Application software interface for Open Calphad

Bo Sundman, March 17, 2016

This interface will be completely revised in the next release.

This is a preliminary description of the subroutines in the OC application software interface for use in software that need thermodynamic data.

The software is written in the new Fortran standard and there is an additional isoC version for use by C++ programs.

The opencalphad software is free with a GNU GPL license and is available at [ocweb] or as a development version at [github].

Documentation updating software

One very difficult thing with software that is alive and changing is to update the documentation. So my feeling for documentation is that once a software is documented it is dead as it is almost impossible to update source code and documentation in parallel.

To handle this I have written a simple documentation update software that uses a basic documentation written as a LaTeX file together with the source code to extract verbatim sections.

The verbatim sections in the source code is enclosed by

```
!\begin{verbatim}
extract from source code
!\end{verbatim}
```

These can be inserted anywhere in the source code. Normally they enclose important sections like data structure definitions and declarations of subroutines and functions. The documentation software finds new or changed verbatim sections as well as verbatim sections that has disappeared and writes a new LaTeX documentation file which require editing to describe the changes and to add cross-references. Using this software I hope to keep the software documented also during the development stage. This document is an example of this software.

Contents

1 Background

The use of accurate and consistent thermodynamic properties in various materials science software is very important. A multicomponent database constructed with the Calphad method to assess all available theoretical and experimental thermodynamic and phase data for unaries, binaries and higher order system is a guarantee to use consistent data but not always accurate. Great care must be taken to validate a database for a particular application as the composition and temperature ranges of accuracy can vary.

Application software for simulation of phase transformation or micro-structure evolution need thermodynamic data as well as kinetic and both of these can be stored in the same database format. Additional data for surface properties, strain/stress etc is often less available but can be integrated also.

The TQ software interface is based on the proposal for a generic thermodynamic software interface, called TQ, from a proposal by G Eriksson et al [95Eri]. It was adopted to the Fortran 77 standard with 6 character limit for subroutine names. In the revised application interface this limit will be removed. The TQ interface is used in slightly different forms in Thermo-Calc and ChemApp software.

2 Data structures

The OC software has a "static" data structure for the elements and phase parameters and a "dynamic" one for each equilibrium with the conditions and calculated results for all phases. One may have several equilibria and each is located with the pointer called "ceq" in most of the subroutines below.

The most important data structures for the application software are defined in the GTP package, the gtp_phasetuples to identify a phase and the phase_varres to store variables and calculated results for a phase.

2.1 The phase tuple

Each phase has a phase tuple index but as there can be several composition sets (due to miscibility gaps) the number of tuples may be larger than the number of phases and change after each calculation.

There is an array PHASETUPLE with all phase tuple records that can be accessed by the application software. **NEVER CHANGE ANY VALUE IN THIS ARRAY**.

```
  TYPE gtp_phasetuple
! for handling a single array with phases and composition sets
```

```
! first index is phase index, second index is composition set
! ADDED also index in phlista (ixphase) and phase_varres (lokvares) and
! nextcs which is nonzero if there is a higher composition set of the phase
! A tuplet index always refer to the same phase+compset.  New tuples with
! the same phase and other compsets are added at the end.
     integer phaseix,compset,ixphase,lokvares,nextcs
  end TYPE gtp_phasetuple
```

2.2 The phase_varres record

The phase_varres record contains most of the information for the phase that is important for the application. It is convenient to access these directly using the "ceq" pointer and the %lokvares index in the phase tuple, for example the Gibbs energy of the phase ptupx per mole formula unit is:

ceq%phase_varres(phasetuple(ptupx%lokvares)%gval(1,1)

