Manual for `tools-for-g16.bash` (0.2.0_alpha2, 2019-03-08)

Martin C Schwarzer

March 8, 2019

Preamble

The following document is currently little more than the already existing cheat sheet. As documentation often goes, it takes a bit more time to do it properly. I apologise for the inconvenience.

1 Introduction

The essence of quantum chemistry is performing calculations with a variety of different programs. Quite a few of these use low level input and output systems. In principle you are dealing with quite a few text-based files.

It is in the nature of the problem, that some of the regular tasks are quite repetitive. In order to ease these repetitive tasks, but more importantly also secure a certain level of consistency, some steps can be automated. This is where this repository, tools-for-g16.bash, comes into play. Within it are contained a few scripts written in bash, which are targeted to aid the use of the quantum chemistry software package Gaussian 16. They cover automatic generation of input files, submissing of these to a queueing system, and utilities to archive and extract results.

Please understand that this project started out to help me with my everyday work. I am happy to hear about suggestions and bugs. I am also fairly certain, that it is and will be a work in progress for quite some time. Be aware that it could be in constant flux.

This 'software' comes with absolutely no warrenty. None. Nada. If you decide to use any of the scripts, it is entirely your resonsibility.

1.1 Preliminary notes on the usage of this manual

To make the manual easier to understand a few abreviations and common notations are used: Anything set in brackets `[ ]` indicates optional arguments or inputs. Arguments in angles `< >` require human input, and a vertical bar `|` indicates alternatives between the inputs or options.

The following abbreviations will be used throughout:

`opt`	Short for option(s)
`ARG`	String type argument
`INT`	Positive integer (including zero)
`NUM`	Signed whole number (including zero)
`FLT`	Floating point number
`DUR`	Duration in format `[[HH:]MM:]SS`

2 Installation & Configuration

The scripts of this repository are not self-contained, they each need access to the resources directory and it has to be called like that.

The easiest way to install the script is to clone the git repository from github.com/polyluxus/tools-for-g16.bash. Alternatively you can download the tarball archive of the latest release from github.com/polyluxus/tools-for-g16.bash/releases/latest.

After unpacking it only needs to be configured. There are a two kinds of file names recognised: `g16.tools.rc` and the hidden `.g16.toolsrc`. The repository comes with an example of the former, therefore updating the repository will (probably) overwrite the file. The latter file is excluded from this process, and therefore has always precedence, so it is generally a safer option to configure.

The scripts will search for these configuration settings in the following order of directories:

1. the path to the script itself
2. the user's home directory

3. the `.config` directory in the user's home directory

4. the current working directory, i.e. from where the script is called.

Only the last found file will be applied.

The repository comes with a configuration script in the `configure` directory. It produces a formatted file like `g16.tools.rc` from old or the default settings. While you can assign an arbitrary filename, I recommend to store as `.g16.toolsrc` in the root directory of the repository.

To make the scripts accessible from anywhere, the directory where they have been stored must be found in the `PATH` variable. Alternatively, you can create softlinks to them in a directory, which is already recognised by `PATH`. A common setting for this is the local `$HOME/bin`. The configure script will, if desired, try to create these softlinks (omitting the `sh` suffix).

3 Utilities

This repository comes with the following scripts and files:

3.1 Input preparation: `g16.prepare.sh`

This tool reads in a file containing a set of cartesian coordinates and writes a Gaussian inputfile with predefined keywords. The script interfaces to Xmol format, Turbomole/ GFN-xTB `coord` format, too.

Usage:

`g16.prepare.sh [opt] <file>`

Options:

`-T <FLT>`	Temperature (kelvin)
`-P <FLT>`	Pressure (atmosphere)
`-r <ARG>`	Add `ARG` to route section
`-R <ARG>`	Specific route section `ARG`
`-l <INT>`	Load predefined route section
`-l list`	Show all predefined route sections
`-t <ARG>`	Adds `ARG` to end of file
`-C <ARG>`	Specify caption/title of job; Replacements: `%F` input filename; `%f` input filename without `.xyz`; `%s` like `%f`, also filtering `start`; `%j` jobname; `%c` charge (with indicator `chrg`); `%M` multiplicity (with indicator `mult`); `%U` unpaired electrons (with indicator `uhf`)
`-j <ARG>`	Jobname (derives filename of generated input; default: `<file>`)
`-j %f`	Jobname is `<file>` filtering `.xyz`
`-j %s`	Jobname is `<file>` filtering `start.xyz`
`-f <ARG>`	Filename of generated input
`-c <NUM>`	Charge (default: 0)
`-M <INT>`	Multiplicity (default: 1; ≥ 1)
`-U <INT>`	Unpaired electrons (unset; ≥ 0)
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.2 Pre-processing: `g16.testroute.sh`

This tool parses a Gaussian 16 inputfile and tests the route section for syntax errors with the Gaussian 16 utility `testrt`.

