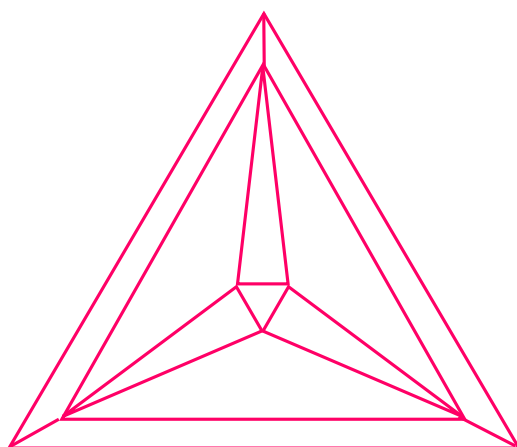


Thermodynamic Calculation Interface

TQ-Interface

(Version 7.0)

Programmer's Guide and Examples



*Thermo-Calc Software AB
Stockholm Technology Park, Björnåsvägen 21
SE-113 47 Stockholm, Sweden*

May, 2008

Thermodynamic Calculation Interface

TQ-Interface

(Version 7.0)

Programmer's Guide and Examples

Developers: Bo Sundman and Qing Chen
Div of Computational Thermodynamics
Dept of Materials Science and Engineering
Royal Institute of Technology
SE - 100 44 Stockholm, Sweden

Supporter: Thermo-Calc Software AB
Stockholm Technology Park
Björnnäsvägen 21
SE - 113 47 Stockholm, Sweden

Copyright © **1995-2008** Foundation of Computational Thermodynamics
Stockholm, Sweden

Copyright:

The Thermo-Calc and DICTRA software are exclusive copyright properties of the STT Foundation (Foundation of Computational Thermodynamics, Stockholm, Sweden). All rights are reserved worldwide!

Thermo-Calc Software AB has exclusive rights for further developing and marketing all kinds of versions of Thermo-Calc and DICTRA software/database/interface packages, worldwide.

This *TQ-Interface Programmer's Guide and Examples*, as well as all other related documentation, is the copyright property of Thermo-Calc Software AB.

It is absolutely forbidden to make any illegal copies of the software, databases, interfaces, and their manuals (User's Guide and Examples Book) and other technical publications (Reference Book and Technical Information). Any unauthorized duplication of such copyrighted products, is a violation of international copyright law. Individuals or organizations (companies, research companies, governmental institutes, and universities) that make or permit to make unauthorized copies may be subject to prosecution.

The utilization of the Thermo-Calc and DICTRA software/database/interface packages and their manuals and other technical information are extensively and permanently governed by the *Thermo-Calc Software AB END USER LICENSE AGREEMENT (EULA)*, which is connected with the software.

Disclaimers:

Thermo-Calc Software AB and the STT Foundation reserve the rights to further developments of the Thermo-Calc and DICTRA software and related software/database/interface products, and to revisions of their manuals and other publications, with no obligation to notify any individual or organization of such developments and revisions. In no event shall Thermo-Calc Software AB and the STT Foundation be liable to any loss of profit or any other commercial damage, including but not limited to special, consequential or other damage.

Please visit the Thermo-Calc Software web site (www.thermocalc.se) for any modification and/or improvement that have been incorporated into the programs, or for any amendment that have made to the contents of the various User's Guides and to the FAQ lists and other technical information publications.

Acknowledgement of Copyright and Trademark Names:

Various third-party software names that are protected by copyright and/or trademarks are mentioned for descriptive purposes, within this User's Guide and other documents of the Thermo-Calc and DICTRA software/database/interface packages. Due acknowledgement is herein made of all such protections.

Contents

1. INTRODUCTION	1
2. USE OF TQ INTERFACE	3
2.1 HOW TO INSTALL AND RUN TQ INTERFACE	3
2.2 HOW TO USE THIS GUIDE.....	4
2.3 FROM VERSION 2.0 TO 3.1	4
2.4 FROM VERSION 3.1 TO 4.0	4
2.5 FROM VERSION 4.0 TO 5.0	5
2.6 FROM VERSION 5.0 TO 6.0	5
2.7 FROM VERSION 6.0 TO 7.0	6
3. BASIC CONCEPTS	7
3.1 NAMES	7
3.2 COMPONENTS	7
3.3 PHASES	8
3.3.1 <i>Phase constituents</i>	8
4. DESCRIPTION OF TQ SUBROUTINES	9
TABLE 1. INITIALIZATION SUBROUTINES	10
TABLE 2. POSSIBLE UNITS USED BY TQSSU AND TQGSU	10
TABLE 3. LEGAL INPUT/OUTPUT OPTIONS USED BY TQSIO AND TQGIO	10
TABLE 4. SUBROUTINES FOR IDENTIFYING COMPONENTS, PHASES, AND CONSTITUENTS.....	11
TABLE 5. SUBROUTINES FOR CHANGING THE STATUS OF COMPONENTS AND PHASES AS WELL AS THE REFERENCE STATE OF COMPONENTS	11
TABLE 6. SUBROUTINES FOR ADDING A CONTRIBUTION TO THE GIBBS ENERGY OF A PHASE.....	11
TABLE 7. LEGAL STATUS FOR COMPONENTS	12
TABLE 8. LEGAL STATUS FOR PHASES	12
TABLE 9. SUBROUTINES FOR SETTING CONDITIONS, STREAMS, OR SEGMENTS FOR EQUILIBRIUM CALCULATIONS.....	12
TABLE 10. POSSIBLE STATE VARIABLES TO BE USED FOR SETTING CONDITIONS IN TQSETC.....	13
TABLE 11. SUBROUTINES AND FUNCTIONS FOR DOING CALCULATION AND GETTING RESULTS	14
TABLE 12. STATE VARIABLE THAT CAN BE USED IN TQGETV AND TQGET1	15
TABLE 13. ADDITIONAL VARIABLES THAT CAN BE USED IN TQGETV AND TQGET1	16
TABLE 14. SUBROUTINES AND FUNCTIONS FOR DEBUGGING AND ERROR HANDLING	16
TABLE 15. SUBROUTINES FOR SPEEDY RETRIEVAL OF IMPORTANT PROPERTIES OF A PHASE	18
TABLE 16. SUBROUTINES FOR RETRIEVAL OF DATA FROM DATABASES	18
4.1 INITIALIZATION SUBROUTINES	19
4.1.1 <i>TQINI(NWG,IWSG)</i>	19
4.1.2 <i>TQSIO(OPTION,IVAL)</i>	19
4.1.3 <i>TQGIO(OPTION,IVAL)</i>	20
4.1.4 <i>TQRFIL(FILE,IWSG)</i>	20
4.1.5 <i>TQSSU(QUANT,UNIT,IWSG)</i>	21
4.1.6 <i>TQGSU(QUANT,UNIT,IWSG)</i>	21
4.1.7 <i>TQSAME(ICODE,IWSG)</i>	21
4.2 SYSTEM SUBROUTINES	22
4.2.1 <i>TQSCOM(NCOM,NAMES,STOI,IWSG)</i>	22
4.2.2 <i>TQGCNOM(NCOM,NAMES,IWSG)</i>	22
4.2.3 <i>TQGSCI(INDEXC,NAME,IWSG)</i>	23
4.2.4 <i>TQGNP (NPH,IWSG)</i>	23
4.2.5 <i>TQGPN (INDEXP,NAME,IWSG)</i>	23
4.2.6 <i>TQGPI (INDEXP,NAME,IWSG)</i>	24
4.2.7 <i>TQGPCN(INDEXP,INDEXC,NAME,IWSG)</i>	24
4.2.8 <i>TQGPCI(INDEXP,INDEXC,NAME,IWSG)</i>	24

4.2.9	<i>TQGCCF(INDEXC,NEL,ELNAM,STOI,MMASS,IWSG)</i>	25
4.2.10	<i>TQGPCS (INDEXP,INDEXC,STOI,MMASS,IWSG)</i>	25
4.2.11	<i>TQGNPC(INDEXP,NPCON,IWSG)</i>	26
4.2.12	<i>TQCSSC (INDEXC,STATUS,IWSG)</i>	26
4.2.13	<i>LOGICAL FUNCTION TQGSSC (INDEXC,IWSG)</i>	26
4.2.14	<i>TQCSP (INDEXP,STATUS,VAL,IWSG)</i>	27
4.2.15	<i>LOGICAL FUNCTION TQGSP (INDEXP,STATUS,VAL,IWSG)</i>	27
4.2.16	<i>TQSETR (INDEXC,INDEXP,TEMP,PRES,IWSG)</i>	28
4.2.17	<i>TQSGA (INDEXP,VALUE,IWSG)</i>	28
4.2.18	<i>TQGGGA (INDEXP,VALUE,IWSG)</i>	28
4.3	CONDITION, STREAM, AND SEGMENT SUBROUTINES	29
4.3.1	<i>TQSETC(STAVAR,INDEXP,INDEXC,VAL,NUMCON,IWSG)</i>	29
4.3.2	<i>TQREMC(NUMCON,IWSG)</i>	30
4.3.3	<i>TQSCURC(IWSG)</i>	30
4.3.4	<i>TQREMAC(IWSG)</i>	30
4.3.5	<i>TQRESTC(IWSG)</i>	30
4.3.6	<i>TQCSTM(IDENT,TEMP,PRESS,IWSG)</i>	31
4.3.7	<i>TQSSC(IDENT,INDEXP,INDEXC,VALUE,NUMIN,IWSG)</i>	31
4.3.8	<i>TQSSIC(STAVAR,VALUE,IWSG)</i>	32
4.3.9	<i>TQDSTM(IDENT,IWSG)</i>	32
4.3.10	<i>TQNSEG(ID,IWSG)</i>	33
4.3.11	<i>TQSSEG(ID,IWSG)</i>	33
4.4	CALCULATION AND RESULTS SUBROUTINES	34
4.4.1	<i>TQCE(TARGET,INDEXP,INDEXC,VALUE,IWSG)</i>	34
4.4.2	<i>TQCEG(IWSG)</i>	34
4.4.3	<i>TQGETV(STAVAR,INDEXP,INDEXC,NUMBER,VALAR,IWSG)</i>	35
4.4.4	<i>TQGETI(STAVAR,INDEXP,INDEXC,VAL,IWSG)</i>	35
4.4.5	<i>DOUBLE PRECISION FUNCTION TQGMU (INDEXC,IWSG)</i>	36
4.4.6	<i>DOUBLE PRECISION FUNCTION TQGGM (INDEXP,IWSG)</i>	36
4.4.7	<i>TQGPD (INDEXP,NSUB,NSCON,SITES,YFRAC,EXTRA,IWSG)</i>	37
4.4.8	<i>TQGDF (IMATR,IPREC,NPH,NCOM,XMATR,XPREC,TEMP,DF,IWSG)</i>	38
4.4.9	<i>TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG)</i>	38
4.5	TROUBLESHOOTING SUBROUTINES	40
4.5.1	<i>TQLS(IWSG)</i>	40
4.5.2	<i>TQLC(IWSG)</i>	40
4.5.3	<i>TQLE(IWSG)</i>	40
4.5.4	<i>TQFASV(IWSG)</i>	41
4.5.5	<i>TQSDMC(INDEXP,IWSG)</i>	41
4.5.6	<i>TQSSPC(INDEXP,YF,IWSG)</i>	41
4.5.7	<i>TQSSV(STAVAR,IP,IC,VALUE,IWSG)</i>	42
4.5.8	<i>TQPINI(IWSG)</i>	42
4.5.9	<i>TQSNL(MAXIT,ACC,YMIN,ADG,IWSG)</i>	42
4.5.10	<i>TQSMNG(NGP,IWSG)</i>	43
4.5.11	<i>STIERR(IERR,SUBR,MESS)</i>	43
4.5.12	<i>ST2ERR(IERR,SUBR,MESS)</i>	44
4.5.13	<i>LOGICAL FUNCTION SGIERR or TQGIERR(IERR)</i>	44
4.5.14	<i>LOGICAL FUNCTION SG2ERR or TQG2ERR(IERR)</i>	44
4.5.15	<i>LOGICAL FUNCTION SG3ERR or TQG3ERR(IERR,SUBR,MESS)</i>	45
4.5.16	<i>RESERR (or TQRSERR)</i>	45
4.6	EXTRA SUBROUTINES	46
4.6.1	<i>TQGMA(INDEXP,TP,YF,VAL,IWSG)</i>	46
4.6.2	<i>TQGMB(INDEXP,TP,VAL,IWSG)</i>	46
4.6.3	<i>TQGMC(INDEXP,VAL,IWSG)</i>	46
4.6.4	<i>TQMDY(INDEXP,VARR,IWSG)</i>	46
4.6.5	<i>TQGMOB(INDEXP,ISP,VAL,IWSG)</i>	47
4.6.6	<i>TQSTP(TP,IWSG)</i>	47
4.6.7	<i>TQSYF(INDEXP,YF,IWSG)</i>	47

4.6.8	<i>TQGSSPI</i> (<i>SPN,ISP,IWSG</i>).....	48
4.6.9	<i>TQCMOBA</i> (<i>INDEXP,ISP,IWSG</i>).....	48
4.6.10	<i>TQCMOBB</i> (<i>INDEXP,IWSG</i>).....	48
4.6.11	<i>TQDGYI</i> (<i>INDEXP,VARR1,VARR2,IWSG</i>).....	48
4.6.12	<i>TQGPHP</i> (<i>INDEXP,NE,NCNV,NC,IWORK,WORK,IWSG</i>).....	49
4.6.13	<i>TQX2Y</i> (<i>INDEXP,NE,NCNV,NC,IWORK,WORK,XF,YF,IWSG</i>).....	49
4.6.14	<i>TQGMDX</i> (<i>IP,NE,NCNV,NC,IWORK,WORK,YF,VARR,GM,DGD,XF,IWSG</i>).....	50
4.7	DATABASE SUBROUTINES (SEE EXAMPLE 12).....	51
4.7.1	<i>TQGDBN</i> (<i>DB_ARR,N,IWSG</i>).....	51
4.7.2	<i>TQOPDB</i> (<i>TDB,IWSG</i>).....	51
4.7.3	<i>TQLIDE</i> (<i>EL_ARR,N,IWSG</i>).....	51
4.7.4	<i>TQAPDB</i> (<i>TDB,IWSG</i>).....	52
4.7.5	<i>TQDEFEL</i> (<i>ELNAM,IWSG</i>).....	52
4.7.6	<i>TQREJEL</i> (<i>ELNAM,IWSG</i>).....	52
4.7.7	<i>TQRESPH</i> (<i>PHNAM,IWSG</i>).....	52
4.7.8	<i>TQREJPH</i> (<i>PHNAM,IWSG</i>).....	53
4.7.9	<i>TQLISPH</i> (<i>PH_ARR,N,IWSG</i>).....	53
4.7.10	<i>TQLISSF</i> (<i>PH_ARR,N,IWSG</i>).....	53
4.7.11	<i>TQGDAT</i> (<i>IWSG</i>).....	54
4.7.12	<i>TQREJS</i> (<i>IWSG</i>).....	54
5.	EXAMPLES.....	55
5.1	EXAMPLE 1.....	55
5.2	EXAMPLE 2.....	57
5.3	EXAMPLE 3.....	59
5.4	EXAMPLE 4.....	62
5.5	EXAMPLE 5.....	65
5.6	EXAMPLE 6.....	67
5.7	EXAMPLE 7.....	70
5.8	EXAMPLE 8.....	73
5.9	EXAMPLE 9.....	74
5.10	EXAMPLE 10.....	76
5.11	EXAMPLE 11.....	78
5.12	EXAMPLE 12.....	81
5.13	EXAMPLE 13.....	84

1. Introduction

TQ is an application programming interface of Thermo-Calc, a general software package for multi-component phase equilibrium calculations. TQ is intended for application programmers to write application programs using the Thermo-Calc kernel without bothering to appreciate its complexity and to follow its continuous improvements and modifications (*Figure 1*). With this programming interface, it is easy to make Thermo-Calc an integral part of application programs such as those for process simulation, microstructure evolution modeling, and materials property prediction. The thermodynamic properties and phase equilibrium data that can be obtained by using TQ include Gibbs energy, enthalpy, entropy, heat capacity, first and second derivatives of Gibbs energy with respect to composition, chemical potential, phase amount, phase composition, partition coefficients, liquidus or solidus points, invariant temperature, heat of reaction, adiabatic combustion temperature, and volume etc. Through appending DICTRA mobility databases into the workspace, one can also obtain assessed mobility or diffusivity data via the TQ interface.

The TQ interface can be applied not only to calculate equilibrium state but also to predict metastable or non-equilibrium state by changing the status of the phases under consideration.

Programming with the TQ interface is simple and easy. The FORTRAN subroutines and functions available in the TQ interface can be classified into six categories according to their purposes (*Figure 2*):

- 1) Initialization — Initializing the workspace, reading the thermodynamic data files, setting units for thermodynamic quantities, and selecting the input and output options.
- 2) System data manipulation — Identifying system components, phases, and constituents and changing their status.
- 3) Setting conditions, streams, and segments — Defining conditions for an equilibrium calculation.
- 4) Performing calculation.
- 5) Obtaining results.
- 6) Troubleshooting — Debugging and Error handling.

Essentially, only subroutines from the first, third, fourth and fifth categories are required for using this interface. In the simplest case, only one or two subroutines are needed from each category.

Same as the Thermo-Calc software, the TQ interface is available for both UNIX platforms and personal computers. It is supplied in the form of DLLs (Dynamically Linked Libraries) so that there is no necessity to recompile existing application programs when a new version of TQ interface is available.

TQ is written in FORTRAN because many other software packages for scientific calculations are still developed in this language. However, the actual computer language to be used to implement application programs is not restricted to FORTRAN, for example a GUI application written in C++ can realize its various functionalities by using TQ subroutines with appropriate calling conventions.