NEVER CHANGE ANY VALUE IN THE PHASE_VARRES RECORD. That may cause severe errors. For example the constitution can be changed by the subroutine tqsphc1, but never by changing the yfr in the record because many other values must also be updated inside the ceq record.

```
  TYPE gtp_phase_varres
! Data here must be different in equilibria representing different experiments
! or calculated in parallel or results saved from step or map.
! nextfree: In unused phase_varres record it is the index to next free record
!    The global integer csfree is the index of the first free record
!    The global integer highcs is the higest varres index used
! phlink: is index of phase record for this phase_varres record
! status2: has phase status bits like ENT/FIX/SUS/DORM
! phstate: indicate state: fix/stable/entered/unknown/dormant/suspended/hidden
!                          2   1      0        -1      -2      -3       -4
! phtupx: phase tuple index
     integer nextfree,phlink,status2,phstate,phtupx
! abnorm(1): amount moles of atoms for a formula unit of the composition set
! abnorm(2): mass/formula unit (both set by call to set_constitution)
! prefix and suffix are added to the name for composition sets 2 and higher
     double precision, dimension(2) :: abnorm
     character*4 prefix,suffix
! constat: array with status word for each constituent, any can be suspended
! yfr: the site fraction array
! mmyfr: min/max fractions, negative is a minumum
! sites: site ratios (which can vary for ionic liquids)
     integer, dimension(:), allocatable :: constat
     double precision, dimension(:), allocatable :: yfr
```

```
     real, dimension(:), allocatable :: mmyfr
     double precision, dimension(:), allocatable :: sites
! for ionic liquid derivatives of sites wrt fractions (it is the charge),
! 2nd derivates only when one constituent is vacancy
! 1st sublattice P=\sum_j (-v_j)*y_j + Qy_Va
! 2nd sublattice Q=\sum_i v_i*y_i
! dpqdy is the abs(valency) of the species, set in set_constitution
! for the vacancy it is the same as the number of sites on second subl.
! used in the minimizer and maybe elsewhere
     double precision, dimension(:), allocatable :: dpqdy
     double precision, dimension(:), allocatable :: d2pqdvay
! disfra: a structure describing the disordered fraction set (if any)
! for extra fraction sets, better to go via phase record index above
! this TYPE(gtp_fraction_set) variable is a bit messy.  Declaring it in this
! way means the record is stored inside this record.
     type(gtp_fraction_set) :: disfra
! ---
! arrays for storing calculated results for each phase (composition set)
! amfu: is amount formula units of the composition set (calculated result)
! netcharge: is net charge of phase
! dgm: driving force
     double precision amfu,netcharge,dgm
! Other properties may be that: gval(*,2) is TC, (*,3) is BMAG, see listprop
! nprop: the number of different properties (set in allocate)
! listprop(1): is number of calculated properties
! listprop(2:listprop(1)): identifies the property stored in gval(1,ipy) etc
!   2=TC, 3=BMAG. Properties defined in the gtp_propid record
     integer nprop
     integer, dimension(:), allocatable :: listprop
! gval etc are for all composition dependent properties, gval(*,1) for G
! gval(*,1): is G, G.T, G.P, G.T.T, G.T.P and G.P.P
! dgval(1,j,1): is first derivatives of G wrt fractions j
! dgval(2,j,1): is second derivatives of G wrt fractions j and T
! dgval(3,j,1): is second derivatives of G wrt fractions j and P
! d2gval(ixsym(i,j),1): is second derivatives of G wrt fractions i and j
     double precision, dimension(:,:), allocatable :: gval
     double precision, dimension(:,:,:), allocatable :: dgval
     double precision, dimension(:,:), allocatable :: d2gval
! added for strain/stress, current values of lattice parameters
     double precision, dimension(3,3) :: curlat
! saved values from last equilibrium for dot derivative calculations
     double precision, dimension(:,:), allocatable :: cinvy
     double precision, dimension(:), allocatable :: cxmol
     double precision, dimension(:,:), allocatable :: cdxmol
  END TYPE gtp_phase_varres
! this record is created inside the gtp_equilibrium_data record
```

2.3 State variable symbols

It is also important to know and understand the syntax for state variables as given in table 1 at the end.

3 Examples of application programs

There are some simple examples of using the OCTQ interface provided with the software. See the TQ3-clean directory. There are examples both using Fortran and C++.

4 Subroutines and functions

The list here is in the order the subroutines appear in the source code. It is not always very logical. Some of them are not yet implemented.

Note there is a separate (as yet undocumented) library for C++ program calling the OC application interface.

4.1 Initiate OCTQ

This subroutine must be called before any of the other subroutines in this interface. It returns a pointer, "ceq" to the equilibrium data structure which is needed in many of the other subroutines.