Usage:

`g16.testroute.sh [opt] <file>`

Options:

`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.3 Input preparation: g16.dissolve.sh

This tool reads in a Gaussian 16 inputfile (of a preferably completed calculation) and adds relevant keywords for solvent corrections. (Utilises the `%OldChk` directive and the `geom/ guess` keywords.)

Usage:

```
g16.dissolve.sh [opt] <file>
```

Options:

`-o <ARG>`	Adds option `ARG` to the `SCRF` keyword.
`-S <ARG>`	Adds option `solvent=ARG` to the `SCRF` option list.
`-O`	Runs an optimisation (preserves or adds `OPT`)
`-r <ARG>`	Add `ARG` to route section
`-t <ARG>`	Adds `ARG` to end of file
`-f <ARG>`	Filename of generated input
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.4 Input preparation: g16.freqinput.sh

This tool reads in a Gaussian 16 inputfile (of a preferably completed calculation) and adds relevant keywords for a frequency calculation. (Utilises the `%OldChk` directive and the `geom/ guess` keywords.)

Usage:

```
g16.freqinput.sh [opt] <file>
```

Options:

`-o <ARG>`	Adds option `ARG` to the `freq` keyword.
`-R`	Adds option `ReadFC` to the `freq` option list.
`-T <FLT>`	Temperature (kelvin)
`-P <FLT>`	Pressure (atmosphere)
`-r <ARG>`	Add `ARG` to route section
`-t <ARG>`	Adds `ARG` to end of file
`-f <ARG>`	Filename of generated input
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.5 Input preparation: g16.ircinput.sh

This tool reads in a Gaussian 16 inputfile from a (previously completed) frequency run and adds relevant keywords for two separate irc calculations. (Utilises the `%OldChk` directive and the `geom/ guess` keywords.)

Usage:

```
g16.ircinput.sh [opt] <file>
```

Options:

`-o <ARG>`	Adds option `ARG` to the `irc` keyword.
`-r <ARG>`	Add `ARG` to route section
`-t <ARG>`	Adds `ARG` to end of file
`-f <ARG>`	Filenametemplate of generated input files; If `<ARG>=jobname.suffix` then it produces `jobname.fwd.suffix` and `jobname.rev.suffix`
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.6 Input preparation: g16.optinput.sh

This tool reads in a Gaussian 16 inputfile preferably from a (previously completed) IRC run and writes and inputfile for a subsequent structure optimisation. (Utilises the `%OldChk` directive and the `geom`/ `guess` keywords.)

Usage:

```
g16.optinput.sh [opt] <file>
```

Options:

`-o <ARG>`	Adds option `ARG` to the `opt` keyword.
`-r <ARG>`	Add `ARG` to route section
`-t <ARG>`	Adds `ARG` to end of file
`-f <ARG>`	Filename of generated input
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.7 Input preparation: g16.spinput.sh

This tool reads in a Gaussian 16 inputfile and writes and inputfile for a subsequent calculation. It is possible to overwrite the existing route section, but still add the `geom`/`guess` directives to base it on. (Utilises the `%OldChk` directive.)

Usage:

```
g16.spinput.sh [opt] <file>
```

Options:

`-r <ARG>`	Add `ARG` to route section
`-R <ARG>`	Overwites route section with `ARG`
`-t <ARG>`	Adds `ARG` to end of file
`-f <ARG>`	Filename of generated input
`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.8 Input submission: g16.submit.sh

This tool parses and then submits a Gaussian 16 inputfile to a queueing system.

Usage:

```
g16.submit.sh [opt] <file>
```

Options:

`-m <INT>`	Memory (megabyte)
`-p <INT>`	Processors
`-d <INT>`	disksize via `MaxDisk` (megabyte)
`-w <DUR>`	Walltime limit
`-e <ARG>`	Specify an environment variable `ARG` in format `<VAR=value>`
`-j <INT>`	Wait for job with ID `INT`
`-H`	Submit with status hold (PBS, SLURM) or `PSUSP` (BSUB)
`-k`	Only create (keep) the jobscript, do not submit it.
`-Q <ARG>`	Queue for which job script should be created `<queue>-<special>` (`<queue>`: `pbs`, `slurm`, `bsub`; `<special>`: `gen` [generic], `rwth`)
`-P <ARG>`	Account to project (BSUB) or account (SLURM); if `ARG` is `default|0|''` presets are overwritten.
`-M <ARG>`	Specify a machine type (only `bsub-rwth`); if `ARG` is `default|0|''` presets are overwritten.
`-u <ARG>`	set user email address (SLURM, BSUB); if `ARG` is `default|0|''` presets are overwritten.
`--`	Close reading options
`-s`	Silence script (incremental)
`-h`	Help file

3.9 Post-processing: g16.getenergy.sh

This tool finds energy statements from Gaussian 16 calculations.