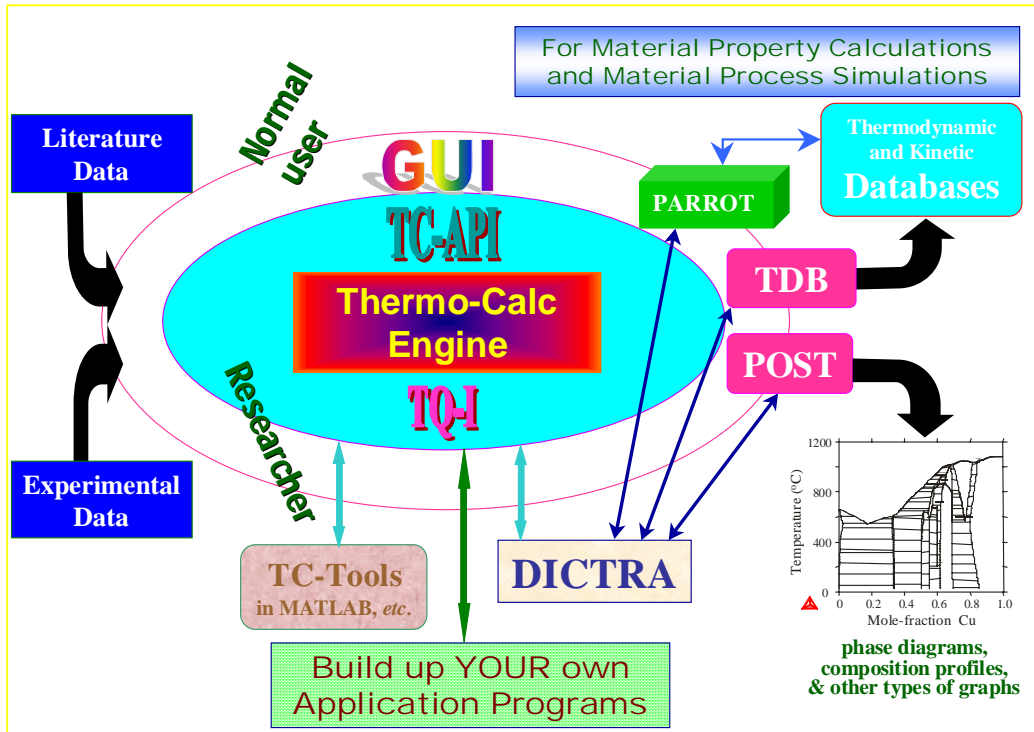


Figure 1. Interfacing with the Thermo-Calc Engine

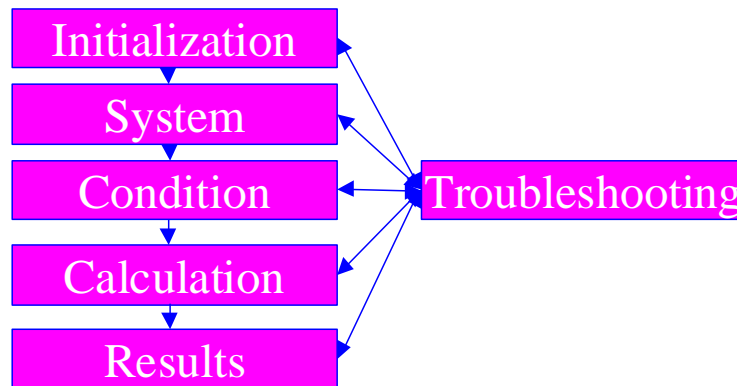


Figure 2. The general structure of the TQ Interface

2. Use of TQ Interface

2.1 How to install and run TQ interface

The TQ interface is only commercially available when it is formally purchased together with the Thermo-Calc Classic (TCC) software/database package. It requires a license key that is generated and provided by Thermo-Calc Software AB based on the hardware information of the installation computer/server provided by the user. The license management for using the TQ interface is similar to that for the Thermo-Calc software; please refer to the *Thermo-Calc Software (TCC and TCW) Installation Guide*.

The TQ interface is available for various operating systems on both PC and Unix computers. These include Windows NT/2000/XP, Windows 95/98/ME, and Linux for PC and SOLARIS and IRIX for Sun Sparc and SGI machines, respectively.

The appropriate compilers required for programming with the TQ interface under various platforms are summarized in the following table:

Hardware		OS	Version	Compiler
PC		Windows	NT4/2000/XP 95/98/ME	Compaq Visual Fortran or Intel Fortran
		Linux	2.2 and above	g77
UNIX Platforms	SUN	SOLARIS	2.5 and above	f77
	SGI	IRIX	? and above	f77

The installation of TQ interface is rather straightforward, *i.e.*, simply unwrap the tar or zip file to any work directory where the user wants. For both Windows and Unix platforms, the TQ interface is supplied as a dynamically linked library since Version 5.0. **All the examples given in this document are provided in a subdirectory called `examples` for Windows and `so_examples` for Unix. Please check the `make` and `run` files (Unix) or `batch` file (Windows) in this directory and see if all examples can be appropriately compiled and linked.**

Along with the TQ interface distribution, a temporary license key file is always provided. The user should act immediately on sending the detailed information about the host-computer identification (serial number) of his/her preferred installation to Thermo-Calc Software AB, so that a permanent license key file can be uniquely generated and provided. If the TQ interface is to be used on the same computer as the TCC installation, such identification information should be the same. The user can usually use the temporal license key file for 1-2 months before it is replaced by the permanent one.

2.2 How to use this guide

Those who are not familiar with the Thermo-Calc software should start from Chapter 3, which defines some basic concepts used in TQ. Chapter 4 gives a full description of the subroutines included in TQ. It is written as a reference manual, organized in a logical order. A glance through Chapter 4 gives you an overview of the steps to program with the TQ interface. Several simple application examples are illustrated in Chapter 5.

An experienced Thermo-Calc user can start by simply copying a suitable example and making it to work for his/her own problems by changing, adding, or deleting some callings to TQ subroutines and functions.

It is strongly recommended that one should compile and link at least one or two examples and make sure the linked executables can be run successfully.

2.3 From version 2.0 to 3.1

The TQ subroutine structure and definitions remained the same for the TQ versions 2.0 and 3.0, whilst the dynamically linked libraries in the TQ distributions were related to different compatible TCC versions (TCC ver. M to TQ ver. 2.0, and TCC ver. N to TQ ver. 3.0). For the TQ version 3.1, the following improvements and modifications were made:

- 1) The subroutine `TQSETC(NP, IP, -1, VAL, NCONP, IWSG)` was changed and it was recommended to be replaced by `TQCSP(IP, 'FIXED', VAL, IWSG)`, which now had one more argument. Correspondingly, `TQREMC(NCONP, IWSG)` was recommended to be replaced by `TQCSP(IP, 'ENTERED', VAL, IWSG)`.
- 2) For the subroutines `TQCSP` and `TQGSP`, one more argument concerning the phase amount was added.
- 3) The subroutines `TQSDMC` and `TQSSPC` missing in TQ version 2.0 became available.
- 4) The subroutines `TQSSV` and `TQPINI` were added.
- 5) The subroutines `TQGMU` and `TQGGM` were added.
- 6) The subroutines `TQSGA` and `TQGGA` were added.

2.4 From version 3.1 to 4.0

Compared to the previous version of the TQ-Interface, the dynamically linked library in the TQ-Interface version 4.0 is compatible to TCC version P.

In addition, some changes have been made to the TQ-Interface it self:

- 1) After calculating equilibrium, it is now possible using TQGET1 to retrieve various kinetic coefficients, e.g. atomic mobilities, tracer diffusion, chemical diffusion and intrinsic diffusion coefficients, see Table 13.
- 2) The subroutine TQGDF was added.
- 3) The subroutine TQSNL was added.

2.5 From version 4.0 to 5.0

The TQ-Interface version 5.0 is compatible to TCC version Q.

It is well-known that TQ provides the fastest engine for thermodynamic equilibrium calculations. Under the circumstances where equilibrium or deviation from equilibrium is taken care of by an application program itself, only properties of individual phases are of interest. In version 5.0, extra subroutines have been added to retrieve Gibbs energy, partial derivative of Gibbs energy w.r.t. site fractions, and mobility data of a phase in an extremely optimized and speedy way (see Table 15):

- 1) TQGMA, TQGMB, and TQGMC for getting Gibbs energy of a phase.
- 2) TQGMDY for getting partial derivative of Gibbs energy w.r.t. site fractions.
- 3) TQGMOB for getting mobility of a species in a phase.
- 4) TQSTP for setting temperature and pressure in order to use TQGMC, TQGMDY, and TQGMOB.
- 5) TQSYF for setting site fractions in order to use TQGMB, TQGMC, TQGMDY, and TQGMOB.
- 6) TQGSSPI for getting index of a species in the defined system.
- 7) TQCMOBA and TQCMOBB for checking if mobility data available.

Reported bugs associated with TQGPD and TQREMC have been removed. Maximum number of equilibrium segments has been increased to 100.

2.6 From version 5.0 to 6.0

Subroutines handling thermodynamic and kinetic database have been added in Version 6. This major upgrade provides application programs a possibility to access directly all Thermo-Calc and/or user-defined databases. Meanwhile, the route to obtain thermodynamic and kinetic data via GES files is still possible without any change. Other important features of this new version include the abilities to calculate nucleation driving force and phase equilibrium under the para-equilibrium conditions, and to obtain the most stable phase equilibrium with the global minimization

technique implemented in Thermo-Calc version R. The following is a list of the new subroutines in TQ version 6:

- 1) TQGDBN to get names of free and licensed databases provided by Thermo-Calc software.
- 2) TQOPDB and TQAPDB to open and append a database respectively.
- 3) TQLIDE, TQDEFEL, and TQREJEL to list, define, and reject element respectively.
- 4) TQLISPH, TQREJPH, and TQRESPH to list, reject, and restore phase respectively.
- 5) TQLISSF to list selected system phases.
- 6) TQGDAT to get data for the defined system from the opened database.
- 7) TQREJS to reject system and reinitiate everything.
- 8) TQGDF2 to get driving force of nucleation and phase equilibrium compositions under para-equilibrium or ortho-equilibrium conditions.
- 9) TQCEG to do equilibrium calculation with the global minimization algorithm.
- 10) TQSMNG to set maximum number of grid points for each phase in global minimization.
- 11) TQSCURC, TQREMAC, and TQRESTC to save, remove, and restore conditions respectively.
- 12) TQGMDX to get first derivatives of Gibbs energy w.r.t. mole fractions. TQX2Y to convert mole fraction to site fraction for phases with no internal degree of freedom. TQGPHP to get constitution properties of a phase, which is necessary before using TQGMDX and TQX2Y.
- 13) TQDGYY to get first and second derivatives of Gibbs energy w.r.t. site fractions.

It should also be mentioned that the undocumented subroutines GPARRD and GPARCD in some of old examples is not supported anymore. Furthermore, the input/output options are initialized by default to 0 in order to avoid standard screen output that can cause serious problems in a GUI programme. As a result, one needs to reset OUTPUT or ERROR option to 6 in order to obtain screen output from SG1ERR, TQLC, TQLE, and TQLS.

2.7 From version 6.0 to 7.0

TQ-Interface version 7.0 is compatible with TCC version S, which means that the greatly improved global minimization technique is available.

3. Basic Concepts

A thermodynamic system is made up of *components* and *phases*. A number of *state variables* define the properties and their relations.

A component is a system wide entity, sometimes this is emphasized by calling it a *system component*. A component has a unique name and some thermodynamic properties are associated with it, for example, amount and activity or chemical potential. At equilibrium the activity or chemical potential of the components are the same in the whole system.

A phase is a system wide entity, which has a composition expressed in the amounts of components, enthalpy content, a volume, and a lot of other properties. The phase has *constituents* that may be different from the components. The constituents have a stoichiometry that can be expressed in terms of the components and possibly a charge. Condensed phases may have an internal structure like sub-lattices or clusters. These clusters may be modeled as constituents.

3.1 Names

A name of a component, phase or constituent must start with a letter A-Z or a-z and contain only letters, digits and the special characters underscore “_”, period “.”, parentheses “(” and “)”, plus “+”, minus “-”, and slash “/”. A name can be maximum 24 characters long.

3.2 Components

The TQ interface maintains a list of components. These are numbered sequentially from 1 up to the number of components. A component has a name which can be identical to a chemical formula or any string of letters like in h2o, c2h2cl_cis, au3cu_cvm1, my-favorite-component. Several subroutines are available for obtaining information about the components and to manipulate them. TQGCOM returns the total number of components and all component names, TQGSCI returns the index of one component name, TQSCOM makes it possible to re-define the components. The component index is used in most subroutines for defining conditions, etc.

Note that components can be suspended by TQCSSC, thus leaving “holes” in the component list because suspending one component will not change the sequential numbering. The logical function TQGSSC can be used to check if a specific component is suspended or not.

3.3 Phases

The TQ interface maintains a list of phases. These are numbered sequentially from 1 up to the number of phases in the system. A phase has many properties and most importantly a list of constituents, see 3.3.1.

The total number of phases is returned by TQGNP; the name of a phase by TQGPN or its index by TQGPI. The number of phase constituents is returned by TQGNPC.

Note that phases can be suspended or set dormant by TQCSP, thus leaving “holes” in the list because suspending one phase will not change the sequential numbering. The logical function TQGSP can be used to check if a specific phase is suspended or not.

3.3.1 Phase constituents

The TQ interface maintains a list of the constituents of each phase. These are numbered sequentially from 1 up to the number of constituents in the phase. The number of constituents can be different in each phase. If a phase has sub-lattices, the numbering will go from the first constituent in the first sub-lattice over all sub-lattices to the last constituent in the last sub-lattice.

The number of constituents of a phase is returned by TQGNPC; the name of a phase constituent by TQGPN or its index by TQGPI. The stoichiometry of a constituent expressed in terms of the components is returned by TQGPCS.

4. Description of TQ Subroutines

The subroutines and functions defined in the TQ interface can be divided into five categories according to their purposes:

- Initialization subroutines (see Table 1)
 - ⇒ Initializing the TQ workspace.
 - ⇒ Reading a thermodynamic data file into the workspace.
 - ⇒ Re-setting default units for temperature, pressure, energy, mass, and volume. A list of possible units is shown in Table 2.
 - ⇒ Changing the program input and output units. The legal options for program input/output are listed in Table 3.
- System subroutines (see Table 4-6)
 - ⇒ Getting number, names, and indexes of the system components, phases, and constituents of each phase.
 - ⇒ Re-define the system components.
 - ⇒ Changing the status of components and phases. The legal status is listed in Table 7 for components and Table 8 for phases.
 - ⇒ Changing the reference state of the system.
 - ⇒ Adding a contribution to the Gibbs energy of a phase.
- Condition, stream, and segment subroutines (see Table 9)
 - ⇒ Setting conditions for a thermodynamic equilibrium calculation. Possible state variables as conditions are listed in Table 10.
 - ⇒ Setting streams. A stream is considered as a non-reactive medium for transferring matter to a reaction zone. It has constant temperature and pressure, and contains one or more phases of a certain composition, i.e., for each stream, temperature, pressure, and input amounts of phase constituents must be defined.
 - ⇒ Setting a new equilibrium segment. Different sets of equilibrium conditions can be defined for the same system in different segments.
- Calculation subroutines (see Table 11)
 - ⇒ Performing calculation.
 - ⇒ Obtaining results. A list of state variables that can be used for getting results is given in Table 12.
- Troubleshooting subroutines (see Table 13)
 - ⇒ Getting information directly from the Thermo-Calc for debugging.
 - ⇒ Re-setting start values for an equilibrium calculation.
 - ⇒ Getting error number and message.

Table 1. Initialization subroutines

<i>Subroutine</i>	<i>Purpose</i>
<u>TOINI(NWG,IWSG)</u>	Initialize TQ interface
<u>TOSIO(OPTION,IVAL)</u>	Set input/output option
<u>TOGIO(OPTION,IVAL)</u>	Get input/output option
<u>TORFIL(FILE,IWSG)</u>	Read thermodynamic data file
<u>TOSSU(QUANT,UNIT,IWSG)</u>	Set unit for a system quantity
<u>TOGSU(QUANT,UNIT,IWSG)</u>	Get unit for a system quantity
<u>TOSAME(ICODE,IWSG)</u>	Check if the system is same

Table 2. Possible units used by TQSSU and TQGSU

<i>Quantity</i>	<i>Unit</i>	<i>Comment</i>
<i>Temperature</i>	K	Kelvin (default)
	C	Celcius. Calculated as K-273.15
	F	Fahrenheit
<i>Volume</i>	M3	Cubic meter (default)
	L	Liter. Calculated as 0.001 M3
	IN3	Cubic inch.
	FT3	Cubic feet
	USG	US gallon
<i>Energy</i>	J	Joule (default)
	Cal	Calories. Calculated as J/4.184
	BTU	British thermal units.
<i>Pressure</i>	Pa	Pascal (default)
	Psi	Pounds/sq. inch
	Bar	Bar. Calculated as 0.00001*Pa
	Atm	Atmosphere. Calculated as Pa/101325
	Torr	Torricelli. Calculated as 758*Pa/101325

Table 3. Legal input/output options used by TQSIO and TQGIO

<i>Option</i>	<i>Meaning</i>	<i>Default value</i>
<i>INPUT</i>	Input unit	0
<i>OUTPUT</i>	Output unit	0
<i>ERROR</i>	Error output	0
<i>LIST</i>	List output	0

Table 4. Subroutines for identifying components, phases, and constituents

<i>Subroutine</i>	<i>Purpose</i>
<u>TOSCOM(NCOM,NAMES,STOI,IWSG)</u>	Set system components
<u>TOGCOM(NCOM,NAMES,IWSG)</u>	Get system components
<u>TOGSCI(INDEXC,NAME,IWSG)</u>	Get system component index
<u>TOGNP(NPH,IWSG)</u>	Get number of phases
<u>TOGPN(INDEXP,NAME,IWSG)</u>	Get phase name
<u>TOGPI(INDEXP,NAME,IWSG)</u>	Get phase index
<u>TOGPCN(INDEXP,INDEXC,NAME,IWSG)</u>	Get phase constituent name
<u>TOGPCI(INDEXP,INDEXC,NAME,IWSG)</u>	Get phase constituent index
<u>TOGCCF(INDEXC,NEL,ELNAM,STOI,MMASS,IWSG)</u>	Get component chemical formula
<u>TOGPCS(INDEXP,INDEXC,STOI,MMASS,IWSG)</u>	Get phase constituent stoichiometry
<u>TOGNPC(INDEXP,NPCON,IWSG)</u>	Get number of phase constituents

Table 5. Subroutines for changing the status of components and phases as well as the reference state of components

<i>Subroutine</i>	<i>Purpose</i>
<u>TOCSSC(INDEXC,STATUS,IWSG)</u>	Change status of system component
<u>TOGSSC(INDEXC,IWSG)*</u>	Get status of system component
<u>TOCSP(INDEXP,STATUS,VAL,IWSG)</u>	Change status of phase
<u>TOGSP(INDEXP,STATUS,VAL,IWSG)*</u>	Get status of phase
<u>TOSETR(INDEXC,INDEXP,TEMP,PRES,IWSG)</u>	Set reference state

* *Logical function.*

Table 6. Subroutines for adding a contribution to the Gibbs energy of a phase

<i>Subroutine</i>	<i>Purpose</i>
<u>TOSGA(INDEXP,VALUE,IWSG)</u>	Set Gibbs energy addition
<u>TOGGA(INDEXP,VALUE,IWSG)</u>	Get Gibbs energy addition

Table 7. Legal status for components

<i>Status</i>	<i>Meaning</i>
ENTERED	The component is included in the system for an equilibrium calculation.
SUSPENDED	The component is excluded from the system and, as a result, some phases may become suspended if their constituents contain this component.

Table 8. Legal status for phases

<i>Status</i>	<i>Meaning</i>
ENTERED	The phase is included in an equilibrium calculation. It may be stable or unstable.
DORMANT	The phase is included in an equilibrium calculation but not allowed to become stable. The phase should be stable if the calculation shows that its driving force is positive (or activity is larger than unity)
FIXED	The phase is included in an equilibrium calculation and it must be stable.
SUSPENDED	The phase is ignored in an equilibrium calculation.