Several equilibria can be created, see the tqcceq subroutine.

```
  subroutine tqini(n,ceq)
! initiate workspace
    implicit none
    integer n ! Not nused, could be used for some initial allocation
    type(gtp_equilibrium_data), pointer :: ceq ! EXIT: current equilibrium
```

4.2 Read database file

The first routine reads all elements and phases from a database with TDB format. The second reads all phases for a selected set of elements.

```
  subroutine tqrfil(filename,ceq)
! read all elements from a TDB file
    implicit none
    character*(*) filename  ! IN: database filename
    character ellista(10)*2  ! dummy
```

```
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
  subroutine tqrpfil(filename,nsel,selel,ceq)
! read TDB file with selection of elements
    implicit none
    character*(*) filename  ! IN: database filename
    integer nsel
    character selel(*)*2  ! IN: elements to be read from the database
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
```

4.3 Get names of components

```
  subroutine tqgcom(n,compnames,ceq)
! get system component names. At present the elements
    implicit none
    integer n                                ! EXIT: number of components
    character*24, dimension(*) :: compnames ! EXIT: names of components
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.4 Get number of phases and composition sets

After reading the database each phase has one composition set. But as several phases may have miscibility gaps new phase tuples may be created after a calculation. This routine should thus be called after each calculation when the grid minimizer is used.

```
  subroutine tqgnp(n,ceq)
! get total number of phases and composition sets
! A second composition set of a phase is normally placed after all other
! phases with one composition set
    implicit none
    integer n    !EXIT: n is number of phases
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
```

4.5 Get name of phase using index

Using the phase tuple index the phase name is returned. For phases with several composition sets it will have a suffix #digit.

```
  subroutine tqgpn(phcsx,phasename,ceq)
! get name of phase+compset tuple with index phcsx
    implicit none
    integer phcsx                   ! IN: index in phase tuple array
!    TYPE(gtp_phasetuple), pointer :: phcs  !IN: phase number and comp.set
    character phasename*(*)      !EXIT: phase name, max 24+8 for pre/suffix
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
```

4.6 Get index of phase using name

If the phase name has a suffix #digit the index of that composition set will be returned.

```
  subroutine tqgpi(phcsx,phasename,ceq)
! get index of phase phasename (including comp.set, ceq not needed ...
    implicit none
    integer phcsx           !EXIT: phase tuple index
    character phasename*(*) !IN: phase name
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
```

4.7 Get name of constituent using index

The constituents are numbered sequentially over all sublattices from the first to the last.

```
  subroutine tqgpcn2(n,c,constituentname,ceq)
! get name of consitutent c in phase n
! NOTE An identical routine with different constituent index is tqgpcn
    implicit none
    integer n !IN: phase number (not phase tuple)
    integer c !IN: constituent index sequentially over all sublattices
    character constituentname*(24) !EXIT: costituent name
    type(gtp_equilibrium_data), pointer :: ceq !IN: current equilibrium
```

4.8 Get index of constituent using name

The elements and components are usually ordered alphabetically.

```
  subroutine tqgpci(n,c,constituentname,ceq)
! get index of constituent with name in phase n
    implicit none
    integer n !IN: phase index
!NO  integer c !IN: extended constituent index: 10*species_number+sublattice
    integer c !IN: sequantial constituent index over all sublattices
    character constituentname*(*)
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.9 Get stoichiometry of constituent

For massbalance calculations the stoichiometry and mass of a constituent may be needed by the application software.

```
  subroutine tqgpcs(n,c,stoi,mass,ceq)
! get stoichiometry of constituent c in phase n
!? missing argument number of elements????
    implicit none
    integer n !IN: phase number
!NO  integer c !IN: extended constituent index: 10*species_number+sublattice
    integer c !IN: sequantial constituent index over all sublattices
    double precision stoi(*) !EXIT: stoichiometry of elements
    double precision mass    !EXIT: total mass
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.10 Get stoichiometry of component

In a future version of OC the use will be able to define other components than the elements.