Usage:

```
g16.getenergy.sh [opt] [<file(s)>]
```

If no files given, it finds energy statements from all log files in the current directory.

Options:

-i <ARG>	Specify input suffix if processing directory
-o <ARG>	Specify output suffix if processing directory
-L	Print the full file and path name (seperated by newline)
--	Close reading options
-s	Silence script (incremental)
-h	Help file

3.10 Post-processing: g16.getfreq.sh

This tool summarises a frequency calculation and extracts the thermochemistry data.

Usage:

```
g16.getfreq.sh [opt] <file(s)>
```

Options:

-v	Incrementally increase verbosity
-V <INT>	Set level of verbosity directly, (0|1|2|3|4)
-c	Separate values by comma (-V0|-V1)
-f <ARG>	Write summary to file instead of screen
--	Close reading options
-s	Silence script (incremental)
-h	Help file

3.11 Post-processing: g16.chk2xyz.sh

A tool to convert a checkpoint file to an `xyz` file. This formats the `chk` first to a `fchk`.

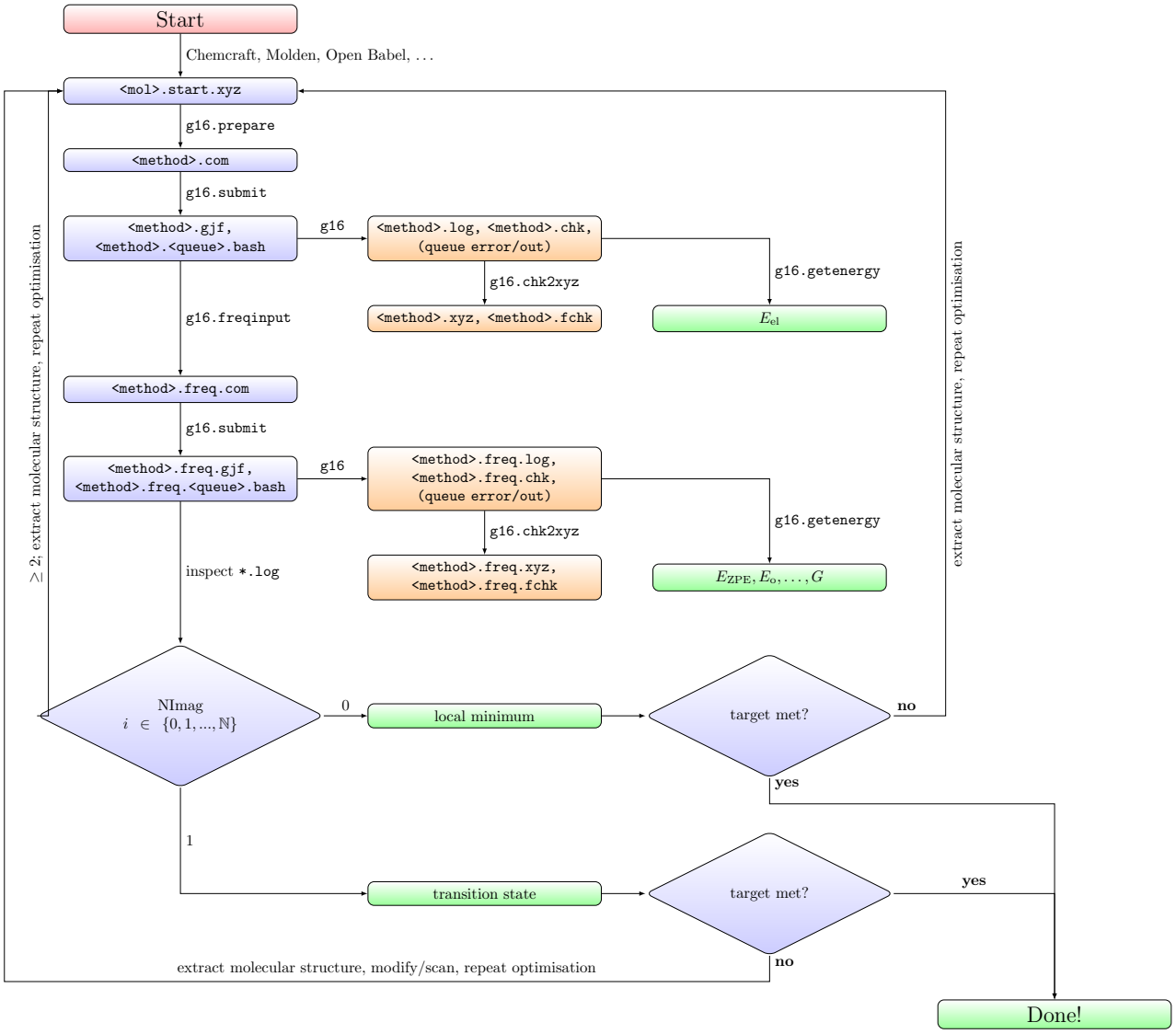
Usage:

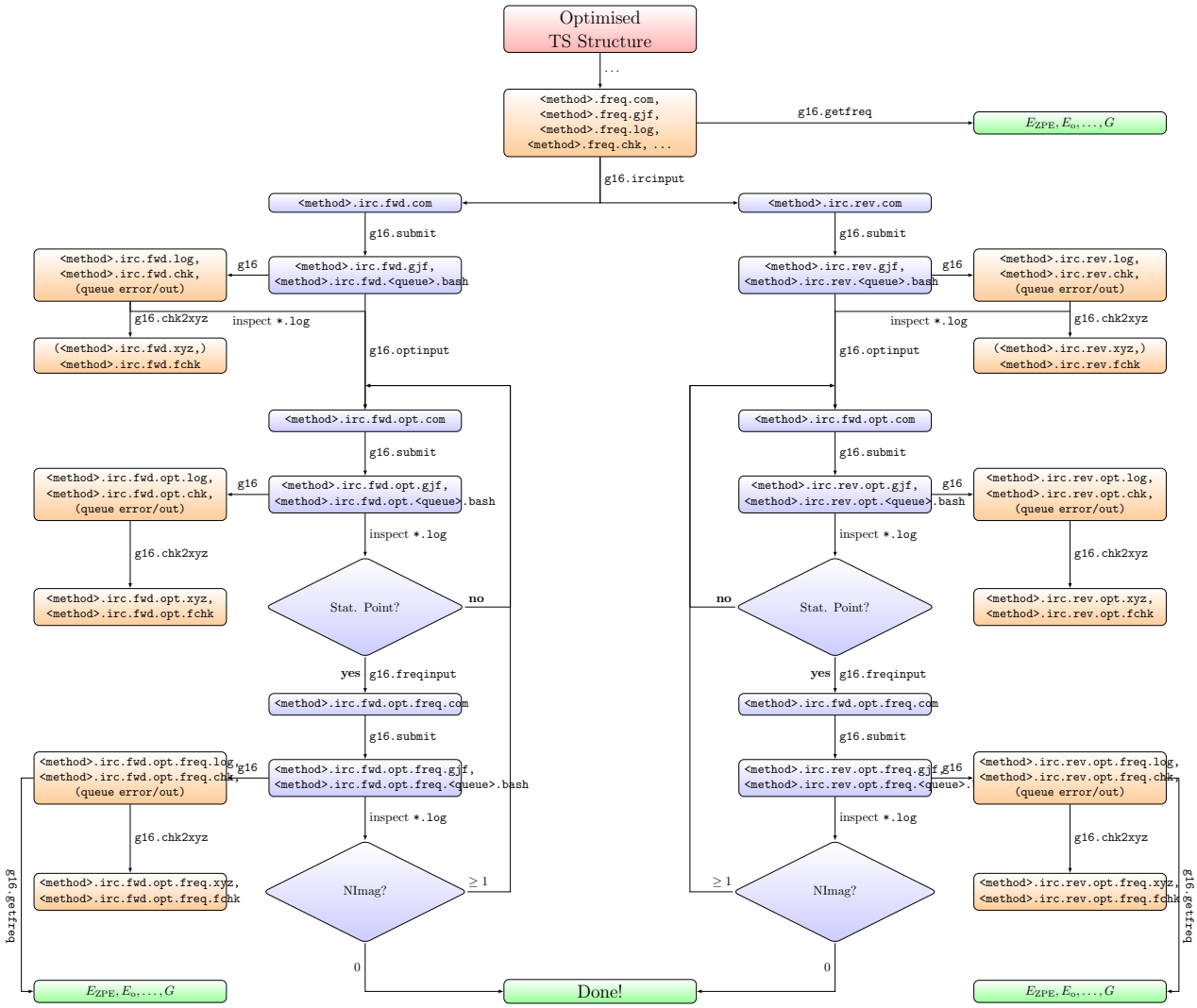
```
g16.chk2xyz.sh [-s] -h | -a | <chk-file(s)>
```

Options:

-a	Formats all checkpointfiles that are found in the current directory
--	Close reading options
-s	Silence script (incremental)
-h	Help file

4 Usage examples





5 Author, Bugs, and the Rest

Martin C Schwarzer
Martin-マーチン polyluxus