Table 9. Subroutines for setting conditions, streams, or segments for equilibrium calculations

<i>Subroutine</i>	<i>Purpose</i>
<u>TOSETC(STAVAR,INDEXP,INDEXC,VAL,NUMCON,IWSG)</u>	Set condition
<u>TOREMC(NUMCON,IWSG)</u>	Remove condition
<u>TOSCURC(IWSG)</u>	Save current conditions
<u>TOREMAC(IWSG)</u>	Remove all conditions
<u>TORESTC(IWSG)</u>	Restore saved conditions
<u>TOCSTM(IDENT,TEMP,PRESS,IWSG)</u>	Create stream
<u>TOSSC(IDENT,INDEXP,INDEXC,VALUE,NUMIN,IWSG)</u>	Set stream constituent amount
<u>TOSSIC(STAVAR,VALUE,IWSG)</u>	Set stream invariant state variable
<u>TODSTM(IDENT,IWSG)</u>	Delete stream
<u>TONSEG(ID,IWSG)</u>	Create new equilibrium segment
<u>TOSSEG(ID,IWSG)</u>	Select equilibrium segment

Table 10. Possible state variables to be used for setting conditions in TQSETC

<i>STAVAR</i>	<i>INDEXP</i>	<i>INDEXC</i>	<i>Meaning</i>	<i>Comments</i>
<i>T</i>			Temperature	of the whole system
<i>P</i>			Pressure	of the whole system
<i>MU</i>	<i>note 1</i>	Yes	Chemical potential	of a system component
<i>MUC</i>	Yes	Yes	Chemical potential	of a phase constituent
<i>AC</i>	<i>note 1</i>	Yes	Activity	of a system component
<i>ACC</i>	Yes	Yes	Activity	of a system constituent
<i>V</i>			Volume	of the whole system
<i>G</i>			Gibbs energy	of the whole system
<i>H</i>			Enthalpy	of the whole system
<i>S</i>			Entropy	of the whole system
<i>N</i>			Moles	of all system components
<i>N</i>		Yes	Moles	of a system component
<i>NP</i>	<i>note 2</i>		Moles	of a phase
<i>M</i>			Total mass	of all system components
<i>M</i>		Yes	Mass	of a system component
<i>MP</i>	<i>note 2</i>		Mass	of a phase
<i>IN</i>	Yes	Yes	Input amount	in moles of phase constituents
<i>IM</i>	Yes	Yes	Input amount	in mass units of phase constituents
<i>X</i>		Yes	Mole fraction	of a system component
<i>W</i>		Yes	Mass (Weight) fraction	of a system component
<i>X%</i>		Yes	Mole percent	of a system component
<i>W%</i>		Yes	Mass (Weight) fraction	of a system component

Note 1: Giving a phase index means to define the reference state. If no phase index is given the previous reference state is used. The default reference state is SER (Standard Element Reference) if the thermodynamic data file is created from a SGTE (Scientific Group Thermodata Europe) database. It is necessary that the phase can exist with the constituent as its single constituent. It is an error to set FCC as reference state for carbon if carbon dissolves interstitially in FCC.

Note 2: Not recommended to be used for setting conditions. To calculate stability limit one should use TQCSP with FIXED status and amount of the phase set to zero.

One may add a normalizing suffix like M (per mole), W (per mass) or V (per volume) on G, H, S, etc.

Table 11. Subroutines and functions for doing calculation and getting results

<i>Subroutine</i>	<i>Purpose</i>
<u>TOCE(TARGET,INDEXP,INDEXC,VALUE,IWSG)</u>	Calculate equilibrium
<u>TOCEG(IWSG)</u>	Calculate global equilibrium
<u>TOGETV(STAVAR,INDEXP,INDEXC,NUMBER, VALAR,IWSG)</u>	Get equilibrium property values
<u>TOGET1(STAVAR,INDEXP,INDEXC,VAL, IWSG)</u>	Get one value
<u>TOGMU(INDEXC, IWSG)*</u>	Get chemical potential value
<u>TOGGM(INDEXP, IWSG)*</u>	Get molar Gibbs energy value
<u>TOGPD(INDEXP,NSUB,NSCON,SITES,YFRAC, EXTRA,IWSG)</u>	Get phase data
<u>TOGDF(IMATR,IPREC,NPH,NCOM,XMATR, XPREC,TEMP,DF,IWSG)</u>	Get driving force for phase transformation. Obsolete
<u>TOGDF2(MODE,IMATR,IPREC,NL,INL,XMATR, TEMP,DF,XPREC,XEM,XEP,MUI,IWSG)</u>	Get driving force and local equilibrium compositions for ortho- or para-equilibrium phase transformation

* *Double precision function.*

Table 12. State variable that can be used in TQGETV and TQGET1

<i>STAVAR</i>	<i>INDEXP</i>	<i>INDEXC</i>	<i>Meaning</i>	<i>Comments</i>
<i>T</i>			Temperature	of the whole system
<i>P</i>			Pressure	of the whole system
<i>MU</i>	(yes)	Yes	Chemical potential	of a system component
<i>MUC</i>	Yes	yes	Chemical potential	of a constituent in a gas phase
<i>AC</i>	(yes)	Yes	Activity	of a system component
<i>ACC</i>	Yes	Yes	Activity	of a constituent in a gas phase
<i>V</i>			Volume	of the whole system
<i>V</i>	Yes		Volume	of a phase
<i>G*</i>			Gibbs energy	of the whole system
<i>G*</i>	Yes		Gibbs energy	of a phase
<i>H*</i>			Enthalpy	of the whole system
<i>H*</i>	Yes		Enthalpy	of a phase
<i>S*</i>			Entropy	of the whole system
<i>S*</i>	Yes		Entropy	of a phase
<i>CP</i>			Heat capacity	of the system
<i>CP</i>	Yes		Heat capacity	of a phase
<i>DG</i>	Yes		Driving force	of a phase
<i>N</i>			Moles	of all system components
<i>N</i>		Yes	Moles	of a system component
<i>NP</i>	Yes		Moles	of a system phase
<i>M</i>			Total mass	of all system components
<i>M</i>		Yes	Mass	of a system component
<i>MP</i>	Yes		Mass	of a system phase
<i>IN</i>	Yes	Yes	Input amount	in moles of phase constituents
<i>IM</i>	Yes	Yes	Input amount	in mass units of phase constituents
<i>X</i>		Yes	Mole fraction	of a component in the whole system
<i>X</i>	Yes	Yes	Mole fraction	of a component in a phase
<i>W</i>		Yes	Mass (Weight) fraction	of a component in the whole system
<i>W</i>	Yes	Yes	Mass (Weight) fraction	of a component in a phase
<i>X%</i>		Yes	Mole percent	of a component in the whole system
<i>X%</i>	Yes	Yes	Mole percent	of a component in a phase
<i>W%</i>		Yes	Mass (Weight) fraction	of a component in the whole system
<i>W%</i>	Yes	Yes	Mass (Weight) fraction	of a component in a phase
<i>Y</i>	Yes	Yes	Constituent fraction	of a phase constituent

* One can add a normalizing suffix like M (per mole), W (per mass) or V (per volume) on G, H, S, etc. R can also be added as a suffix on G, H, S in order to get a value which is calculated with respect to the reference state specified by calling TQSETR.

Table 13. Additional variables that can be used in TQGETV and TQGET1

STAVAR	IND EXP	IND EXC	Meaning	Unit
<i>M(phase,J)</i>			Mobility coefficient where J=diffusing specie	m ² /s
<i>LOGM(phase,J)</i>			¹⁰ log of the mobility coefficient	m ² /s
<i>DT(phase,J)</i>			Tracer diffusion coefficient where J=diffusing specie	m ² /s
<i>LOGDT(phase,J)</i>			¹⁰ log of the tracer diffusion coefficient	m ² /s
<i>DC(phase,J,K,N)</i>			Chemical diffusion coefficient where K=gradient specie, and N=reference specie	m ² /s
<i>LOGDC(phase,J,K,N)</i>			¹⁰ log of the chemical diffusion coefficient	m ² /s
<i>DI(phase,J,K,N)</i>			Intrinsic diffusion coefficient	m ² /s
<i>LOGDI(phase,J,K,N)</i>			¹⁰ log of the intrinsic diffusion coefficient	m ² /s
<i>QC(phase,J,K,N)</i>			$Q=R(\ln(DC\{T_1\})-\ln(DC\{T_1+\varepsilon\}))/((1/(T_1+\varepsilon))-1/T_1)$	J/mol
<i>QT(phase,J)</i>			$Q=R(\ln(DT\{T_1\})-\ln(DT\{T_1+\varepsilon\}))/((1/(T_1+\varepsilon))-1/T_1)$	J/mol
<i>QI(phase,J,K,N)</i>			$Q=R(\ln(DI\{T_1\})-\ln(DI\{T_1+\varepsilon\}))/((1/(T_1+\varepsilon))-1/T_1)$	J/mol
<i>FC(phase,J,K,N)</i>			$D_0=\exp(\ln(DC\{T_1\})+Q/R/T_1)$	m ² /s
<i>FI(phase,J,K,N)</i>			$D_0=\exp(\ln(DI\{T_1\})+Q/R/T_1)$	m ² /s
<i>FT(phase,J)</i>			$D_0=\exp(\ln(DT\{T_1\})+Q/R/T_1)$	m ² /s

Table 14. Subroutines and functions for debugging and error handling

Subroutine	Purpose
<u>TOLS(IWSG)</u>	List status
<u>TOLC(IWSG)</u>	List conditions
<u>TOLE(IWSG)</u>	List equilibrium
<u>TOFASV(IWSG)</u>	Force automatic start values
<u>TOSDMC(INDEXP,IWSG)</u>	Set default major constituent
<u>TOSSPC(INDEXP,YF,IWSG)</u>	Set start phase constitution
<u>TOSSV(STAVAR,IP,IC,VALUE,IWSG)</u>	Set start value of a state variable
<u>TOPINI(IWSG)</u>	Reinitiate the calculation workspace
<u>TQSNL(MAXIT,ACC,YMIN,ADG,IWSG)</u>	Set numerical limits
<u>TQSMO(STRING,IWSG)</u>	Set minimization options
<u>STIERR(IERR,SUBR,MESS)</u>	Set error code and give message
<u>ST2ERR(IERR,SUBR,MESS)</u>	Set error code
<u>SGIERR(IERR)*</u>	Get error code and give message
<u>SG2ERR(IERR)*</u>	Get error code
<u>SG3ERR(IERR,SUBR,MESS)*</u>	Get error code and message
<u>RESERR</u>	Reset error code and message

* *Logical function.*

Table 15. Subroutines for speedy retrieval of important properties of a phase

<i>Subroutine</i>	<i>Purpose</i>
<u>TOGMA(INDEXP,TP,YF,VAL,IWSG)</u>	Get Gibbs energy of a phase*
<u>TOGMB(INDEXP,TP,VAL,IWSG)</u>	Get Gibbs energy of a phase*
<u>TOGMC(INDEXP,VAL,IWSG)</u>	Get Gibbs energy of a phase*
<u>TOGMDY(INDEXP,VARR,IWSG)</u>	Get 1 st partial derivative of Gibbs energy w.r.t. site fractions.*
<u>TODGYI(INDEXP,VARR1,VARR2,IWSG)</u>	Get 1 st and 2 nd partial derivative of Gibbs energy w.r.t. site fractions.*
<u>TOGPHP(INDEXP,NE,NCNV,NC,IWORK,WORR,IWSG)</u>	Get constitutional properties of a phase.*
<u>TOX2Y(INDEXP,NE,NCNV,NC,IWORK,WORR,X,YF,IWSG)</u>	Convert mole fraction to site fraction for phases with no internal degree of freedom.*
<u>TOGMDX(INDEXP,NE,NCNV,NC,IWORK,WORR,YF,DGDY,GM,DGDY,X,IWSG)</u>	Convert 1 st partial derivative of Gibbs energy w.r.t. site fractions to that w.r.t. mole fractions.*
<u>TOGMOB(INDEXP,ISP,VAL,IWSG)</u>	Get mobility of a species in a phase
<u>TOSTP(TPARR,IWSG)</u>	Set temperature and pressure for TQGMC, TQGMDY, and TQGMOB
<u>TOSYF(INDEXP,YF,IWSG)</u>	Set site fractions for TQGMB, TQGMC, TQGMDY, and TQGMOB
<u>TOGSSPI(SPN,ISP,IWSG)</u>	Get index of a system species
<u>TOCMOBA(INDEXP,ISP,IWSG)</u>	Check if mobility of a species in a phase available
<u>TOCMOBB(INDEXP,IWSG)</u>	Check if mobility data for a phase available

*in SI unit for one mole of formula unit

Table 16. Subroutines for retrieval of data from databases

<i>Subroutine</i>	<i>Purpose</i>
<u>TOGDBN(DB_ARR,N,IWSG)</u>	Get lists of database names
<u>TOOPDB(TDB,IWSG)</u>	Open or switch to a database
<u>TOLIDE(EL_ARR,N,IWSG)</u>	List database elements
<u>TOAPDB(TDB,IWSG)</u>	Append a database
<u>TODEFEL(ELNAM,IWSG)</u>	Select an element
<u>TOREJEL(ELNAM,IWSG)</u>	Reject a selected element
<u>TOREJPH(PHNAM,IWSG)</u>	Reject a phase or all phases
<u>TORSPH(PHNAM,IWSG)</u>	Restore a phase
<u>TOLISPH(PH_ARR,N,IWSG)</u>	List phases related to the selected element(s)
<u>TOLISSF(PH_ARR,N,IWSG)</u>	List retained phases for the selected element(s)
<u>TOGDAT(IWSG)</u>	Get data from the selected database
<u>TOREJS(IWSG)</u>	Reject defined system and reinitiate workspace

4.1 Initialization Subroutines

4.1.1 TQINI(NWG,IWSG)

Full name: Initialize TQ Interface.

Purpose: With this subroutine the application program initializes the Thermo-Calc package for thermodynamic calculations. It must be called before using any other subroutines in the TQ interface.

Arguments:	Name	Type	Value set on call or returned
	NWG	Integer	Set to size of the workspace IWSG.
	IWSG	Integer array	Memory area for storage of data inside the package.

Comments: IWSG is an integer array argument used in most TQ subroutines. It is used by the subroutines within the TQ interface and should not be manipulated by the application program. Different implementations of the TQ interface may use IWSG differently but basically it should be used to store or identify the storage of thermodynamic data local to the application.

The workspace should be set large enough for storing the thermodynamic data. A typical value for NWG would be 80000.

Only one time can this subroutine be called in the same application program, otherwise the common block data will be destroyed.

4.1.2 TQSIO(OPTION,IVAL)

Full name: Set Input/output Option.

Purpose: With this subroutine the application program can re-direct input and output from the Thermo-Calc package.

Arguments:	Name	Type	Value set on call or returned
	OPTION	Character*8	Set to a value given in Table 3.
	IVAL	Integer	Set to an internal value.

Comments: OPTION is a character identifying the Input/Output option. The current internal value is set to the value in IVAL. If the value is illegal the error condition is set.

4.1.3 TQGIO(OPTION,IVAL)

Full name: Get Input/output Unit.

Purpose: Obtain a value of Input/Output option.

Arguments:	Name	Type	Value set on call or returned
	OPTION	Character*8	Set to a value given in Table 3.
	IVAL	Integer	Return the current internal value.

Comments: OPTION is a character identifying the Input/Output option. IVAL is an integer where its current internal value is returned.

4.1.4 TQRFIL(FILE,IWSG)

Full name: Read File.

Purpose: Read a thermodynamic data file in the Thermo-Calc format.

Arguments:	Name	Type	Value set on call or returned
	FILE	Character*60	Legal file name.
	IWSG	Integer array	Workspace.

Comments: The default set of components is supplied by the thermodynamic input file.

The thermodynamic data file should contain at least the following information.

System	⇒ name of the elements
	⇒ molecular mass for elements
	⇒ list of phases
Phase	⇒ list of constituents
	⇒ type of solution model (if not fixed composition)
	⇒ thermodynamic model parameters
Constituents	⇒ name
	⇒ chemical formula (stoichiometric matrix)
	⇒ molecular mass
	⇒ thermodynamic properties

All this data are not necessarily stored separately, for example the molecular weight for a constituent can be calculated from the masses of the elements.

The TQ interface is not intended to read from a database or a database file and thus selections of data from a database must be made in the Thermo-Calc and then stored in a GES file by using the `save` command in the Gibbs-Energy-System module inside the Thermo-Calc. When the GES file has been read into the workspace by this subroutine it is possible to manipulate data by changing components and status for components or phases.

4.1.5 TQSSU(QUANT,UNIT,IWSG)

Full name: Set System Unit.

Purpose: Set the unit for a quantity (like mass, volume, etc.).

Arguments:	Name	Type	Value set on call or returned
	QUANT	Character*60	Set to a legal quantity as listed in Table 2.
	UNIT	Character*60	Set to a legal unit, see below in Table 2.
	IWSG	Integer array	Workspace.

Comments: Default units are SI unless changes are made by this subroutine. The legal quantities and units are listed in Table 2

4.1.6 TQGSU(QUANT,UNIT,IWSG)

Full name: Get System Unit.

Purpose: To find what units the TQ interface is currently using for a system quantity.

Arguments:	Name	Type	Value set on call or returned
	QUANT	Character*60	Set to a legal quantity listed in Table 2.
	UNIT	Character*60	Return the current unit.
	IWSG	Integer array	Workspace.

Comments: The legal quantities and units are listed in Table 2.

4.1.7 TQSAME(ICODE,IWSG)

Full name: Same System.

Purpose: With this subroutine the application program can check if the thermochemical system has been changed, i.e., not just the conditions but the components or the phases. This is useful if several independent systems operate on the same equilibrium description.

Arguments:	Name	Type	Value set on call or returned
	ICODE	Integer	Returns an internal code.
	IWSG	Integer array	Workspace.

Comments: ICODE is an integer with positive value identifying current system. If ICODE is not the same next time TQSAME is called, the system has been changed.

This routine may have to be used if the set of components or the set of phases has been changed. The value of ICODE is changed if there are changes of the components, phases, etc., but not with changes in the conditions, or values of thermodynamic model parameters etc.

4.2 System Subroutines

4.2.1 TQSCOM(NCOM,NAMES,STOI,IWSG)

Full name: Set System Component.

Purpose: A new set of system components can be defined. The new number of components must be the same as previously. The number of system components can be changed by suspending a component by TQCSSC.

Arguments:	Name	Type	Value set on call or returned
	NCOM	Integer	Set to the number of components.
	NAMES	Character*24 array	Set to component names.
	STOI	Double precision matrix	Stoichiometry matrix in old components.
	IWSG	Integer array	Workspace.

Comments: The set of components must be linearly independent. The names of the new system components are given in NAMES. STOI is a matrix with dimension STOI(1:NCOM,1:NCOM) which gives the stoichiometry of the new components expressed in the old ones.

The default set for components is taken from in the input thermodynamic data file.

Legal values for the array elements in NAMES are constituent names.

The components will be numbered as 1... NCOM in the order they are supplied in this call. The conversion from component name to index is also done by TQGSCI.

Example: To transform from the components A, B, C to A2B, B4C C2 use the following values of STOI:

A2B	2.0	1.0	0.0
B4C	0.0	4.0	1.0
C2	0.0	0.0	2.0

4.2.2 TQGCOT(NCOM,NAMES,IWSG)

Full name: Get System Component.

Purpose: Get components of a system

Arguments:	Name	Type	Value set on call or returned
	NCOM	Integer	Return the current number of components.
	NAMES	Character*24 array	Return the current names of components.
	IWSG	Integer array	Workspace.

Comments: The number of components are returned in NCOM and their names are returned in NAMES. They are returned in an internal sequential order of the TQ interface. In other subroutines one must in some cases use the index of a component rather than the name. See also TQGSCI.

4.2.3 TQGSCI(INDEXC,NAME,IWSG)

Full name: Get System Component Index.

Purpose: Get index of a system component.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Return the index of the component.
	NAMES	Character*24	Set to a component name.
	IWSG	Integer array	Workspace.

Comments: This is a way to translate from a name to an index. In order to translate from a component index to a name, use TQGCOM. The application program may call TQGCOM only once and maintain itself a list of component names stored by indices.

4.2.4 TQGNP (NPH,IWSG)

Full name: Get Number of Phases.

Purpose: With this subroutine application program can get numbers of phases.

Arguments:	Name	Type	Value set on call or returned
	NPH	Integer	Return the number of phases.
	IWSG	Integer array	Workspace.

Comments: The phases may have any status. They are numbered sequentially from 1 to NPH. Phases with miscibility gap and thus having more than one composition set are counted separately.

4.2.5 TQGPN (INDEXP,NAME,IWSG)

Full name: Get Phase Name.

Purpose: With this subroutine application program can convert a phase index to the name of the phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to the index of a phase.
	NAME	Character*24	Return the name of the phase.
	IWSG	Integer array	Workspace.