```
  subroutine tqgccf(n1,n2,elnames,stoi,mass,ceq)
! get stoichiometry of component n1
! n2 is number of elements (dimension of elnames and stoi)
    implicit none
    integer n1 !IN: component number
    integer n2 !EXIT: number of elements in component
    character elnames(*)*(2) ! EXIT: element symbols
    double precision stoi(*) ! EXIT: element stoichiometry
    double precision mass    ! EXIT: component mass (sum of element mass)
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.11 Get number of constituents of a phase

The heading says all.

```
  subroutine tqgnpc(n,c,ceq)
! get number of constituents of phase n
    implicit none
    integer n !IN: Phase number
    integer c !EXIT: number of constituents
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.12 Set phase status

A phase can have 4 status: SUSPENDED, DORMANT, ENTERED and FIXED. The corresonding values of newstat is -3, -2, 0, and +2. The values -1 and +1 are used internally to indicate if the phase is stable or not.

A suspended phase (-3) is ignored at calculations but will appear in listings. A dormant phase (-2) can never be set as stable but it is included in the calculations and its composition is adjusted to calculate is least negative driving force. If the driving force is positive the phase would like to be stable.

The entered status (0) is the default, during an equilibrium calculation the phase will be set stable if that decreases the total Gibbs energy. In "val" the user should supply an estimate of the amount of the phase in moles, for example zero. This will be used as start value in an equilibrium calculation.

The fix status (+2) means it is a condition that the phase should be stable. The value of "val" is used in the calculation and if val is zero then one is at the limit of stability of the phase. For example the melting temperature can be calculated by setting the liquid phase as fixed with zero moles.

```
  subroutine tqphtupsts(tup,newstat,val,ceq)
! set status of phase tuple,
    integer tup,newstat
    double precision val
    type(gtp_equilibrium_data), pointer :: ceq  ! IN: current equilibrium
```

4.13 Set condition

This is the routine to set values of the external state variables which control the system and are necessary to calculate the equilibrium. Each condition is set separately and the user must set as many as required by the Gibbs phase rule, $n+2$, where n is the number of components.

The user must be careful using the correct state variables according to table 1. It is simplest to set conditions on T, P and amounts of all components. With any other set of conditions it is possible the equilibrium does not exist.

Note that setting a phase to the fix status is considered as a condition.

```
  subroutine tqsetc(stavar,n1,n2,value,cnum,ceq)
! set condition
! stavar is state variable as text
! n1 and n2 are auxilliary indices
! value is the value of the condition
! cnum is returned as an index of the condition.
! to remove a condition the value sould be equial to RNONE ????
! when a phase indesx is needed it should be 10*nph + ics
! SEE TQGETV for doucumentation of stavar etc.
    implicit none
    integer n1             ! IN: 0 or phase tuple index or component number
    integer n2             ! IN: 0 or component number
    integer cnum           ! EXIT: sequential number of this condition
    character stavar*(*)   ! IN: character with state variable symbol
```

```
    double precision value ! IN: value of condition
    type(gtp_equilibrium_data), pointer :: ceq  ! IN: current equilibrium
```

4.14 Calculate equilibrium

When the degrees of freedom are zero one can calculate the equilibrium with this subroutine.

Note that "target" is not used in the OC software. A special feature is that if "n1=-1" the grid minimizer will not be called. That is useful if the user already calculated an equilibrium using the grid minimizer and after that changed the values of the conditions only small amounts.

```
  subroutine tqce(target,n1,n2,value,ceq)
! calculate quilibrium with possible target
! Target can be empty or a state variable with indices n1 and n2
! value is the calculated value of target
    implicit none
    integer n1,n2,mode
    character target*(*)
    double precision value
    logical confirm
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

4.15 Get state variable value

For this routine the user must again use the table 1 in order to obtain the desired value. Note that negative values of n1 and n2 usually means "all", i.e. an array of value is returned, for example all mole fractions or the amount of all phases.

