requirements).
→ **Dscf**

`$thize <real>`
Integral storage threshold related to integral size.
→ **Dscf**

`$title`
This keyword provides a title for your project. The title follows in the next line.
→ system specifications, section 4.2.2

`$total_energy`
MD instruction requiring stack
→ **Frog**

`$traloop`
Number of loops over occupied orbitals performed in the semi–direct MP2 run.
→ **Mpgrad**

`$turbomole file=<string>`
MD command telling MD program where to look for `$grad` etc.
→ **Frog**

`$uhf`
General keyword for UHF (Unrestricted Hartree–Fock) handling in Dscf and Grad.
→ system specifications, section 4.2.2

`$uhfmo_alpha`
This data group contains the UHF MO vector for alpha spin.
→ general keywords, section 4.2.1

`$uhfmo_beta`
This data group contains the UHF MO vector for beta spin.
→ general keywords, section 4.2.1

`$velocity`

MD instruction to input new velocity
→ **Frog**

`$vibrational normal modes`

This data block contains the vibrational normal modes in cartesian coordinates as computed by the module **Aoforce**.

`$vibrational spectrum`

This data block contains harmonic frequencies and intensities of the normal modes as computed by the module **Aoforce**.

`$xdb`

This option supresses all system calls in the **Turbomole** programs (which are used only to provide the date and the time), because sometimes these calls don't work if the source code was compiled using an option for a symbolic debugger.