Comments: The conversion from phase name to phase index is done by TQGPI. Note that phases with miscibility gaps must appear with each possible composition set as a separate phase. These are named as BCC#1, BCC#2 etc.

4.2.6 TQGPI (INDEXP,NAME,IWSG)

Full name: Get Phase Index.

Purpose: With this subroutine the application program can get the index of a named phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Return the index of a phase.
	NAME	Character*24	Set to a phase name.
	IWSG	Integer array	Workspace.

Comments: The conversion from phase index to phase name is done by TQGPN.

4.2.7 TQGPCN(INDEXP,INDEXC,NAME,IWSG)

Full name: Get Phase Constituent Name.

Purpose: With this subroutine application program can get the name of an indexed constituent.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	INDEXC	Integer	Set to the constituent index.
	NAME	Character*24	Return the constituent name.
	IWSG	Integer array	Workspace.

Comments: If the same species appear in more than one sublattice site of a phase, they will be named as A#2, A#3, etc., which means A on the second sublattice and A on the third sublattice, etc.

The opposite conversion is done by TQGPCI.

4.2.8 TQGPCI(INDEXP,INDEXC,NAME,IWSG)

Full name: Get Phase Constituent Index.

Purpose: With this subroutine application program can get the index of a constituent if its name is known.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	INDEXC	Integer	Return the constituent index.
	NAME	Character*24	Set to the constituent name.
	IWSG	Integer array	Workspace.

Comments: The opposite conversion is done by TQGPCN.

4.2.9 TQGCCF(INDEXC,NEL,ELNAM,STOI,MMASS,IWSG)

Full name: Get Component Chemical Formula.

Purpose: With this subroutine application program can get the stoichiometry array for a system component in terms of element.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Set to system a component index.
	NEL	Integer	Number of elements in chemical formula.
	ELNAM	Character*2 array	Element symbols.
	STOI	Double precision array	Stoichiometry array.
	MMASS	Double precision	Total mass.
	IWSG	Integer array	Workspace.

Comments: With this subroutine it is possible to obtain the “real” elements in a component. All other subroutines just deal with a name that does not have to be related to the actual chemical formula. This is also the only subroutine that can provide the symbols of the actual elements in the system.

Note: the dimension of ELNAM and STOI will be NEL (number of elements in the component).

4.2.10 TQGPCS (INDEXP,INDEXC,STOI,MMASS,IWSG)

Full name: Get Phase Constituent Stoichiometry.

Purpose: With this subroutine application program can obtain the stoichiometry of a constituent expressed in the system components and also the molecular mass.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	INDEXC	Integer	Set to the constituent index.
	STOI	Double precision array	Return the stoichiometry array.
	MMASS	Double precision	Return the mass.
	IWSG	Integer array	Workspace.

Comments: This does not give the chemical formula in terms of elements for the constituent. The dimension of STOI will be NCOM (number of components) got by calling TQGCOM.

4.2.11 TQGNPC(INDEXP,NPCON,IWSG)

Full name: Get Number of Phase Constituent.

Purpose: With this subroutine the number of constituents in a phase can be obtained.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	NPCON	Integer	Return the number of the phase constituents.
	IWSG	Integer array	Workspace.

Comments: To have also the names, fractions etc. of the constituents, use TQGPD.

4.2.12 TQCSSC (INDEXC,STATUS,IWSG)

Full name: Change Status of System Component.

Purpose: With this subroutine the application program can change status for a system component.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Set to a component index.
	STATUS	Character*12	Set to the new status (see Table 7).
	IWSG	Integer array	Workspace.

Comments: The legal values for STATUS are **ENTERED** and **SUSPENDED**. See also Table 7.

By suspending a system component some phases may also become suspended if they contain this component. For example, in the system Fe-O-S if O is suspended all phases that must dissolve oxygen is automatically suspended. The fraction of oxygen is set to zero in phases that can dissolve oxygen but can also exist without oxygen.

4.2.13 LOGICAL FUNCTION TQGSSC (INDEXC,IWSG)

Full name: Get Status of System Component.

Purpose: This function returns TRUE if the system component is **ENTERED** or FALSE if it is **SUSPENDED**.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Set to a component index.
	IWSG	Integer array	Workspace.

Comments: The legal values for STATUS are given in Table 6.

4.2.14 TQCSP (INDEXP,STATUS,VAL,IWSG)

Full name: Change Status of Phase.

Purpose: With this subroutine the application program can change status for a phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	STATUS	Character*12	Set to the status code (see Table 8).
	VAL	Double precision	Set to phase amount in number of moles.
	IWSG	Integer array	Workspace.

Comments: The legal values for STATUS are given in Table 8.

For **ENTERED** phase, VAL is provided as a start value. It is normally set to zero if the phase is not likely to be stable and one if expected to be stable

Setting a phase **SUSPENDED** or **DORMANT** is a way to calculate a metastable equilibrium if the phase would be stable. With the **DORMANT** status one can know if it would be stable or not. For these two statuses, VAL is irrelevant and may be simply put to zero.

For **FIXED** phase the exact amount of the phase must be given. Please note that the amount is in number of moles of atoms, which means that the vacancy in a sublattice phase will not be included. Therefore, for a vacancy-containing sublattice phase with **FIXED** status, it is impossible to set VAL to the total number of moles in the system.

Setting a phase **FIXED** will decrease the degrees of freedom in the system by 1. To restore the lost degree of freedom the phase should be reset **ENTERED**.

Set a **FIXED** phase to zero amount is the best way to get the phase stability limits like liquidus or solidus.

4.2.15 LOGICAL FUNCTION TQGSP (INDEXP,STATUS,VAL,IWSG)

Full name: Get Status of Phase.

Purpose: This function is TRUE if the phase is **ENTERED** or **FIXED**. If the phase is **SUSPENDED** or **DORMANT** it is FALSE. The status is also returned in STATUS. The application program can test the status of a phase by calling this function.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	STATUS	Character*12	Return the current status code.
	VAL	Double precision	Return the phase amount.
	IWSG	Integer array	Workspace.

Comments: The legal values for STATUS are listed for Table 8.

4.2.16 TQSETR (INDEXC,INDEXP,TEMP,PRES,IWSG)

Full name: Set Reference State.

Purpose: With this subroutine the reference state of a system component can be reset.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Set to a component index.
	INDEXP	Integer	Set to a phase index.
	TEMP	Double precision	Set to a temperature value.
	PRES	Double precision	Set to a pressure value.
	IWSG	Integer array	Workspace.

Comments: By default the reference state for a component is determined by the thermodynamic data file. With this subroutine an application may select a different reference state if the one in the data file does not suit a calculation purpose.

If the current temperature or pressure should be used for the calculation, the value given should not be larger than zero.

4.2.17 TQSGA (INDEXP,VALUE,IWSG)

Full name: Set Gibbs energy Addition.

Purpose: With this subroutine an amount of extra contribution can be added to the Gibbs energy of a phase

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	VALUE	Double precision	Set to the value of extra contribution.
	IWSG	Integer array	Workspace.

Comments: The extra contribution may be due to elastic strain energy, surface energy, etc.

4.2.18 TQGGA (INDEXP,VALUE,IWSG)

Full name: Get Gibbs energy Addition.

Purpose: With this subroutine the contribution added to the Gibbs energy of a phase can be retrieved

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	VALUE	Double precision	Return the value of extra contribution.
	IWSG	Integer array	Workspace.

4.3 Condition, Stream, and Segment Subroutines

4.3.1 TQSETC(STAVAR,INDEXP,INDEXC,VAL,NUMCON,IWSG)

Full name: Set Condition.

Purpose: To set conditions for an equilibrium calculation.

Arguments:	Name	Type	Value set on call or returned
	STAVAR	Character*8	Set as a state variable listed in Table 10.
	INDEXP	Integer	Set as a phase index (if needed).
	INDEXC	Integer	Set as a component or constituent index (if needed).
	VAL	Double precision	Set to the value.
	NUMCON	Integer	Returned as an identification of the condition.
	IWSG	Integer array	Workspace.

Comments: In STAVAR the mnemonic of the state variable must be given, see Table 10. In some cases just the mnemonic is needed, like for temperature or pressure, but in many cases a phase index or a component index must be used to specify the condition. If both a phase index and a constituent index is supplied the condition will be set for the specified constituent in the specified phase.

The application program must set exactly the same number of conditions as degrees of freedom in the defined system. The degrees of freedom are equal to the number of system components plus two (usually temperature and pressure). Setting a phase **FIXED** using TQCSP decrease the degrees of freedom in the system by 1. Resetting the phase **ENTERED** using TQCSP will restore one degree of freedom.

Possible combinations of STAVAR and indices are listed in Table 10.

In the table it is shown that the same value of STAVAR may be used with or without index. In the case there should not be an index, the value of INDEXP or INDEXC must be negative.

Some combination of conditions may be thermodynamically impossible. The TQ interface will make its best to provide relevant help for such cases.

Examples: Set the temperature to 800 Celsius

```
CALL TQSSU('Temperature','C',IWSG)
CALL TQSETC('T',-1,-1,800.0D0,NCOND,IWSG)
```

Set the incoming amount of a liquid phase constituent named Al₂O₃ to 1.5 moles

```
CALL TQGPI(INDEXP,'LIQUID',IWSG)
CALL TQGPCI(INDEXP,INDEXC,'AL2O3',IWSG)
CALL TQSETC('IN',INDEXP,INDEXC,1.5D0,NCOND,IWSG)
```

Set the mass percent of the system component Cr to 13%.

```
CALL TQGSCI(INDEX,'cr',IWSG)
CALL TQSETC('W%',-1,INDEX,13.0D0,NCOND,IWSG)
```

Set the total amount of system to 1.0 mole components

```
CALL TQSETC('N',-1,-1,1.0D0,NCOND,IWSG)
```

Set the mole fraction of H₂O in GAS to 5 mol percent

```
CALL TQGPI(INDEXP,'GAS',IWSG)
```

```
CALL TQGPCI (INDEXP, INDEXC, 'H2O1', IWSG)
CALL TQSETC ('X', INDEXP, INDEXC, 0.05D0, NCOND, IWSG)
```

4.3.2 TQREMC(NUMCON,IWSG)

Full name: Remove Condition.

Purpose: Remove the condition numbered NUMCON.

Arguments:	Name	Type	Value set on call or returned
	NUMCON	Integer	Set to a condition number.
	IWSG	Integer array	Workspace.

Comments: TQSETC and TQCSTM return an index for each condition set. This value must be supplied in this call. In order to change a condition to something else, not just a new value, one must first remove the condition. If one just wants to change the value of a condition one may call TQSETC again instead.

4.3.3 TQSCURC(IWSG)

Full name: Save Current Conditions.

Purpose: Save all conditions in case they need to be restored.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: The saved conditions can be restored if necessary by using TQRESTC.

4.3.4 TQREMAC(IWSG)

Full name: Remove All Conditions.

Purpose: Remove all conditions.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: TQREMAC provides the easiest way to remove all conditions. After calling TQREMAC, one can set completely new or restore previously saved conditions.

4.3.5 TQRESTC(IWSG)

Full name: Restore Condition.

Purpose: Restore saved conditions.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: Before calling TQRESTC, one needs to remove all present conditions.

4.3.6 TQCSTM(IDENT,TEMP,PRESS,IWSG)

Full name: Create Stream.

Purpose: To set the system conditions by stream input. Stream calculations are useful when calculating differences between an initial state and a final state. The streams define the initial state of the system components by specifying reactants of different phases at given temperatures and pressures.

Arguments:	Name	Type	Value set on call or returned
	IDENT	Character*24	Set as identifier of the stream.
	TEMP	Double precision	Input temperature of stream.
	PRESS	Double precision	Input pressure of stream.
	IWSG	Integer array	Workspace.

Comments: A stream is a non-reacting media for transferring matter to a reaction zone. A stream may contain several phases at the same given temperature and pressure. Phases with different temperatures and pressures should be grouped into different streams. Several streams can be transferred to a reaction zone. The input constituents of each phase do not react in a stream.

4.3.7 TQSSC(IDENT,INDEXP,INDEXC,VALUE,NUMIN,IWSG)

Full name: Set Stream Constituent Amount.

Purpose: Set the amount of phase constituent in a stream.

Arguments:	Name	Type	Value set on call or returned
	IDENT	Character*24	Set as identifier of the stream.
	INDEXP	Integer	Set as a phase index
	INDEXC	Integer	Set as a constituent index.
	VALUE	Double precision	Set to an amount of the constituent INDEXC in the stream.
	NUMIN	Integer	Returned as identification of the input constituent in the stream.
	IWSG	Integer array	Workspace.

Comments: The last one will take effect if the amount of the same phase constituent have been set several times, i.e., the amount can not be set additively.

4.3.8 TQSSIC(STAVAR,VALUE,IWSG)

Full name: Set Stream Invariant State Variable.

Purpose: To specify the invariant state variable for calculating the reaction of all streams.

Arguments:	Name	Type	Value set on call or returned
	STAVAR	Character*8	Set as the mnemonic of a state variable.
	VALUE	Double precision	Set to change in value of STAVAR.
	IWSG	Integer array	Workspace.

Comments: The state variables that could be used are G, H, S, and V with a suffix D, which means difference between initial and final states of the reaction.

Examples: Calculation of adiabatic temperature for knallgas.

```
DIMENSION TPA(2)
...
C...set input temperature and pressure
TEMP=298.15D0
PRES=1.0D5
C...create the stream
CALL TQCSTM('knallgas',TEMP,PRES,IWSG)
C...set amount of H2 and O2 in the stream
CALL TQGPCI(1,INDEXC,'H2',IWSG)
CALL TQSSC('knallgas',1,INDEXC,2.0D0,NUMIN,IWSG)
CALL TQGPCI(1,INDEXC,'O2',IWSG)
CALL TQSSC('knallgas',1,INDEXC,1.0D0,NUMIN,IWSG)
C...set the global temperature and pressure for the reaction
CALL TQSETC('T',-1,-1,500.0D+0,NUMC,IWSG)
CALL TQSETC('P',-1,-1,PRES,NUMC,IWSG)
C...get the enthalpy of reaction
CALL TQCE(' ',-1,-1,0.0D+0,IWSG)
CALL TQGETV1('HD',-1,-1,ENT,IWSG)
WRITE(*,*)'Calculated enthalpy of reaction are '
&, ENT, ' at 500 K. '
C...set that the enthalpy shall be constant in the calculation
CALL TQSSIC('HD',0.0D0,IWSG)
C...calculate
CALL TQCE('T',-1,-1,1.0D+0,IWSG)
C...get temperature
CALL TQGETV1('T',-1,-1,TEMP,IWSG)
WRITE(*,*)'Calculated temperature ',TEMP
```

4.3.9 TQDSTM(IDENT,IWSG)

Full name: Delete Stream.

Purpose: Delete all or a stream.

Arguments:	Name	Type	Value set on call or returned
	IDENT	Character*24	Set as identifier of the stream.
	IWSG	Integer array	Workspace.

Comments: Using an empty string as IDENT will remove all the streams entered.

4.3.10 TQNSEG(ID,IWSG)

Full name: New Equilibrium Segment.

Purpose: With this subroutine application program can create a new equilibrium description with the same thermodynamic data. This subroutine is useful when simulating several equilibria representing local conditions, for example, in the reactor simulator.

Arguments:	Name	Type	Value set on call or returned
	ID	Character*24	Set as identifier of the equilibrium segment.
	IWSG	Integer array	Workspace.

Comments: TQNSEG does not read any thermodynamic file. This must have already been done with TQRFIL.

4.3.11 TQSSEG(ID,IWSG)

Full name: Select Equilibrium.

Purpose: When the application program has created several equilibrium segments using TQNSEG, this subroutine makes it possible to select a current equilibria which the subroutine calls refer to.

Arguments:	Name	Type	Value set on call or returned
	ID	Character*24	Set to an equilibrium identification.
	IWSG	Integer array	Workspace.

4.4 Calculation and Results Subroutines

4.4.1 TQCE(TARGET,INDEXP,INDEXC,VALUE,IWSG)

Full name: Calculate Equilibrium.

Purpose: Calculate the equilibrium with current settings of conditions or streams.

Arguments:	Name	Type	Value set on call or returned
	TARGET	Character*8	Set to a state variable, if necessary.
	INDEXP	Integer	Set to a phase index, if necessary.
	INDEXC	Integer	Set to a component index, if necessary.
	VALUE	Double precision	Set to an estimate of the target variable.
	IWSG	Integer array	Workspace.

Comments: Some software may need a TARGET specified for certain types of calculations. A TARGET is a state variable as specified in TQSETC or Table 10. When working with Thermo-Calc, it is only useful in stream reaction calculations, where an initial guess of the target variable may be of some help. Otherwise, TARGET is normally set as an empty string and the values of INDEXP, INDEXC, and VALUE are irrelevant.

Examples: Calculate enthalpy for an equilibrium gas mixture SO₃, SO₂ and O₂. Input SO₃ 2%, O₂ 10% and 88% SO₂.

```
...
CALL TQGPI('GAS',INDEXP,IWSG)
C...set temperature, pressure and total amount of moles
CALL TQSETC('T',-1,-1,800.0D0,NCOND,IWSG)
CALL TQSETC('P',-1,-1,1.0D5,NCOND,IWSG)
CALL TQSETC('N',-1,-1,1.0D0,NCOND,IWSG)
C...set mole fraction of SO2 and O2
CALL TQGPI(INDEXP,'GAS',IWSG)
CALL TQGPCI(INDEXP,INDEXC,'SO2',IWSG)
CALL TQSETC('IN',INDEXP,INDEXC,8.8D-1,NCOND,IWSG)
CALL TQGPCI(INDEXP,INDEXC,'O2',IWSG)
CALL TQSETC('IN',INDEXP,INDEXC,1.0D-1,NCOND,IWSG)
CALL TQGPCI(INDEXP,INDEXC,'SO3',IWSG)
CALL TQSETC('IN',INDEXP,INDEXC,2.0D-2,NCOND,IWSG)
CALL TQCE(' ',0,0,0.0D+0,IWSG)
CALL TQGETV1('H',-1,-1,ENT,IWSG)
...
```

NOTE: In this way a application program can calculate the incoming enthalpy into the system. If there is more than one incoming flow it can calculate the enthalpies for each flow and sum them up.

4.4.2 TQCEG(IWSG)

Full name: Calculate Equilibrium Global.

Purpose: Calculate Equilibrium using Global Minimization Algorithm.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: The use of global minimization algorithm is meant to avoid metastable or unstable equilibrium and to obtain truly stable equilibrium. This is mainly due to its ability to find automatically miscibility gap and create accordingly new composition sets. As a consequence, the number of phases may increase after calling TQCEG. The newly added phases (new composition sets of old phases) are always put in the end of the phase list. In this way, the indexes of old phases remain the same as before (See Example 13). The global minimization technique starts with discretizing the composition space and calculating Gibbs energy values at each grid point for each phase at a given temperature. This usually leads to a significant increase of computation time. Therefore, it is not recommended to use TQCEG in time-critical application programs. In the cases where phases involved are well known, for example, identifying the local equilibrium at a phase interface, it is absolutely not necessary to use TQCEG. If TQCEG is needed, irrelevant phases should better be rejected in the beginning when fetching thermodynamic data from a database.

4.4.3 TQGETV(STAVAR,INDEXP,INDEXC,NUMBER,VALAR,IWSG)

4.4.4 TQGET1(STAVAR,INDEXP,INDEXC,VAL,IWSG)

Full name: Get Values.

Purpose: These subroutines return the value of any variable in the system after an equilibrium calculation, for example,

- thermodynamic properties for phases and constituents.
- temperature, pressure and volume of the system.
- amount of the system, a phase or a constituent.