A frequent mistake is to think that the derivative of the Gibbs energy with respect to a constituent fraction is the chemical potential of that constituent. This is not the case and the chemical potential can only be obtained by the state variable MU using this subroutine.

```
  subroutine tqgetv(stavar,n1,n2,n3,values,ceq)
! get equilibrium results using state variables
! stavar is the state variable IN CAPITAL LETTERS with indices n1 and n2
! n1 can be a phase tuple index, n2 a component index
! n3 at the call is the dimension of the array values,
! changed to number of values on exit
! value is an array with the calculated value(s), n3 set to number of values.
    implicit none
    integer n1,n2,n3
    character stavar*(*)
    double precision values(*)
    type(gtp_equilibrium_data), pointer :: ceq  !IN: current equilibrium
```

```
!=========================================================
! stavar must be a symbol listed below
! IMPORTANT: some terms explained after the table
! Symbol  index1,index2                    Meaning (unit)
!.... potentials
! T     0,0                                              Temperature (K)
! P     0,0                                              Pressure (Pa)
! MU    component,0 or ext.phase.index*1,constituent*2  Chemical potential (J)
! AC    component,0 or ext.phase.index,constituent      Activity = EXP(MU/RT)
! LNAC  component,0 or ext.phase.index,constituent      LN(activity) = MU/RT
!...... extensive variables
! U     0,0 or ext.phase.index,0   Internal energy (J) whole system or phase
! UM    0,0 or ext.phase.index,0       same per mole components
! UW    0,0 or ext.phase.index,0       same per kg
! UV    0,0 or ext.phase.index,0       same per m3
! UF    ext.phase.index,0              same per formula unit of phase
! S*3   0,0 or ext.phase.index,0   Entropy (J/K)
! V     0,0 or ext.phase.index,0   Volume (m3)
! H     0,0 or ext.phase.index,0   Enthalpy (J)
! A     0,0 or ext.phase.index,0   Helmholtz energy (J)
! G     0,0 or ext.phase.index,0   Gibbs energy (J)
! ..... some extra state variables
! NP    ext.phase.index,0          Moles of phase
! BP    ext.phase.index,0          Mass of moles (kg)
! Q     ext.phase.index,0          Internal stability/RT (dimensionless)
! DG    ext.phase.index,0          Driving force/RT (dimensionless)
!....... amounts of components
! N     0,0 or component,0 or ext.phase.index,component    Moles of component
! X     component,0 or ext.phase.index,component   Mole fraction of component
! B     0,0 or component,0 or ext.phase.index,component     Mass of component
! W     component,0 or ext.phase.index,component   Mass fraction of component
! Y     ext.phase.index,constituent*1                    Constituent fraction
!........ some parameter identifiers
! TC    ext.phase.index,0               Magnetic ordering temperature
! BMAG  ext.phase.index,0               Aver. Bohr magneton number
! MQ&   ext.phase.index,constituent    Mobility
! THET  ext.phase.index,0               Debye temperature
! LNX   ext.phase.index,0               Lattice parameter
! EC11  ext.phase.index,0               Elastic constant C11
! EC12  ext.phase.index,0               Elastic constant C12
! EC44  ext.phase.index,0               Elastic constant C44
!........ NOTES:
! *1 The ext.phase.index is   10*phase_number+comp.set_number
! *2 The constituent index is 10*species_number + sublattice_number
! *3 S, V, H, A, G, NP, BP, N, B and DG can have suffixes M, W, V, F also
!---------------------------------------------------------------
```

```
! special addition for TQ interface: d2G/dyidyj
! D2G + phase tuple
!-------------------------------------------------------------------
```

4.16 Get phase constitution

This is redundant as the constitution can be obtained using the "ceq" record pointer and the phase_varres index in the phase tuple. For the phase tuple tupix:

ceq%phase_varres(phasetuple(tupix)%lokvares)%yfr

But in may anyway be useful to get also the complete information this way rather than digging in the phase_varres record.

```
  subroutine tqgphc1(n1,nsub,cinsub,spix,yfrac,sites,extra,ceq)
! tq_get_phase_constitution
! This subroutine returns the sublattices and constitution of a phase
! n1 is phase tuple index
! nsub is the number of sublattices (1 if no sublattices)
! cinsub is an array with the number of constituents in each sublattice
! spix is an array with the species index of the constituents in all sublattices
! sites is an array of the site ratios for all sublattices.
! yfrac is the constituent fractions in same order as in spix
! extra is an array with some extra values:
!    extra(1) is the number of moles of components per formula unit
!    extra(2) is the net charge of the phase
    implicit none
    integer n1,nsub,cinsub(*),spix(*)
    double precision sites(*),yfrac(*),extra(*)
    type(gtp_equilibrium_data), pointer :: ceq
```

4.17 Set phase constitution

This allows the user to set the constitution of a phase.

NEVER CHANGE THE CONSTITUTION BY SETTING VALUES IN THE YFR ARRY IN THE PHASE_VARRES RECORD. There are many additional variables that must be updated when a new constitution is set.

```
  subroutine tqsphc1(n1,yfra,extra,ceq)
! tq_set_phase_constitution
! To set the constitution of a phase
! n1 is phase tuple index
! yfra is an array with the constituent fractions in all sublattices
! in the same order as obtained by tqgphc1
```

```
! extra is an array with returned values with the same meaning as in tqgphc1
! NOTE The constituents fractions are normallized to sum to unity for each
!      sublattice and extra is calculated by tqsphc1
! T and P must be set as conditions.
    implicit none
    integer n1
    double precision yfra(*),extra(*)
    type(gtp_equilibrium_data), pointer :: ceq
```

4.18 Calculate phase properties

The first routine returns the values in arrays, the second just the index to the ceq record. With the new phase tuple record these routines are more or less redundant.

```
  subroutine tqcph1(n1,n2,n3,gtp,dgdy,d2gdydt,d2gdydp,d2gdy2,ceq)
! tq_calculate_phase_properties
!vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
! WARNING: this is not a subroutine to calculate chemical potentials
! those can only be made by an equilibrium calculation.
! The values returned are partial derivatives of G for the phase at the
! current T, P and phase constitution.  The phase constitution has been
! obtained by a previous equilibrium calculation or
! set by the subroutine tqsphc
! It corresponds to the "calculate phase" command.
!
! NOTE that values are per formula unit divided by RT,
! divide also by extra(1) in subroutine tqsphc1 to get them per mole component
!
!^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
! calculate G and some or all derivatives for a phase at current composition
! n1 is the phase tuple index
! n2 is 0 if only G and derivatves wrt T and P, 1 also first drivatives wrt
!    compositions, 2 if also 2nd derivatives
! n3 is returned as number of constituents (dimension of returned arrays)
! gtp is an array with G, G.T, G:P, G.T.T, G.T.P and G.P.P
! dgdy is an array with G.Yi
! d2gdydt is an array with G.T.Yi
! d2gdydp is an array with G.P.Yi
! d2gdy2 is an array with the upper triangle of the symmetrix matrix G.Yi.Yj
! reurned in the order:  1,1; 1,2; 1,3; ...
!                             2,2; 2,3; ...
!                                  3,3; ...
! for indexing one can use the integer function ixsym(i1,i2)
    implicit none
    integer n1,n2,n3
```

```
    double precision gtp(6),dgdy(*),d2gdydt(*),d2gdydp(*),d2gdy2(*)
    type(gtp_equilibrium_data), pointer :: ceq
  subroutine tqcph2(n1,n2,n3,n4,ceq)
! tq_calculate_phase_properties
!vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
! WARNIG: this is not a subroutine to calculate chemical potentials
! those can only be made by an equilibrium calculation.
! The values returned are partial derivatives of G for the phase at the
! current T, P and phase constitution.  The phase constitution has been
! obtained by a previous equilibrium calculation or
! set by the subroutine tqsphc
! It corresponds to the "calculate phase" command.
!
! NOTE that values are per formula unit divided by RT,
! divide also by extra(1) in subroutine tqsphc1 to get them per mole component
!
!^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
! calculate G and some or all derivatives for a phase at current composition
! n1 is the phase tuple index
! n2 is type of calculation (0, 1 or 2)
! n3 is returned as number of constituents
! n4 is index to ceq%phase_varres(lokres)% with all results
! for indexing one can use the integer function ixsym(i1,i2)
    implicit none
    integer n1,n2,n3,n4
    type(gtp_equilibrium_data), pointer :: ceq
```

4.19 Delete equilibrium

If the user have created several equilibria he can also delete them. The first equilibrium returned by tqini cannot be deleted.

```
  subroutine tqdceq(name)
! delete equilibrium with name
    implicit none
    character name*24
!    integer n1
    type(gtp_equilibrium_data), pointer :: newceq,ceq
```

4.20 Copy equilibrium

This routine creates a new equilibrium record and returns a pointer in “neweq”. The routine copies the current values in the the “ceq” record as initial data, including conditions and results. The user can then change these independently from any other equilibrium.

```
  subroutine tqcceq(name,n1,newceq,ceq)
! copy_current_equilibrium to newceq
! creates a new equilibrium record with name with values same as ceq
! n1 is returned as index
    implicit none
    character name*24
    integer n1
    type(gtp_equilibrium_data), pointer :: newceq,ceq
```

4.21 Select equilibrium

The user can select the equilibrium for the following calls by changing the pointer "ceq". It may be easier to maintain pointers in his own software.

```
  subroutine tqselceq(name,ceq)
! select current equilibrium to be that with name.
! Note that equilibria can be deleted and change number but not name
    implicit none
    character name
    type(gtp_equilibrium_data), pointer :: ceq
```

5 Subroutines in alphabetical order

This is all subroutines in alphabetical order. As the programmer has access to the source code he may also call directly subroutines in the different packages but these subroutines may be changed without notice in a future release. To avoid such problems please contact the development team for proposals of new routines in the software interface.