With TQGETV an array of values can be returned; with TQGET1 a single value only.

Arguments:	Name	Type	Value set on call or returned
	STAVAR	Character*32	Set to mnemonic of state variable
	INDEXP	Integer	Set to a phase index
	INDEXC	Integer	Set to a component or constituent index
	NUMBER	Integer	Set to the number of values in VALAR.
	VALAR	Double precision array	Return the values
	VAL	Double precision	Return the value
	IWSG	Integer array	Workspace.

Comments: If an equilibrium is not established, the error code will be set on return. In Table 12 an extended set of state variables is given, which can be used for obtaining values.

Examples:

```

Get temperature of the system
  CALL TQGET1 ('T', -1, -1, VAL, IWSG)
Get overall mole fraction of system component Cr
  CALL TQGSCI (INDEXC, 'CR', IWSG)
  CALL TQGET1 ('X', -1, INDEXC, VAL, IWSG)
Get overall mole fractions of all components
  CALL TQGETV ('x', -1, 0, NCOM, VALAR, IWSG)
Get activity of system component SiC
  CALL TQGSCI (INDEXC, 'sic', IWSG)
  CALL TQGET1 ('AC', -1, INDEXC, VAL, IWSG)
Get activity of gas phase constituent SiC (gas is phase 1)
  CALL TQGPCI (1, INDEXC, 'sic', IWSG)
  CALL TQGET1 ('AC', 1, INDEXC, VAL, IWSG)
Get total mass of system
  CALL TQGET1 ('M', -1, -1, VAL, IWSG)

```

```

    or CALL TQGET1 ('M', 0, 0, VAL, IWSG)
Get total mass of liquid phase
    CALL TQGPI (INDEXP, 'LIQUID', IWSG)
    CALL TQGET1 ('MP', INDEXP, -1, VAL, IWSG)
Get mass of all constituents of liquid phase
    CALL TQGPI (INDEXP, 'LIQUID', IWSG)
    CALL TQGETV ('IM', INDEXP, 0, NVAL, VALAR, IWSG)
Get mass of SIC in liquid phase
    CALL TQGPI (INDEXP, 'LIQUID', IWSG)
    CALL TQGPCI (INDEXP, INDEXC, 'sic', IWSG)
    CALL TQGET1 ('IM', INDEXP, INDEXC, VAL, IWSG)
Get volume of GAS phase
    CALL TQGET1 ('V', 1, -1, VAL, IWSG)
Get constituent mole fraction of H2O in GAS
    CALL TQGPCI (1, INDEXC, 'h2o', IWSG)
    CALL TQGET1 ('Y', 1, INDEXC, VAL, IWSG)
Get partial pressure of H2O in GAS (equal to the total pressure times the constituent
mole fraction)
    CALL TQGPCI (1, INDEXC, 'h2o', IWSG)
    CALL TQGET1 ('Y', 1, INDEXC, VAL, IWSG)
    CALL TQGET1 ('p', -1, -1, PVAL, IWSG)
    PH2O = PVAL*VAL
Get chemical potentials of all constituents in slag
    CALL TQGPI (INDEXP, 'slag', IWSG)
    CALL TQGETV ('MUC', INDEXP, 0, NCON, VALAR, IWSG)

```

4.4.5 DOUBLE PRECISION FUNCTION TQGMU (INDEXC,IWSG)

Full name: Get Chemical Potential.

Purpose: This function returns the chemical potential of a component in a faster way.

Arguments:	Name	Type	Value set on call or returned
	INDEXC	Integer	Set to a component index
	IWSG	Integer array	Workspace.

4.4.6 DOUBLE PRECISION FUNCTION TQGGM (INDEXP,IWSG)

Full name: Get Molar Gibbs Energy.

Purpose: This function returns the molar Gibbs energy of a phase in a faster way.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index
	IWSG	Integer array	Workspace.

4.4.7 TQGPD (INDEXP,NSUB,NSCON,SITES,YFRAC,EXTRA,IWSG)

Full name: Get Phase Data.

Purpose: With this subroutine the application program can get data for the constituents of a phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index
	NSUB	Integer	Return the number of sublattices.
	NSCON	Integer array	Return the number of constituents on each sublattice.
	SITES	Double precision array	Return the number of sites on each sublattice.
	YFRAC	Double precision array	Return the fractions of the constituents.
	EXTRA	Double precision array	Return some special values (see Comments)
	IWSG	Integer array	Workspace.

Comments: With this subroutine the application program can determine the structure of the phase and the fraction of the constituents and other things. Note that YFRAC is constituent fraction, not mole fractions.

A substitutional phase will have NSUB equal to 1, which is identical to no sublattice. That is true for the gas phase too. The maximum number of sublattices are 10.

The constituents of a phase are numbered sequentially from 1 for the first constituent on the first sublattice, to NPCON (See TQGNPC) for the last constituent on the last sublattice. NSCON(L) is the number of constituents on sublattice L. The sum of NSCON over all sublattices is equal to NPCON. Note that constituents that are **DORMANT** and **SUSPENDED** still are counted in NPCON and NSCON. They also have a fraction in YFRAC (which must be zero of course).

EXTRA may contain extra information about the phase, total mass for example. These are yet to be defined.

Examples: To list the constituent names and fractions by sublattices. It is assumed that there are max 10 sublattices and max 500 constituents on all sublattices all together.

```

        DIMENSION NSCON(10),SITES(10),YFRAC(500),EXTRA(5)
        CHARACTER NAME*24
        LOGICAL TQGSPC
        ...
        CALL TQGPN(INDEXP,NAME,IWSG)
        CALL TQGPD(INDEXP,NSUB,NSCON,SITES,YFRAC,EXTRA,
&IWSG)
        KK=0
        WRITE(*,190)NAME,NSUB
190     FORMAT(' The phase ',A,' has ',I2,' sublattices')
        DO 300 LS=1,NSUB
            WRITE(*,191)LS,SITES(LS),NSCON(LS)
191     FORMAT('On sublattice ',I2,' there are ',F8.4,
&' sites and',I3,' constituents')
            DO 200 LC=1,NSCON(LS)
                KK=KK+1
                CALL TQGPCN(INDEXP,KK,NAME,IWSG)
                WRITE(*,192)NAME,YFRAC(KK)
192     FORMAT('Constituent ',A,' has fraction',

```

```

&1P1E15.8)
200      CONTINUE
300      CONTINUE

```

4.4.8 TQGDF (IMATR,IPREC,NPH,NCOM,XMATR,XPREC,TEMP,DF,IWSG)

Full name: Get the driving force of phase transformation

Purpose: With this subroutine the application program can get data on the driving force for a phase transformation

Arguments:	Name	Type	Value set on call or returned
	IMATR	Integer	Set index of matrix phase
	IPREC	Integer	Set index of precipitate phase
	NPH	Integer	Set total number of phases in the system
	NCOM	Integer	Set total number of components in the system
	XMATR	Double precision array	Set composition (in mole-fraction) array of the matrix phase
	XPREC	Double precision array	Set composition (in mole-fraction) array of the precipitate phase
	TEMP	Double precision	Set temperature in Kelvin
	DF	Double precision	Return driving force in J/mol
	IWSG	Integer array	Workspace.

Comments: XPREC can be inputs or outputs, depending on whether its values are known before the calculation or not. If unknown, the values of XPREC should be set to zero or negative when calling this subroutine and on return one obtains the composition of the precipitate at which the maximum driving force is available.

4.4.9 TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG)

Full name: Get the driving force of nucleation and local equilibrium concentration for a phase transformation under para- or ortho-equilibrium condition.

Purpose: With this subroutine the application program can obtain data on both the chemical driving force for the nucleation of a precipitate and the local equilibrium concentration at the matrix/precipitate interface under para- or ortho-equilibrium conditions.

Arguments:	Name	Type	Value set on call or returned
	MODE	Integer	Set type of output and type of composition to use (± 1 , ± 2 , and ± 3 correspond to mole fraction, weight fraction, and U-fraction respectively. Obviously, ± 3 has nothing to do with para-equilibrium calculations. If negative, calculate and output only driving force data. This will save the time for

		equilibrium calculation when you are not interested in local equilibrium concentrations)
IMATR	Integer	Set index of matrix phase
IPREC	Integer	Set index of precipitate phase
NIE	Integer	Set number of interstitial element(s). Zero implies no para-equilibrium calculation
IIE	Integer array	Set index of interstitial element(s), only relevant for para-equilibrium condition
XMATR	Double precision array	Set composition of matrix phase. Composition type depends on MODE
TEMP	Double precision	Set temperature in Kelvin
DF	Double precision	Return driving force in J/mol
XPREC	Double precision array	Return composition of the precipitate phase at the maximum driving force under para/ortho-equilibrium condition or set to a known composition of the precipitate in order to get the driving force of phase transformation. Composition type depends on MODE
XEM	Double precision array	Return, if both MODE and DF are positive, local equilibrium composition of matrix phase. Composition type depends on MODE
XEP	Double precision array	Return, if both MODE and DF positive, local equilibrium composition of precipitate phase. Composition type depends on MODE
MUI	Double precision array	Return, if both MODE and DF are positive, chemical potential of interstitial elements. Relevant for only para-equilibrium calculation.
IWSG	Integer array	Workspace.

Comments: For ortho-equilibrium calculation, XPREC can be inputs or outputs, depending on whether its values are known before the calculation or not. If unknown, the values of XPREC should be set to zero or negative when calling this subroutine and on return one obtains the composition of the precipitate at which the maximum driving force is available. The use of this subroutine for the ortho-equilibrium calculation supercedes that of the preceding subroutine TQGDF. A demonstration of this subroutine can be found in Example 11.

4.5 Troubleshooting Subroutines

4.5.1 TQLS(IWSG)

Full name: List Status.

Purpose: Listing status of all components, phases, and species in a system.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.

Comments: If necessary, use this subroutine to check if the status of all components, phases, and species has been correctly set in an application program. It should only be used for debugging purpose.

4.5.2 TQLC(IWSG)

Full name: List Conditions.

Purpose: Listing conditions set for the current equilibrium calculation.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.

Comments: If necessary, use this subroutine to check if the conditions for an equilibrium calculation in the application program has been correctly set. It should only be used for debugging purpose.

4.5.3 TQLE(IWSG)

Full name: List Equilibrium.

Purpose: Listing results from the most recent equilibrium calculation. The output will depend on the package used and the listing will appear on the current output unit.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.

Comments: If necessary, use this subroutine to check if an equilibrium calculation is successful. It should only be used for debugging purpose.

4.5.4 TQFASV(IWSG)

Full name: Force Automatic Start Value.

Purpose: To force automatic start-values for all phases in a single equilibrium calculation.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: It is not necessary unless the calculation fails.

4.5.5 TQSDMC(INDEXP,IWSG)

Full name: Set Default Major Constituents.

Purpose: To set the major phase constituents to the default ones defined in the thermodynamic data file.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set as a phase index
	IWSG	Integer array	Workspace.

Comments: Major constituents in a phase can be set in the GES module of Thermo-Calc and then saved into a thermodynamic data file for the use of this interface.

4.5.6 TQSSPC(INDEXP,YF,IWSG)

Full name: Set Start Phase Constitution.

Purpose: To set start-values for the constitution of an individual phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set as a phase index
	YF	Double precision array	Set to the site fraction of each constituent.
	IWSG	Integer array	Workspace.

Comments: It is not necessary unless the calculation fails, especially when involving a miscibility gap or an ordering phase.

4.5.7 TQSSV(STAVAR,IP,IC,VALUE,IWSG)

Full name: Set Start Variable.

Purpose: To set start-value for a state variable.

Arguments:	Name	Type	Value set on call or returned
	STAVAR	Character*8	Set as a state variable listed in Table 10.
	IP	Integer	Set as a phase index (if needed).
	IC	Integer	Set as a component or constituent index (if needed).
	VALUE	Double precision	Set to the value.
	IWSG	Integer array	Workspace.

Comments: It is not necessary unless the calculation fails.

4.5.8 TQPINI(IWSG)

Full name: Poly-3 reINItiation.

Purpose: Reinitiate the Poly-3 workspace in Thermo-Calc kernel.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: Preparing for a fresh calculation.

4.5.9 TQSNL(MAXIT,ACC,YMIN,ADG,IWSG)

Full name: Set Numerical Limits

Purpose: To set the Numerical Limits to be used inside Poly-3.

Arguments:	Name	Type	Value set on call or returned
	MAXIT	Double precision	Set maximum number of iterations when calculating equilibrium. Default value is 500.
	ACC	Double precision	Set required relative accuracy when calculating equilibrium. Default value is 1E-6.
	YMIN	Double precision	Set smallest fraction to assign to unstable constituents. Default value is 1E-30.
	ADG	Character*1	Specify if the calculation should be forced to converge also for the meta stable phases. Legal options are Y or N, where Y means yes and N means no. N is default.
	IWSG	Integer array	Workspace.

Comments: It is not necessary unless the calculation fails.

4.5.10 TQSMNG(NGP,IWSG)

Full name: Set Maximum Number of Grid points for each phase.

Purpose: To change the maximum number of grid points that can be used for each phase.

Arguments:	Name	Type	Value set on call or returned
	NGP	Integer array	Number of grid points.
	IWSG	Integer array	Workspace.

Comments: The global minimization technique starts with discretizing the composition space and calculating Gibbs energy values at each grid point for each phase. To balance its efficiency and robustness, an appropriate density of grid points should be chosen. The default value of NGP is 20000. In practice, the number of grid points generated during a normal calculation is much less than this value. However, under certain circumstances, one does need to increase the density of grid point for some phases in order to find a true stable equilibrium.

4.5.11 ST1ERR(IERR,SUBR,MESS)

Full name: Set Error Code and Give Message.

Purpose: This is called when an error that cannot be handled by the current program unit occurs. The error message is printed on the error unit but also saved internally in the error handling package. The program unit should return to the calling program.

Arguments:	Name	Type	Value set on call or returned
	IERR	Integer	Set to an error code.
	SUBR	Character*6	Set to the current subroutine name.
	MESS	Character*72	Set to the error message to be printed.

Comments: The error-handling routines are those defined by SGTE for use in the thermodynamic model package. Note that the error-handling is constructed in such a way that when a subroutine detects an error it cannot handle, it should first call an ST* subroutine to set an appropriate error code and then return to the calling subroutine. In that subroutine the error code should be tested, and possibly that subroutine can correct the error and proceed, otherwise it should return to its calling subroutine and so on, until either the error is corrected or the top level of the program is reached.

In this way it is possible to design a program where minor problems at a low level do not cause program to terminate. Instead, the error will be passed up to a higher level where it can be corrected or ignored. The normal subroutines to use are ST2ERR to set the error code and SG2ERR to check it. The other subroutines are less used.

NOTE: The TQ subroutines will normally not clear the error code when they are called. An error set in an earlier subroutine but not tested and detected after that call may cause strange error messages later on. This should be used only for fatal or almost fatal errors.

4.5.12 ST2ERR(IERR,SUBR,MESS)

Full name: Set Error Code.

Purpose: This is called when an error occurs that cannot be handled by the current program unit. The program unit should return to the calling program.

Arguments:	Name	Type	Value set on call or returned
	IERR	Integer	Set to an error code.
	SUBR	Character*6	Set to the current subroutine name.
	MESS	Character*72	Set to the error message to be printed.

Comments: Identical to ST1ERR except that it is silent, i.e., no error message is printed. This should be the normal subroutine to call when detecting errors that should be handled by a higher level of the program.

4.5.13 LOGICAL FUNCTION SG1ERR or TQG1ERR(IERR)

Full name: Get Error Code and Give Message.

Purpose: This is a logical function which could be called after calling a TQ subroutine that can detect an error when the error message should be displayed. If there is an error the function value is .TRUE. and the appropriate error code is in IERR. This subroutine will also print the error message on the error unit.

Arguments:	Name	Type	Value set on call or returned
	IERR	Integer	Set to the error code.

Comments: This should be used when the error is almost fatal. Note that it is possible that the error message is already printed by ST1ERR. Use SG2ERR in most cases. If no error the function value is .FALSE. and IERR is zero.

4.5.14 LOGICAL FUNCTION SG2ERR or TQG2ERR(IERR)

Full name: Get Error Code.

Purpose: This is a logical function which should be called after calling any TQ subroutine that can detect an error. If there is an error the function value is .TRUE. and the appropriate error code is in IERR. This subroutine will not print the error message.

Arguments:	Name	Type	Value set on call or returned
	IERR	Integer	Set to the error code.

Comments: This should be used for the normal error checking. Note that it is possible that the error message has already been printed by ST1ERR. The program may be able to handle the error to pass it on upwards. If no error the function value is .FALSE. and

IERR is zero.

Examples:

```
LOGICAL SG2ERR
...
CALL TQCE(' ',IWSG)
IF(SG2ERR(IERR)) GOTO 900
...
900 RETURN
```

4.5.15 LOGICAL FUNCTION SG3ERR or TQG3ERR(IERR,SUBR,MESS)

Full name: Get Error Code and Message.

Purpose: This is a logical function which could be called after calling any TQ subroutine that can detect an error. If there is an error the function value is .TRUE. and the appropriate error code is in IERR, the subroutine that detected the error in SUBR and the message in MESS. No printing on the error unit. This is useful if the calling program wants to print the message itself in an appropriate context.

Arguments:	Name	Type	Value set on call or returned
	IERR	Integer	Return the error code.
	SUBR	Character*6	Return the name of the subroutine detecting an error.
	MESS	Character*72	Return the error message.

Comments: This should be used when the error testing subroutine wants to handle the printing of the error message itself. It is possible that the error message has already been printed by ST1ERR.

4.5.16 RESERR (or TQRSERR)

Full name: Reset Error Code and Message

Purpose: This subroutine will reset the error code. A subsequent call to the SG* functions will give no error.

Arguments: None

Comments: This should be used when the error has been cleared so that execution can continue. Unless the error code is cleared by this subroutine the SG* functions will continue to report the same error.

4.6 Extra Subroutines

4.6.1 TQGMA(INDEXP,TP,YF,VAL,IWSG)

4.6.2 TQGMB(INDEXP,TP,VAL,IWSG)

4.6.3 TQGMC(INDEXP,VAL,IWSG)

Full name: Get Gibbs Energy – Method A, B, and C.

Purpose: Getting Gibbs energy of a phase if temperature, pressure, and site fractions are given as arguments or by other subroutines shown below in this Section.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	TP	Double precision array	Set to temperature and pressure values.
	YF	Double precision array	Set to site fraction values in the index order of phase constituent.
	VAL	Double precision	Return Gibbs energy value.
	IWSG	Integer array	Workspace.

Comments: The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy of a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.

4.6.4 TQGM DY(INDEXP,VARR,IWSG)

Full name: Get Gibbs Energy and its partial Derivative w.r.t. y-fraction.

Purpose: Getting Gibbs energy and its 1st partial derivatives with respect to site fractions for a phase if temperature, pressure, and site fractions have been given by other subroutines shown below in this Section.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	VARR	Double precision array	Return values of Gibbs energy and its 1 st partial derivatives w.r.t. site fractions in the index order of phase constituents.
	IWSG	Integer array	Workspace.

Comments: The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy and its 1st partial derivative w.r.t site fractions for a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest. This subroutine is demonstrated in Example 9

4.6.5 TQGMOB(INDEXP,ISP,VAL,IWSG)

Full name: Get Mobility

Purpose: Getting mobility of a species in a phase with the temperature, pressure, and site fractions given by other subroutines shown below in this Section.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	ISP	Integer	Set to a system species index
	VAL	Double precision	Return species or atomic mobility value.
	IWSG	Integer array	Workspace.