Please note also that results from a calculation can be directly accessed in the phase_varres record for each phase in the equilibrium record and the phase tuple record has the index to this record for each phase.

tqcceq	Create and copy equilibrium
tqce	Calculate equilibrium
tqcph1	Calculate phase properties
tqcph2	Calculate phase properties
tqdceq	Delete equilibrium
tqgccf	Get stoichiometry of component
tqgcom	Get the component names
tqgetv	Get state variable value
tqgnp	Get number of phases and composition sets
tqgnpc	Get number of constituents in phase
tqgpci	Get index of phase constituent using name
tqgpcn2	Get phase constituent index using name
tqgpcs	Get stoichiometry of constituent
tqgphc1	Get phase constitution
tqgpi	Get index of phase using name
tqgpn	Get name of phase using index
tqini	Initiate the TQ interface
tqphtupsts	Set phase status
tqrfil	Read a complete database
tqrpfil	Read selected elements from a database
tqselceq	Select equilibrium
tqsetc	Set condition
tqsphc1	Set phase constitution

6 Summary

That is all!

References

[95Eri] G Eriksson, P Spenser and H Sippola, 2nd Colloquium on Process Simulations, pp 115-126, June 1995, HUT, Espoo, Finland.

[07Luk] H L Lukas, S G Fries and B Sundman, *Computational Thermodynamics, the Calphad method*, Cambridge univ press (2007)

[ocweb] http://www.opencalphad.org

[github] the opencalphad repository at http://www.gitbub.com/sundmanbo/opencalphad

[15Sun1] B Sundman, U Kattner, M Palumbo and S G Fries, OpenCalphad - a free thermodynamic software, Integrating Materials and Manufacturing Innovation, **4**:1 (2015), open access

Table 1: A very preliminary table with the state variables and their internal representation. Some model parameter properties are also included. Note the letter z used in some symbols like Sz means the optional normalizing, i.e. M, W, V or F.

Symbol	Id	Index		Normalizing	Meaning
		1	2	suffix	
Intensive properties					
T	1	-	-	-	Temperature
P	2	-	-	-	Pressure
MU	3	component	-/phase	-	Chemical potential
AC	4	component	-/phase	-	Activity
LNAC	5	component	-/phase	-	LN(activity)
Extensive properties					
U	10	-/phase#set	-	-	Internal energy for system
UM	11	-/phase#set	-	M	Internal energy per mole
UW	12	-/phase#set	-	W	Internal energy per mass
UV	13	-/phase#set	-	V	Internal energy per m^3
UF	14	phase#set	-	F	Internal energy per formula unit
Sz	2z	-/phase#set	-	-	entropy
Vz	3z	-/phase#set	-	-	volume
Hz	4z	-/phase#set	-	-	enthalpy
Az	5z	-/phase#set	-	-	Helmholtz energy
Gz	6z	-/phase#set	-	-	Gibbs energy
NPz	7z	phase#set	-	-	Moles of phase
BPz	8z	phase#set	-	-	Mass of phase
Qz	9z	phase#set	-	-	Stability of phase
DGz	10z	phase#set	-	-	Driving force of phase
Nz	11z	-/phase#set/comp	-/comp	-	Moles of component
X	111	phase#set/comp	-/comp	0	Mole fraction
X%	111	phase#set/comp	-/comp	100	Mole per cent
Bz	12z	-/phase#set/comp	-/comp	-	Mass of component
W	122	phase#set/comp	-/comp	0	Mass fraction
W%	122	phase#set/comp	-/comp	100	Mass per cent
Y	130	phase#set	const#subl	-	Constituent fraction
Some model parameter identifiers					
TC	-	phase#set	-	-	Curie temperature
BMAG	-	phase#set	-	-	Aver. Bohr magneton number
MQ&X	-	phase#set	constituent	-	Mobility of X
THET	-	phase#set	-	-	Debye temperature