Comments: Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the atomic or species mobility in a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest. The use of this subroutine is demonstrated in Example 9.

4.6.6 TQSTP(TP,IWSG)

Full name: Set Temperature and Pressure

Purpose: Setting temperature and pressure.

Arguments:	Name	Type	Value set on call or returned
	TP	Double precision array	Set temperature and pressure.
	IWSG	Integer array	Workspace.

Comments: This subroutine is used before calling TQGMC, TQGMDY, TQDGY, and TQGMOB. See Example 9.

4.6.7 TQSYF(INDEXP,YF,IWSG)

Full name: Set Site Fractions

Purpose: Setting site fractions for a phase.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	YF	Double precision array	Set to site fraction values in the index order of the phase constituents.
	IWSG	Integer array	Workspace.

Comments: This subroutine is used before calling TQGMB, TQGMC, TQGMDY, TQDGY, and TQGMOB. See Example 9.

4.6.8 TQGSSPI(SPN,ISP,IWSG)

Full name: Get System Species Index

Purpose: Getting index of a system species with given name.

Arguments:	Name	Type	Value set on call or returned
	SPN	Character*24	Set to a system species name.
	ISP	Integer	Return index value of the system species
	IWSG	Integer array	Workspace.

Comments: Useful if one want to use TQGMOB. See Example 9.

4.6.9 TQCMOBA(INDEXP,ISP,IWSG)

4.6.10 TQCMOBB(INDEXP,IWSG)

Full name: Check if Mobility data available – Method A and B

Purpose: Check if mobility data have been appended into thermodynamic data file.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index
	ISP	Integer	Set to a system species index
	IWSG	Integer array	Workspace.

Comments: No comments.

4.6.11 TQDGYI(INDEXP,VARR1,VARR2,IWSG)

Full name: Get Gibbs Energy and its 1st and 2nd Partial Derivative w.r.t. site-fractions.

Purpose: Getting Gibbs energy and its 1st and 2nd partial derivatives with respect to site fractions for a phase if temperature, pressure, and site fractions have been given by other subroutines shown below in this Section.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	VARR1	Double precision array	Return values of Gibbs energy and its 1 st partial derivatives w.r.t. site fractions in the index order of phase constituents.
	VARR2	Double precision array	Return values of 2 nd partial derivatives of Gibbs energy w.r.t. site fractions in the index order of IR: IR=J+I*(I-1)/2, I>=J;

$IR=I+J*(J-1)/2$, $I<J$, where I and J are row and column indexes of phase constituents, respectively.
Workspace.

IWSG Integer array

Comments: The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy and its 1st and 2nd partial derivatives w.r.t site fractions for a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.

4.6.12 TQGPHP(INDEXP,NE,NCNV,NC,IWORK,WORK,IWSG)

Full name: Get phase constitution properties.

Purpose: Getting phase constitution properties such as number of components, number of constituents, number of constituents without counting vacancies, etc.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	NE	Integer	Return number of components
	NCNV	Integer	Return number of constituents without counting vacancies
	NC	Integer	Return number of constituents
	IWORK	Integer array	Return values needed in X to Y conversion, array size $\geq 4*NCNV$
	WORK	Double precision array	Return values needed in X to Y conversion, array size $\geq (NE+1)*NCNV$
	IWSG	Integer array	Workspace.

Comments: This subroutine is designed to speed up conversions of quantities involving mole fractions and site fractions in dynamic calculations where such operations are needed at each local time and space grid point. For each phase involved, one call of this subroutine is enough for subsequent conversions concerning this phases. See Example 10.

4.6.13 TQX2Y(INDEXP,NE,NCNV,NC,IWORK,WORK,XF,YF,IWSG)

Full name: Get Y-fraction given X-fraction.

Purpose: Converting mole fractions to site fractions in a phase without internal degree of freedom.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	NE	Integer	Set to number of components
	NCNV	Integer	Set to number of constituents without counting vacancies
	NC	Integer	Set to number of constituents
	IWORK	Integer array	Set to values needed in X to Y conversion
	WORK	Double precision array	Set to values needed in X to Y conversion
	XF	Double precision array	Set to mole fractions

YF	Double precision array	Return site fractions
IWSG	Integer array	Workspace.

Comments: This subroutine uses the phase constitution properties obtained by TQGPHP as input. See Example 10.

4.6.14 TQGMDX(IP, NE, NCV, NC, IWORK, WORK, YF, VARR, GM, DGDX, XF, IWSG)

Full name: Get Gibbs energy and its partial derivative w.r.t. X-fraction.

Purpose: Converting Gibbs energy and its 1st partial derivatives with respect to site fractions (VARR obtained by calling TQGM DY) to that w.r.t mole fractions for a phase.

Arguments:	Name	Type	Value set on call or returned
	IP	Integer	Set to a phase index.
	NE	Integer	Set to number of components
	NCNV	Integer	Set to number of constituents without counting vacancies
	NC	Integer	Set to number of constituents
	IWORK	Integer array	Set to values needed in X to Y conversion
	WORK	Double precision array	Set to values needed in X to Y conversion
	YF	Double precision array	Set to site fractions
	VARR	Double precision array	Set to Gibbs energy and its first derivative with respect to site fractions
	GM	Double precision	Return Gibbs energy
	DGDX	Double precision array	Return Gibbs energy and its first derivative with respect to mole fractions
	XF	Double precision array	Return mole fractions
	IWSG	Integer array	Workspace.

Comments: This subroutine uses the phase constitution properties obtained by TQGPHP as input. Note VARR obtained by calling TQGM DY is in unit of J/mole of formula unit. GM and DGDX in the present subroutine is in unit of J/mole of atoms. For the use of this subroutine together with TQGPHP and TQX2Y, please see Example 10.

4.7 Database Subroutines (See Example 12)

4.7.1 TQGDBN(DB_ARR,N,IWSG)

Full name: Get DataBase Names and Number.

Purpose: Get the names and total number of thermodynamic and kinetic databases listed in the database initiation file of Thermo-Calc: TC_INITD.TDB.

Arguments:	Name	Type	Value set on call or returned
	DB_ARR	Character*24 array	Return database names.
	N	Integer	Return total number of databases available.
	IWSG	Integer array	Workspace.

Comments: IERR = 1001 Failed to find the initiation file.

4.7.2 TQOPDB(TDB,IWSG)

Full name: Open DataBase.

Purpose: Open a thermodynamic or kinetic database.

Arguments:	Name	Type	Value set on call or returned
	TDB	Character*256	Set to the name of a database.
	IWSG	Integer array	Workspace.

Comments: IERR = 1001 Failed to find the initiation file.
IERR = 1002 Database or its license not available.

4.7.3 TQLIDE(EL_ARR,N,IWSG)

Full name: List Database Elements.

Purpose: List all elements available in the chosen database.

Arguments:	Name	Type	Value set on call or returned
	EL_ARR	Character*2 array	Return the names of all elements.
	N	Integer	Return the total number of elements.
	IWSG	Integer array	Workspace.

Comments:

4.7.4 TQAPDB(TDB,IWSG)

Full name: Append DataBase.

Purpose: Append a thermodynamic or kinetic database.

Arguments:	Name	Type	Value set on call or returned
	TDB	Character*256	Set to the name of a database.
	IWSG	Integer array	Workspace.

Comments: IERR = 1002 Database or its license not available.

4.7.5 TQDEFEL(ELNAM,IWSG)

Full name: Define Element.

Purpose: Define a system element.

Arguments:	Name	Type	Value set on call or returned
	ELNAM	Character*2	Set to the name of an element.
	IWSG	Integer array	Workspace.

Comments: IERR = 1011 Element not included in the chosen database.
IERR = 1012 Element already defined.

4.7.6 TQREJEL(ELNAM,IWSG)

Full name: Reject Element.

Purpose: Reject a defined system element.

Arguments:	Name	Type	Value set on call or returned
	ELNAM	Character*2	Set to the name of an element.
	IWSG	Integer array	Workspace.

Comments: IERR = 1013 Element not included in the chosen database.
IERR = 1014 Element already rejected.

4.7.7 TQRESPH(PHNAM,IWSG)

Full name: Restore Phase.

Purpose: Restore a rejected system phase.

Arguments:	Name	Type	Value set on call or returned
	PHNAM	Character*24	Set to a phase name.
	IWSG	Integer array	Workspace.
Comments:	IERR = 1015	Phase not included in the chosen database.	
	IERR = 1016	Phase already restored.	

4.7.8 TQREJPH(PHNAM,IWSG)

Full name: Reject Phase.

Purpose: Reject a system phase.

Arguments:	Name	Type	Value set on call or returned
	PHNAM	Character*24	Set to a phase name.
	IWSG	Integer array	Workspace.
Comments:	IERR = 1017	Phase not included in the chosen database.	
	IERR = 1018	Phase already rejected.	

4.7.9 TQLISPH(PH_ARR,N,IWSG)

Full name: List System Phase.

Purpose: List all phases (both rejected and restored) available for the defined system.

Arguments:	Name	Type	Value set on call or returned
	PH_ARR	Character*24 array	Return phase names.
	N	Integer	Return the total number of phases
	IWSG	Integer array	Workspace.

Comments:

4.7.10 TQLISSF(PH_ARR,N,IWSG)

Full name: List Selected System Phase.

Purpose: List phases not rejected for the defined system.

Arguments:	Name	Type	Value set on call or returned
	PH_ARR	Character*24 array	Return phase names.
	N	Integer	Return the total number of phases
	IWSG	Integer array	Workspace.

Comments:

4.7.11 TQGDAT(IWSG)

Full name: Get Data.

Purpose: Get data for the defined system from the chosen database.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments:

4.7.12 TQREJS(IWSG)

Full name: Reject system.

Purpose: Reject the defined system and reinitiate the workspace in order to do a completely new calculation for a different system selected from the same or a different database.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.

Comments: In any application programs, TQINI (see 4.1.1) should be called only one time. If there is a need to do a completely new calculation on a totally different system without exiting the application program, one should call TQREJS instead before going to (open a new database and) define a new system, get data, and make calculations.

5. Examples

5.1 Example 1

```
C/*****
C/*
C/* This is a part of the source code samples demonstrating the use *
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc *
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational *
C/* Thermodynamics *
C/*
C/* This sample program shows how to retrieve data from a Thermo- *
C/* Calc data file, then define a set of conditions for a single *
C/* equilibrium calculation, get the equilibrium phases and their *
C/* amounts and compositions. The method of calculating the liquidus *
C/* and solidus temperature is also demonstrated. *
C/*
C/* Coded by: Qing Chen and Bo Sundman *
C/* Department of Materials Science and Engineering *
C/* Royal Institute of Technology *
C/* SE - 10044, Stockholm, Sweden *
C/*
C/*****

      program tqex01

      implicit double precision (a-h, o-z)

C...dimension of the workspace should be large enough
      parameter(nwg=80000)
C
      dimension iwsg(nwg)
      character*24 names(5),name
      character*8 stavar, sname
      logical pstat, tqgsp, sg2err

C...initiate the workspace
      call tqini(nwg,iwsg)

C...read the thermodynamic data file which was created by using
C...the GES module inside the Thermo-Calc software package
      call tqrfil('TQEX01',iwsg)

C...get component names in the system
      call tqgcom(icom,names,iwsg)
      print *, 'This system has the following components:'
      print *, (names(i),i=1,icom)
      print *
      print *

C...get number of phases in the system
      call tqgnp(iph, iwsg)
      print *, 'This system has', iph, ' phases:'

C...get names and status of the phases in the system
      do 10 i=1, iph
         call tqgpn(i, name, iwsg)
         pstat=tqgsp(i, sname, an, iwsg)
         print *, i, ' ', name, ' ', sname, ' ', an
10      continue

C...set the units of some properties.
      call tqssu('ENERGY','CAL',iwsg)
      call tqssu('T','C',iwsg)

C...set the condition for a single equilibrium calculation
      call tqsetc('T',-1,-1,500.0D0,icont,iwsg)
      call tqsetc('N',-1,-1,1.00D0,iconn,iwsg)
```

```

        call tqsetc('P',-1,-1,101325.0D0,iconp,iwsg)
        call tqsetc('W%',-1, 2, 4.0D0, iconw, iwsg)

c...calculate equilibrium
        call tqce(' ', 0, 0, 0.0D0, iwsg)
        if (sg2err(ierr)) goto 900

        call tqget1('GM',-1,-1,val,iwsg)
        print *, 'At T = 500.0 C, Wt% Mg = 4'
        print *, 'The Gibbs energy of the system is ', val, ' Cal/mol.'

c...find the equilibrium phase(s) and their compositions
        call writepx(iph,iwsg)

c...find the equilibrium liquidus and solidus temperature of this alloy
c...
c...change the status of liquid to FIXED with amount 1
c and then remove the temperature condition
        call tqgpi(iliq, 'liq', iwsg)
        call tqcsp(iliq, 'fixed', 1.0d0, iwsg)
        call tqremc(iconw,iwsg)
        call tqce(' ',0,0,0.0d0,iwsg)
        if (sg2err(ierr)) goto 900
        call tqget1('t',-1,-1,valt,iwsg)
        print *
        print *, 'Tliquidus = ', valt, ' C'
c...find the equilibrium phase(s) and composition(s)
        call writepx(iph,iwsg)

c...with fixed amount of liquid to 0
        call tqcsp(iliq, 'fixed', 0.0d0,iwsg)
        call tqce(' ',0,0,0.0d0,iwsg)
        if (sg2err(ierr)) goto 900
        call tqget1('t',-1,-1,valt,iwsg)
        print *
        print *, 'Tsolidus = ', valt, ' C'
c...find the equilibrium phase(s) and composition(s)
        call writepx(iph,iwsg)

c...find the temperature where the equilibrium solidification is half done
        call tqsetc(iliq, 'fixed', 0.5d0, iwsg)
        call tqce(' ',0,0,0.0d0,iwsg)
        if (sg2err(ierr)) goto 900
        call tqget1('t',-1,-1,valt,iwsg)
        print *
        print *, 'Tmid = ', valt, ' C'
c...find the equilibrium phase(s) and composition(s)
        call writepx(iph,iwsg)
        goto 1000

c... error handling
900 print *, 'Calculation failed!'

1000 continue
        end

        subroutine writepx(iph,iwsg)
        implicit double precision (a-h, o-z)
        dimension iwsg(*)
        character*24 name
        print *, 'phase name      composition, Wt%Mg      amount'
10  format(2x,A,2(10x,f8.3))
        do 100 i = 1, iph
            call tqget1('DG', i, -1, val, iwsg)
            if (val.eq.0.0d+0) then
                call tqgpn(i, name, iwsg)
                call tqget1('np', i, -1, valnp, iwsg)
                call tqget1('w%', i, 2, valw, iwsg)
                print 10, name(1:lens(name)), valw, valnp
            endif
        do 100 continue
        return
        end

```

5.2 Example 2

```

C/*****
C/*
C/* This is a part of the source code samples demonstrating the use *
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc *
C/* *
C/* Copyright (1996, 1999, 2000) Foundation for Computational *
C/* Thermodynamics *
C/* *
C/* This simple sample program calculate the To line for the fcc and *
C/* bcc phase in the Fe-C system. *
C/* *
C/* Coded by: Lars Hoglund and Malin Selleby *
C/* Department of Materials Science and Engineering *
C/* Royal Institute of Technology *
C/* SE - 10044, Stockholm, Sweden *
C/* *
C/*****

      program tqex02

C... a simple program to get the To line for the fcc and bcc phase
C... in the Fe-C system

      implicit double precision (a-h,o-z)
      double precision P,N
      parameter (nwsg=80000)
      dimension iwsg(nwsg)
      character*32 gfile,phas1,phas2

C
      gfile='TQEX02'

C
      phas1='FCC_A1'
      phas2='BCC_A2'

C
      Xmin=1.D-3
      Xmax=9.D-3
      dX=1.D-3
      Tmin0=1800.D0
      Tmax0=1670.D0

C
C.. initialization
      call tqini(nwsg,iwsg)
      call tqrfil(gfile,iwsg)

C
      call tqgpi(iph1,phas1,iwsg)
      call tqgpi(iph2,phas2,iwsg)

      call tqgsci(icmp,'C',iwsg)

      eps=1.D-6

      do 2000,XC=Xmin,Xmax,dX

          Tmax=Tmax0
          Tmin=Tmin0
          T=(Tmax0+Tmin0)*0.5D0
          P=101325.d0
          N=1.0D0
          its=0

          call tqcsp(iph1,'SUSPENDEED',0.D0,iwsg)
          call tqcsp(iph2,'SUSPENDEED',0.D0,iwsg)

          call tqsetc('P',-1,-1 ,P ,numcon,iwsg)
          call tqsetc('T',-1,-1 ,T ,numcon,iwsg)
          call tqsetc('N',-1,-1 ,N ,numcon,iwsg)
          call tqsetc('X',-1,icmp,XC,numcon,iwsg)

100      continue

          its=its+1

```

```

call tqsetc('T',-1,-1 ,T ,numcon,iwsg)

call tqcsp(iph1,'ENTERED',1.D0,iwsg)
call tqcsp(iph2,'SUSPENDEDED',0.D0,iwsg)
call tqce(' ',0,0,0.0D0,iwsg)
gm1 = tqggm(iph1,iwsg)
C
call tqcsp(iph1,'SUSPENDEDED',0.D0,iwsg)
call tqcsp(iph2,'ENTERED',1.D0,iwsg)
call tqce(' ',0,0,0.0D0,iwsg)
gm2 = tqggm(iph2,iwsg)
C
gmd=(gm1-gm2)/gm1
if (abs(gmd).le.eps) goto 200
if (gmd.lt.0.D0) then
  Tmin=T
  T=(T+Tmax)*0.5D0
endif
if (gmd.gt.0.D0) then
  Tmax=T
  T=(T+Tmin)*0.5D0
endif
goto 100
C
200 continue

write(*,*)' X(C),T,its ',XC,T,its

2000 continue
end

```

5.3 Example 3

```

C/*****
C/*
C/* This is a part of the source code samples demonstrating the use *
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc *
C/* *
C/* Copyright (1996, 1999, 2000) Foundation for Computational *
C/* Thermodynamics *
C/* *
C/* This sample program simulate the non-equilibrium solidification *
C/* under the Scheil-Guilliver condition. *
C/* *
C/* Coded by: Qing Chen and Bo Sundman *
C/* Department of Materials Science and Engineering *
C/* Royal Institute of Technology *
C/* SE - 10044, Stockholm, Sweden *
C/* *
C/*****

PROGRAM TQEX03
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NWG=80000,NC=20,ZERO=0.0D0)
DIMENSION IWSG(NWG)
DIMENSION TP(5),WP(NC)
CHARACTER FILE*60,NCOM(NC)*24,ASK*60
LOGICAL SG1ERR, SG2ERR

WRITE(*,1)
1 FORMAT(
&' '/
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&' '/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This sample program simulate the non-equilibrium solidification'/
&'under the Scheil-Guilliver condition.'/
&' '/
&'Qing Chen and Bo Sundman'/
&'Department of Materials Science and Engineering'/
&'Royal Institute of Technology, S-100 44, Stockholm, Sweden'/)

C...initializing
CALL TQINI(NWG,IWSG)

10 FORMAT(1X,A,$)
20 FORMAT(A)

C...read thermodynamic data from a GES file
WRITE(6,10)'Thermodynamic data file /TQEX03/: '
READ(5,20)FILE
IF(LEN_TRIM(FILE).LE.0) FILE='TQEX03.GES5'
CALL TQRFIL(FILE,IWSG)
IF(SG2ERR(IERR)) THEN
WRITE(6,*)'Failed to open the data file!'
GOTO 900
ENDIF

C...input of T and P
100 WRITE(6,10)'Temperature (C) /1800/: '
READ(5,*)TP(1)
WRITE(6,10)'Pressure (bar) /101325/: '
READ(5,*)TP(2)

C...input of composition of components, first the number of components
CALL TQGCNOM(NC,NCOM,IWSG)
IF(SG1ERR(IERR)) GOTO 900
SUM=100
DO 200 I=1,NC-1
ASK='Weight percent of '//NCOM(I)
150 WRITE(6,10)ASK(1:LEN_TRIM(ASK))//': '
READ(5,*)WP(I)
IF(WP(I).LE.ZERO .OR. WP(I).GE.SUM) THEN
WRITE(*,*)'Value out of limits'

```

```

                GOTO 150
            ENDIF
            SUM=SUM-WP(I)
200 CONTINUE
C...set equilibrium condition, P
    VAL=TP(2)
    CALL TQSETC('P',-1,-1,VAL,NCOND,IWSG)
C...set equilibrium condition, N
    CALL TQSETC('N',-1,-1,1.0D0,NCOND,IWSG)
C...set system composition, W
    DO 300 I=1,NE-1
        VAL=WP(I)
        CALL TQSETC('W%',-1,I,VAL,NCOND,IWSG)
300 CONTINUE
C...set unit of temperature to Celsius degree
    CALL TQSSU('TEMP','C',IWSG)
C...set equilibrium condition, T
305 VALT=TP(1)
    CALL TQSETC('T',-1,-1,VALT,NCONT,IWSG)
C...calculate equilibrium
    CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...calculate the liquidus temperature by
C...first removing the temperature condition
    CALL TQREMC(NCONT,IWSG)
C...then getting the liquid index and fixing the amount to 1
    CALL TQGPI(IDP,'LIQUID',IWSG)
    CALL TQCSP(IDP,'FIXED ', 1.0D0, IWSG)
C...do a equilibrium calculation and get the liquidus temperature
    CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...occasionally, it may be difficult get the calculation done. see
C...how we are going to handle this situation
    IF(SG2ERR(IERR)) THEN
        CALL RESERR
        CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
        IF(SG2ERR(IERR)) THEN
            WRITE(*,*)'Sorry, try a reasonable starting temperature'
            CALL RESERR
            GOTO 100
        ENDIF
    ENDIF
    CALL TQGET1('T ', 0, 0, VALT, IWSG)
    WRITE(*,*)'The liquidus temperature is calculated: ',VALT,' C'
C...restore the normal condition by setting liquid ENTERED and perform a
C...calculation
    CALL TQCSP(IDP, 'ENTERED', 1.0D, IWSG)
    CALL TQSETC('T',-1,-1,VALT,NCONT,IWSG)
    CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
    IF(SG2ERR(IERR)) THEN
        CALL RESERR
        CALL TQFASV(IWSG)
        CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
        IF(SG2ERR(IERR)) THEN
            WRITE(*,*)'Sorry, try again.'
            TP(1)=VALT+100.0D0
            CALL RESERR
            GOTO 305
        ENDIF
    ENDIF
    ENDIF
    CALL TQGET1('NP', IDP, 0, VALP, IWSG)
    DH=0.0D0
    TSTEP=1.0D0
    WRITE(*,399)
+ 'Temperature, C      Liquid fraction  Latent heat change, J/mol'
399 format(5x,A)
    WRITE(*,800)VALT,VALP,DH
C...now we are going to perform the scheil-gulliver simulation
C...displace composition of liquid phase
400 DO 500 I=1,NE-1
    CALL TQGET1('W',IDP,I,DVAL,IWSG)
    CALL TQSETC('W',-1,I,DVAL,NCON,IWSG)
500 CONTINUE
    CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...lower the temperature
    VALT=VALT-TSTEP
    CALL TQSETC('T',-1,-1,VALT,NCONT,IWSG)
C...get H of liquid phase of the overall composition at this temperature

```

```

C...before calculation
  CALL TQGET1('HM',IDP,-1,VALH1,IWSG)
  CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...get H of the system after calculation
  CALL TQGET1('HM',-1,-1,VALH2,IWSG)
C...calculate latent heat evolution
  DH=DH+VALP*(VALH2-VALH1)
  CALL TQGET1('NP',IDP,-1,VALLP,IWSG)
C...calculate remaining liquid fraction
  VALP=VALP*VALLP
  WRITE(*,800)VALT,VALP,DH
800  FORMAT(3(F17.4))
C...if remaining liquid fraction less or equal 0.01, end
  IF(VALP.GT.0.01) GOTO 400
900  CONTINUE
  END

```



```

CALL TQSSU('MASS ', 'gram', IWSG)

c Legierungszusammensetzung initialisieren 1=Al, 2=Mg, 3=Si
c0(3) = 0.06
c0(2) = 0.06
iKomp=1
c0(1) = 1. - c0(3) - c0(2)

c Zustandsgrößen festlegen, d.h. Konzentrationen, und
c Liquidustemperatur berechnen, Temperatur ist target-Variable
c Legierungskonzentration setzen
do i=2, nKomp
    call tqsetc( 'X ', 0, i, c0(i), numcon, IWSG)
enddo
call tqsetc( 'N ', 0, 0, 1.D0, numcon, IWSG)
call tqsetc( 'P ', 0, 0, 1.D5, numcon, IWSG)

c first a calculation at fixed T at low T with little or no liquid
call tqsetc( 'T ', 0, 0, 3000.D0, numt, IWSG)
call tqce( ' ', 0, 0, 0.D0, IWSG)
c calculate when amount liquid ist 1, Liquidus
call tqremc( numt, IWSG)
call tqcsp( liquid, 'fixed', 1.D0, IWSG)
call tqce( ' ', 0, 0, 0.D0, IWSG)
call tqget1( 'T ', 0, 0, Tliq, IWSG)
c calculate when amount liquid ist 0 but stable, solidus temp.
call tqsetc(liquid, 'fixed', 0.D0, IWSG)
call tqce( ' ', 0, 0, 0.D0, IWSG)
if(sg2err(ierr)) write(*,*)ierr
call tqget1( 'T ', 0, 0, Tsol, IWSG)
call tqcsp(liquid, 'entered', 1.D0, IWSG)
ittt=Tliq+0.5
Tstart=ittt
call tqsetc( 'T ', 0, 0, Tstart, numt, IWSG)
call tqce( ' ', 0, 0, 0.D0, IWSG)
if(sglerr(ierr)) write(*,*)ierr
call reserr
Write(*,*) 'Tliq = ', real( Tliq)
Write(*,*) 'Tsol = ', real( Tsol)
Write(20,*) '# Tliq = ', real( Tliq)
Write(20,*) '# Tsol = ', real( Tsol)

T = Tliq
fracsol = 0.0
dT = 1.0
fsneu = 0

c Ermitteln, welche Phase (bzw. Index) aus liquid ausgeschieden wird
c die Aktivitaeten aller Phasen werden in ac_phase geschrieben
c alle mit ac = 1 sind stabil
dgmax=-1.0D2
ii=0
if(sg2err(ierr)) then
    write(*,*)ierr
    call reserr
endif
66  ii=ii+1
    if(ii.ne.liquid) then
        jj=ii
        indexc=0
        call tqget1( 'dg ', jj, indexc, dgm, IWSG)
        if(sglerr(ierr)) call reserr
        if(dgm.gt. dgmax) then
            dgmax=dgm
            solid = ii
        endif
    endif
    if(ii.lt.nPhase) goto 66
77  continue
C
    if(sglerr(ierr)) call reserr
    call tqgpn( solid, namep, IWSG)
    write(*,*) namep, ' wird als erste feste Phase stabil'

c cliq wird auf Legierungskonz. gesetzt
do i=2, nKomp

```

```

        cliq(i) = c0(i)
    enddo
    cliqalt = cliq(iKomp)
c Enthalpie des Systems bei Liquidustemperatur
    if(sglerr(ierr)) call reserr
        call tqget1( 'HM ', 0, 0, Hliq, IWSG)

        write(*,*)'H(liq): ',Hliq
        WRITE(20,15) Real(T), REAL (fracsol), REAL (0.),
*                   REAL (cliqalt), REAL (cliqalt)
15  FORMAT( F8.2, F8.4, 3F8.4)

Ccccccccccccccccccccccccccccccccccccccc Schleife ueber Temperatur
c Aktivitaet von liquid = 1 -> Phase ist stabil
    fl=1.0D0
    DO 113 WHILE ( .TRUE. )
        Write(*,*) T,fl,c0(2),c0(3)
c Wenn eine zweite feste Phase stabil ist: Ende
C        do i=1, nPhase
            i=0
107        i=i+1
                jj=i
                if(jj.ne.liquid .and. jj.ne.solid) then
                    call tqget1( 'dg ', jj, 0, dgm, IWSG)
                    if(dgm .eq. 0.0D0) then
                        call tqgpn( i, namep, IWSG)
                        Write(*,*) 'Eine dritte Phase ist stabil,'
                        Write(*,*) namep
                        Write(*,*) 'keine 3-Phasengleichgewichte'
                        goto 150
                    endif
                endif
            endif
C        enddo
            if(i.lt.nphase) goto 107
            T = T - dT
c Gleichgewicht bei T berechnen mit cliq als neuer Legierungskonz.
C I removed the lever rule equilibrium calculation
            call tqsetc('T',0,0,T,numt,iwsg)
            call tqsetc('X',0,2,c0(2),numcon,iwsg)
            call tqsetc('X',0,3,c0(3),numcon,iwsg)
            call tqce(' ',0,0,0.0D0,iwsg)
C
            if(sglerr(ierr)) call reserr
            call tqget1('X',liquid,2,c0(2),iwsg)
            call tqget1('X',liquid,3,c0(3),iwsg)
            call tqget1('NP',liquid,0,dfl,iwsg)
            fl=fl*dfl
            fracsol=1.0-fl
c Ergebnisse ausgeben: fs als Fkt. von T
            WRITE(20,10) Real(T), REAL (fracsol), c0(2), c0(3)
10          FORMAT( F8.2, F8.4, 3F8.4)
113  ENDDO
150  write(*,*) 'Anteil Primary phase', fracsol
c Enthalpie des System bei Solidustemperatur
    call tqget1( 'HM ', 0, 0, Hsol, IWSG)
    write(*,*) 'delta Hmelt', Hliq - Hsol
    write(*,*) ' Note: this include the enthalpy due to Cp'
    close(20)
    CALL SYSCPU(1,JCPU)
    write(*,*)'Cpu time ',JCPU-ICPU
    END

```

5.5 Example 5

```
C/*****
C/*
C/* This is a part of the source code samples demonstrating the use *
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc *
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational *
C/* Thermodynamics *
C/*
C/* This example demonstrates how to use stream calculation to get *
C/* the enthalpy of a reaction, i.e., the enthalpy difference between *
C/* the reaction products at one temperature and the reactants at *
C/* another temperature. By setting the enthalpy of reaction to zero, *
C/* the adiabatic temperature can be easily calculated. *
C/*
C/* Coded by: Qing Chen and Bo Sundman *
C/* Department of Materials Science and Engineering *
C/* Royal Institute of Technology *
C/* SE - 10044, Stockholm, Sweden *
C/*
C/*****

PROGRAM TQEX05

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

PARAMETER (NWG=80000,NC=20,ZERO=0.0D0)
PARAMETER (NELSP=10)
PARAMETER (NWP1=500000,NWP2=40000)
COMMON/TQ1/IWSE(NWP1),IWSW(NWP2)
DIMENSION IWG(NWG)
CHARACTER FILE*60,SNAME*9,NAME*24
LOGICAL SG2ERR,SG1ERR,PSTAT,TQGSP

1 write(*,1)
format(
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This example demonstrates how to use stream calculation to get'/
&'the enthalpy of a reaction, i.e., the enthalpy difference'/
&'between the reaction products at one temperature and the '/
&'reactants at another temperature. By setting the enthalpy of '/
&'reaction to zero, the adiabatic temperature can be easily '/
&'calculated.'/
&' '/
&'In this example, C3H8 is burned in oxygen. The thermodynamic'/
&'data file used contains data retrieved from the SSUB database.'/
&'Inputting a negative amount of C3H8 will stop the runing of '/
&'the program.'/
&' '/
&'Qing Chen and Bo Sundman'/
&'Department of Materials Science and Engineering'/
&'Royal Institute of Technology, S-100 44, Stockholm, Sweden'/)

C...initializing
CALL TQINI(NWG,IWSG)

10 FORMAT(1X,A,$)
20 FORMAT(A)

C...read thermodynamic data from a GES file
WRITE(6,10)'Thermodynamic data file /TQEX05/: '
READ(5,20)FILE
IF(LEN_TRIM(FILE).LE.0) FILE='TQEX05.GES5'
CALL TQRFIL(FILE,IWSG)
IF(SG2ERR(IERR)) THEN
WRITE(6,*)'Failed to open the data file!'
GOTO 900
ENDIF
```

```

C...get the index of the gas phase.
  CALL TQGPPI(IP,'GAS',IWSG)
  IF(SG2ERR(IERR)) GOTO 920
C...create a stream named STREAM1 and input the gas phase at a
C...temperature and one atmosphere
C...ask for the temperature of the stream
15  CONTINUE
    WRITE(6,10)'Stream temperatur (K) /298.15/: '
    READ(5,*)TIN
    CALL TQCSTM('STREAM1',TIN,1.0D5,IWSG)
    IF(SG2ERR(IERR)) GOTO 930
C...get the index of the consitituent C3H8 in the gas phase
  CALL TQGPCI(IP,IC,'C3H8',IWSG)
  IF(SG2ERR(IERR)) GOTO 940
C...ask and set the amount of C3H8 in the stream
  WRITE(6,10)'Moles of C3H8 /1/: '
  READ(5,*)C3H8
  IF(C3H8.LT.0.0D0) STOP ' *** have a nice day ! *** '
  CALL TQSSC('STREAM1',IP,IC,C3H8,0,IWSG)
C...get the index of the constituent O2
  CALL TQGPCI(IP,IC,'O2',IWSG)
C...ask and set the amount of O2 in the stream
  WRITE(6,10)'Moles of O2 /5/: '
  READ(5,*)O2
  CALL TQSSC('STREAM1',IP,IC,O2,0,IWSG)
C...ask and set the reaction temperature
  WRITE(6,10)'Reaction temperature (K) /3000/: '
  READ(5,*)TREA
C...set the reaction temperature and pressure
  call tqsetc('t',-1,-1,TREA,inu,iwsg)
  call tqsetc('p',-1,-1,1.0D5,inu,iwsg)
  CALL TQCE(' ',0,0,0.0D0,IWSG)
  IF(SG2ERR(IERR)) GOTO 900
C...get the enthalpy of the reaction
  CALL TQGET1('HX',-1,-1,VAL,IWSG)
  WRITE(*,*)'Initial Enthalpy of reactants = ', VAL, ' J'
  CALL TQGET1('H',-1,-1,VAL,IWSG)
  WRITE(*,*)'Enthalpy of products at ',TREA,'K is ', VAL, ' J'
  CALL TQGET1('HD',-1,-1,VAL,IWSG)
  WRITE(*,*)'Enthalpy of reaction at ',TREA,'K is ', VAL,' J'
C...specify the invariant state variable for the reaction calculation
  CALL TQSSIC('HD',0.0D0,IWSG)
C...make an estimation of the result temperature, if you can
  CALL TQCE('T',0,0,3000.0D0,IWSG)
C...get the temperature
  CALL TQGET1('T',-1,-1,VAL,IWSG)
  WRITE(*,*)'The adiabatic reaction temperature is ',VAL,' K'
C...delete stream
  CALL TQDSTM(' ',IWSG)
  GOTO 15
900  GOTO 990
910  GOTO 990
920  GOTO 990
930  GOTO 990
940  GOTO 990
950  GOTO 990
990  CONTINUE
      END

```

5.6 Example 6

```
C/*****
C/*
C/* This is a part of the source code samples demonstrating the use
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational
C/* Thermodynamics
C/*
C/* This example demonstrates how to use stream calculation to obtain
C/* the chill factors in the steelmaking industry.[1]
C/*
C/*
C/* Coded by: Qing Chen and Bo Sundman
C/* Department of Materials Science and Engineering
C/* Royal Institute of Technology
C/* SE - 10044, Stockholm, Sweden
C/*
C/* Reference
C/* [1] O.Kubaschewski and C.B. Alock, Metallurgical Thermochemistry,
C/* 1979, Page 211.
C/*****
```

```
PROGRAM TQEX06
```

```
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NWG=80000,NC=20,ZERO=0.0D0)
PARAMETER (NELSP=10)
PARAMETER (NWP1=500000,NWP2=40000)
COMMON/TQ1/IWSE(NWP1),IWSW(NWP2)
DIMENSION IWSG(NWG)
CHARACTER FILE*60
LOGICAL SG2ERR
```

```
WRITE(*,1)
1  FORMAT(
&' '/
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&' '/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This example demonstrates how to use stream calculation to '/
&'obtain the chill factors in the steelmaking industry[1].'/
&' '/
&' [1] O.Kubaschewski and C.B. Alock, Metallurgical Thermochemistry,
&' /
&' 1979, Page 211.'/
&' '/
&'Qing Chen and Bo Sundman'/
&'Department of Materials Science and Engineering'/
&'Royal Institute of Technology, S-100 44, Stockholm, Sweden'/)
```

```
C...initializing
CALL TQINI(NWG,IWSG)
```

```
10  FORMAT(1X,A,§)
20  FORMAT(A)
```

```
C...read thermodynamic data from a GES file
WRITE(6,10)'Thermodynamic data file /TQEX06/: '
READ(5,20)FILE
IF(LEN_TRIM(FILE).LE.0) FILE='TQEX06.GES5'
CALL TQRFIL(FILE,IWSG)
IF(SG2ERR(IERR)) THEN
WRITE(6,*)'Failed to open the data file!'
GOTO 900
ENDIF
```

```
C...change temperature unit to C
CALL tqssu('T','C',iwsg)
```

```

C...create a stream named LIQUID_FE and input Fe liquid at the 1600 C
C...and one atmosphere
    Tliq_fe = 1600.0D0
    CALL TQCSTM('LIQUID_FE',Tliq_fe,1.0D5,IWSG)
    IF(SG2ERR(IERR)) GOTO 930

C...get the index of the liquid phase.
    CALL TQGPI(IPL,'LIQUID',IWSG)
    IF(SG2ERR(IERR)) GOTO 920

C...get the index of the consitituent Fe in the liquid phase
    CALL TQGPCI(IPL,IC,'FE',IWSG)
    IF(SG2ERR(IERR)) GOTO 940

C...set the amount of Fe in the stream to 49 mole
    CALL TQSSC('LIQUID_FE',IPL,IC,49.0D0,0,IWSG)

C...create another stream named MASTER_FESI and input Fe1Si1 at the room
C...temperature and one atmosphere

    CALL TQCSTM('MASTER_FESI',25.0D0,1.0D5,IWSG)
    IF(SG2ERR(IERR)) GOTO 930

C...get the index of the master alloy FE1SI1_S phase.
    CALL TQGPI(IPS,'FE1SI1_S',IWSG)
    IF(SG2ERR(IERR)) GOTO 920

C...set the amount of FE1SI1 in the stream to 1 mole
    CALL TQSSC('MASTER_FESI',IPS,1,1.0D0,0,IWSG)

C...set the reaction pressure
    call tqsetc('p',-1,-1,1.0D5,inu,iwsg)

C...specify the invariant state variable for the reaction calculation
    CALL TQSSIC('DH',0.0D0,IWSG)

C...make a estimation of the result temperature, if you can
    CALL TQCE('T',0,0,1500.0D0,IWSG)

C...get the temperature
    CALL TQGET1('T',-1,-1,VAL,IWSG)
    DT = VAL - Tliq_fe

    WRITE(*,*)'DT = ', DT

    CALL TQGET1('X',-1,2,VAL,IWSG)
    print *, 'The comp. of the final solution is ',VAL,' at% Si'

    CALL TQGET1('DG',IPS,-1,VAL,IWSG)
    if(val.lt.0.0D0) then
        WRITE(*,*)'The FE1SI1_S phase should not be stable !'
    else
        Write(*,*)'Strange, FE1SI1_S phase is still stable !'
    endif

C...set the amount of Fe in the stream to 98 mole
    CALL TQSSC('LIQUID_FE',IPL,IC,98.0D0,0,IWSG)
C...make a estimation of the result temperature, if you can
    CALL TQCE('T',0,0,1500.0D0,IWSG)
C...get the temperature
    CALL TQGET1('T',-1,-1,VAL,IWSG)
    DT = VAL - Tliq_fe

    WRITE(*,*)'DT = ', DT

    CALL TQGET1('X',-1,2,VAL,IWSG)
    print *, 'The comp. of the final solution is ',VAL,' at% Si'

    CALL TQGET1('DG',IPS,-1,VAL,IWSG)
    if(val.lt.0.0D0) then
        WRITE(*,*)'The FE1SI1_S phase should not be stable !'
    else
        Write(*,*)'Strange, FE1SI1_S phase is still stable !'
    endif

```

```
900 GOTO 990
910 GOTO 990
920 GOTO 990
930 GOTO 990
940 GOTO 990
990 CONTINUE
END
```

5.7 Example 7

```

C/*****
C/*
C/* This is a part of the source code samples demonstrating the use
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational
C/* Thermodynamics
C/*
C/* This sample program calculate the A3 temperature of a steel
C/* and determine the influence of each alloying element on this
C/* temperature. It demonstrates that some very special quantities,
C/* such as the composition derivative of temperature, can be
C/* obtained easily via the TQ interface.
C/*
C/* Coded by:
C/*
C/* Qing Chen and Bo Sundman
C/* Department of Materials Science and Engineering
C/* Royal Institute of Technology
C/* S-100 44, Stockholm, Sweden
C/*
C/*****

PROGRAM TQEX07

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NWG=80000,NC=6,ZERO=0.0D0)
DIMENSION IWSG(NWG),NCW(NC)
DIMENSION TP(5),WP(NC),WPD(NC)
CHARACTER FILE*60,NCOM(NC)*2,ASK*60,DTDW*10
LOGICAL SG1ERR, SG2ERR
DATA WPD/0.3D0, 1.5D0, 97.3D0, 0.5D0, 0.1D0, 0.3D0/

WRITE(*,1)
1 FORMAT(
&' '/
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&' '/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This example calculate the A3 temperature of a steel '/
&'and determine the influence of each alloying element on this'/
&'temperature. It demonstrates that some very special quantities,'/
&'such as the composition derivative of temperature, can be '/
&'obtained easily via the TQ interface.'/
&' '/
&'Qing Chen and Bo Sundman'/
&'Department of Materials Science and Engineering'/
&'Royal Institute of Technology, S-100 44, Stockholm, Sweden'/)

C...initializing
CALL TQINI(NWG,IWSG)

10 FORMAT(1X,A,$)
20 FORMAT(A)

C...read thermodynamic data from a GES file
WRITE(6,10)'Thermodynamic data file /TQEX07/: '
READ(5,20)FILE
IF(LEN_TRIM(FILE).LE.0) FILE='TQEX07.GES5'
CALL TQRFIL(FILE,IWSG)
IF(SG2ERR(IERR)) THEN
WRITE(6,*)'Failed to open the data file!'
GOTO 900
ENDIF

C...input of T and P
100 WRITE(6,10)'Temperature (C) /1000/: '
READ(5,*)TP(1)

```

```

WRITE(6,10)'Pressure (bar) /100000/: '
READ(5,*)TP(2)
C...input of composition of components, first the number of components
CALL TQGC(NE,NCOM,IWSG)
IF(SG1ERR(IERR)) GOTO 900
SUM=100
DO 200 I=1,NE
  IF(NCOM(I).EQ.'FE') GOTO 200
  ASK='Weight percent of '//NCOM(I)
150  WRITE(6,10)ASK(1:LEN_TRIM(ASK))//': '
  READ(5,*)WP(I)
  IF(WP(I).GE.SUM) THEN
    WRITE(*,*)'Value out of limits'
    GOTO 150
  ELSEIF( WP(I).LE.0.0) THEN
    STOP '*** Having a nice day! ***'
  ENDIF
  SUM=SUM-WP(I)
200  CONTINUE

C...set equilibrium condition, P
VAL=TP(2)
CALL TQSETC('P',-1,-1,VAL,NCOM,IWSG)
C...set equilibrium condition, N
CALL TQSETC('N',-1,-1,1.0D00,NCOM,IWSG)
C...set system composition, W
DO 300 I=1,NE
  IF(NCOM(I).EQ.'FE') GOTO 300
  VAL=WP(I)
  CALL TQSETC('W%',-1,I,VAL,NCW(I),IWSG)
300  CONTINUE
C...set unit of temperature to Celsius degree
CALL TQSSU('TEMP','C',IWSG)
C...set equilibrium condition, T
305  VALT=TP(1)
  CALL TQSETC('T',-1,-1,VALT,NCOM,IWSG)
C...calculate equilibrium
CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...calculate the A3 temperature by
C...first removing the temperature condition
CALL TQREMC(NCOM,IWSG)
C...then getting the bcc index and fixing the amount to 0
CALL TQGPI(IDP,'BCC',IWSG)
CALL TQCS(IDP,'FIXED', 0.0D0, IWSG)
C...do a equilibrium calculation and get the A3 temperature
CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
C...occasionally, it may be difficult get the calculation done. see
C...how we are going to handle this situation
IF(SG2ERR(IERR)) THEN
  CALL RESERR
  CALL TQCE(' ', 0, 0, 0.0D0, IWSG)
  IF(SG2ERR(IERR)) THEN
    WRITE(*,*)'Sorry, try a reasonable starting temperature or
&composition.'
    CALL RESERR
    CALL TQPINI(IWSG)
    GOTO 100
  ENDIF
ENDIF
CALL TQGET1('T ', 0, 0, VALT, IWSG)
WRITE(*,*)'The A3 temperature is calculated as ',VALT,' C'
C...now we shall see how a small change of the alloy composition
C...can alter this temperature. This is obtained by getting a very
C...special quantity --- the derivative of this temperature with respect
C...to the content of an alloying element, i.e. T.W(A).
DO 400 I=1,NE
  IF(NCOM(I).EQ.'FE') GOTO 400
  DTDW='T.W('//NCOM(I)
  DTDW(LEN(DTDW)+1:)=') '
  CALL TQGET1(DTDW,-1,-1,VALD,IWSG)
C...remember what we get is a value with respect to the mass fraction
C...in order to have the value referred to the mass percent, we should
C...divide the value by 100.
  VALD=VALD/100
  IF(VALD.GT.0) THEN
    WRITE(*,*)'The A3 temperature increases ',ABS(VALD),' C with

```

```

&1% increase of ',NCOM(I)
ELSE
WRITE(*,*)'The A3 temperature decreases ',ABS(VALD),' C with
&1% increase of ',NCOM(I)
ENDIF
400 CONTINUE
C...if we want to decrease A3 temperature by 20 C, calculate how much
C...Mn we should add to the alloy
VALT=VALT-20
CALL TQSETC('T',-1,-1,VALT,NCONT,IWSG)
CALL TQGSCI(IMN,'MN',IWSG)
CALL TQREMC(NCW(IMN),IWSG)
CALL TQCE(' ',0,0,0.0D0,IWSG)
CALL TQGET1('W% ',0,IMN,VALT,IWSG)
WRITE(*,*)'A decrease of the A3 temperature by 20 C needs',
&VALT-WP(IMN),'% more of Mn. '
WRITE(*,*)'Let''s try another composition. A negative
& input will terminate the program.'
CALL TQPINI(IWSG)
GOTO 100
900 CONTINUE
END

```

5.8 Example 8

```

C/*****
C/*   This is a part of the source code samples demonstrating the use   *
C/*   of the Thermodynamic Calculation Interface(TQ) for Thermo-Calc   *
C/*   Copyright (1995) Foundation for Computational Thermodynamics.   *
C/*   *
C/*   This simple sample program display the diffusion matrix in a   *
C/*   multicomponent system.   *
C/*   *
C/*   Lars Hoglund   *
C/*   Department of Materials Science and Engineering   *
C/*   Royal Institute of Technology   *
C/*   S-100 44, Stockholm, Sweden   *
C/*****

      program tqex08

      implicit double precision (a-h,o-z)
      double precision P,N
      parameter (nwsg=80000)
      dimension iwsg(nwsg)
      character*32 gfile,phas1,phas2

      gfile='TQEX08.GES5'

      phas1='FCC_A1'

C...  init
      call tqini(nwsg,iwsg)
      call tqrfil(gfile,iwsg)

      call tqgpi(iph1,phas1,iwsg)

      call tqgsci(icmp1,'C',iwsg)
      call tqgsci(icmp2,'CR',iwsg)

      P=101325.d0
      N=1.0D0
      T=1000.D0

      xFC=0.01
      xFCR=0.1

      call tqcsp(iph1,'ENTERED',1.D0,iwsg)

      call tqsetc('P',-1,-1 ,P ,numcon,iwsg)
      call tqsetc('T',-1,-1 ,T ,numcon,iwsg)
      call tqsetc('N',0,0 ,N ,numcon,iwsg)
      call tqsetc('X',-1,icmp1,XFC,numcon,iwsg)
      call tqsetc('X',-1,icmp2,XFCR,numcon,iwsg)

      call tqce(' ',0,0,0.0D+0,iwsg)
      call tqget1('DC(FCC,C,C,FE) ',-1,-1,dc1,iwsg)
      call tqget1('DC(FCC,C,CR,FE) ',-1,-1,dc2,iwsg)
      call tqget1('DC(FCC,CR,C,FE) ',-1,-1,dc3,iwsg)
      call tqget1('DC(FCC,CR,CR,FE) ',-1,-1,dc4,iwsg)

      123  format('Diffusion matrix in FCC Fe-Cr-C ',/,
&         1x,2(12X,A2),/,
&         2(1X,A3,2(2X,G12.4),/))
      write(*,123)'C','CR','C ',dc1,dc2,'CR ',dc3,dc4

      end

```

5.9 Example 9

```

C/*****
C/*      This is a part of the source code samples demonstrating the use      *
C/*      of the Thermodynamic Calculation Interface(TQ) for Thermo-Calc      *
C/*      Copyright(1995-2004) Foundation for Computational Thermodynamics.    *
C/*      *
C/*      This simple sample program demonstrate the use of extra speedy      *
C/*      subroutines for retrieving energy and mobility data.                  *
C/*      *
C/*      Qing Chen and Bo Sundman                                             *
C/*      Department of Materials Science and Engineering                       *
C/*      Royal Institute of Technology                                         *
C/*      S-100 44, Stockholm, Sweden                                          *
C/*****

      program tqex09

      implicit double precision (a-h,o-z)
      parameter (maxc=100,maxc1=maxc+1,maxs=10)
      parameter (one=1.d0, zero=0.d0)
      parameter (nwsg=80000)
      dimension iwsg(nwsg)
      character*24 gfile,phas,spname(3)
      dimension tp(5),nkl(maxs),knr(maxc),yf(maxc),as(maxs),extra(5)
      dimension x(maxc1)
      integer isp(3)
      logical sglerr,sg2err,tqcmobb

C      gfile='TQEX08'
      phas='FCC_A1'

      spname(1)='C'
      spname(2)='CR'
      spname(3)='FE'

C..  initialize
      call tqini(nwsg,iwsg)
C..  read datafile
      call tqrfil(gfile,iwsg)
C..  get phase index
      call tqgpi(iph,phas,iwsg)
      if(sglerr(ierr))goto 900
C..  check if mobility data available
      if(.not.tqcmobb(iph,iwsg)) goto 900
C..  get phase constitution
      call tqgpd(iph,nsg,nkl,as,yf,extra,iwsg)
C      nsg = number of sublattices
C      nkl(nsg) = number of constituents at each sublattice
C      yf(sigma(nkl(i))) = site fraction in the same order
C      as(nsg) = number of sites in each sublattice.
C      extra(5) = others
C..  input conditions
      tp(1)=1000.d0
      tp(2)=1.d5
      xc=1.d-2
      xcr=5.d-2
      xfe=1.d0-xc-xcr
      yf(1)=xcr/(1.d0-xc)
      yf(2)=xfe/(1.d0-xc)
      yf(3)=as(1)/as(2)*xc/(1.d0-xc)
      yf(4)=1.d0-yf(3)
C..  input temperature and pressure
      call tqstp(tp,iwsg)
C..  input site fractions
      call tqsyf(iph,yf,iwsg)
C..  get gibbs energy and its first derivatives
      call tqgmdy(iph,x,iwsg)

      write(6,*)'System: C-CR-FE'
      write(6,*)'Phase: ',phas
      write(6,*)'Constitution: (CR,FE)1(C,VA)1'

```

```

write(6,*)'Temperature:',tp(1)
write(6,*)'Pressure: ',tp(2)
write(6,20)xc,xcr
20  format(' X(C)=' ,1P,E12.4, ' X(CR)=' ,E12.4)
write(6,30)(x(i)/(as(1)+as(2)*yf(3)),i=1,5)
30  format(' Gm=' ,1P,E14.7,
&      /' GmY(CR#1)=' ,E14.7, ' GmY(FE#1)=' ,E14.7,
&      /' GmY(C#2)=' ,E14.7, ' GmY(VA#2)=' ,E14.7)
C..  calculate chemical potentials from partial derivatives
amuc=(x(4)-x(5))/as(2)
amucr=(x(1)+(1-yf(4))*(x(5)-x(4))+(1-yf(1))*(x(2)-x(3)))/as(1)
amufe=(x(1)+(1-yf(4))*(x(5)-x(4))+(1-yf(2))*(x(3)-x(2)))/as(1)
write(6,40)amuc,amucr,amufe
40  format(' MU(C)=' ,1P,E14.7, ' MU(CR)=' ,E14.7, ' MU(FE)=' ,E14.7)
C..  get system species index
do i=1,3
    call tqgsspi(spname(i),isp(i),iwsg)
    if(sglerr(ierr)) goto 900
enddo
C..  get mobility data
do k=1,3
    call tqgmob(iph,isp(k),x(k),iwsg)
    if(sglerr(ierr)) goto 900
enddo
write(6,50)(x(i),i=1,3)
50  format(' M(C)=' ,1P,E14.7, ' M(CR)=' ,E14.7, ' M(FE)=' ,E14.7)
900  continue
end

```

5.10 Example 10

```
C/*****
C/* This is a part of the source code samples demonstrating the use *
C/* of the Thermodynamic calcULATION Interface (TQ) for Thermo-Calc *
C/* *
C/* Copyright (1996,1999,2004) Foundation for Computational *
C/* Thermodynamics *
C/* *
C/* This sample program does same thing as Example 9 except that it *
C/* demonstrates how to convert mole fractions to *
C/* site fractions and first derivatives of Gm w.r.t. site fractions *
C/* to that w.r.t. mole fractions. People feel comfortable *
C/* with site fractions and first derivatives w.r.t. them may skip *
C/* this example. *
C/* *
C/* Coded by: *
C/* *
C/* Qing Chen *
C/* Thermo-Calc Software AB *
C/* Stockholm Technology Park *
C/* S-113 47, Stockholm, Sweden *
C/*****
```

```
program tqex10

implicit double precision (a-h,o-z)
parameter (maxc=100,maxc1=maxc+1,maxs=10)
parameter (one=1.d0, zero=0.d0)
parameter (nwsg=80000)
dimension iwsg(nwsg),iwork(4*maxc)
character*24 gfile,phas,spname(3)
dimension tp(5),yf(maxc)
dimension dgdxc(maxc),x(maxc),amu(maxc),dgdy(maxc1)
dimension work(2000)
integer isp(3)
logical tqg2err,tqcmobb

WRITE(*,1)
1 FORMAT(
&' '/
&' This is a part of the source code samples demonstrating '/
&' the use of the Thermodynamic calcULATION interface (TQ) '/
&' for Thermo-Calc'/'
&' '/
&' Copyright (1996,1999,2004) Foundation for Computational '/
&' Thermodynamics'/'
&' '/
&' This sample program does same thing as Example 9 except '/
&' that it shows how to convert mole fractions to '/
&' site fractions and first derivatives of Gm w.r.t. site '/
&' fractions to that w.r.t. mole fractions. '/
&' '/
&' Qing Chen'/'
&' Thermo-Calc Software AB'/'
&' Stockholm Technology Park, S-113 47, Stockholm, Sweden'/'

gfile='TQEX08'
phas='FCC_A1'

spname(1)='C'
spname(2)='CR'
spname(3)='FE'

C.. initialize
call tqini(nwsg,iwsg)

C.. read datafile
call tqrfil(gfile,iwsg)

C.. get phase index
call tqgpi(iph,phas,iwsg)
```

```

        if(tqg2err(ierr)) goto 900
C..  check if mobility data available
    if(.not.tqcmobb(iph,iwsg)) goto 900

C..  input conditions
    tp(1)=1000.d0
    tp(2)=1.d5
    x(1)=1.d-2
    x(2)=5.d-2
    x(3)=1.d0-x(1)-x(2)

C...  convert mole fractions to y-fractions
C...  note: only possible if no internal degree of freedom in the phase

C...  get phase property
    call tqgphp(iph,ne,ncnv,nc,iwork,work,iwsg)
C...  converting
    call tqx2y(iph,ne,ncnv,nc,iwork,work,x,yf,iwsg)

C..  input temperature and pressure
    call tqstp(tp,iwsg)
C..  input site fractions
    call tqsyf(iph,yf,iwsg)
C..  get gibbs energy and its first derivatives wrt site fractions
    call tqgmdy(iph,dgdy,iwsg)
C...  convert to first derivatives wrt mole fractions
    call tqgmdx(iph,ne,ncnv,nc,iwork,work,yf,dgdy,
&              gm,dgdx,x,iwsg)

    write(6,*) 'System: C-CR-FE'
    write(6,*) 'Phase: ',phas
    write(6,*) 'Constitution: (CR,FE)1(C,VA)1'
    write(6,*) 'Temperature:',tp(1)
    write(6,*) 'Pressure: ',tp(2)
    write(6,20)(x(i),i=1,3)
20  format(' X(C)=' ,1P,E12.4,' X(CR)=' ,E12.4,' X(FE)=' ,E12.4)
    write(6,30)gm,(dgdx(i),i=1,3)
30  format(' Gm=' ,1P,E14.7,
&        /' GmX(C)=' ,E14.7,' GmX(CR)=' ,E14.7,
&        /' GmX(FE)=' ,E14.7)

C...  calculate chemical potentials using a standard formula
    do i=1,3
        amu(i)=gm
        do j=1,3
            amu(i)=amu(i)+(delta(i,j)-x(j))*dgdx(j)
        enddo
    enddo
    write(6,40)(amu(i),i=1,3)
40  format(' MU(C)=' ,1P,E14.7,' MU(CR)=' ,E14.7,' MU(FE)=' ,E14.7)
C..  get system species index
    do i=1,3
        call tqgsspi(spname(i),isp(i),iwsg)
        if(tqg2err(ierr)) goto 900
    enddo
C..  get mobility data
    do k=1,3
        call tqgmob(iph,isp(k),x(k),iwsg)
        if(tqg2err(ierr)) goto 900
    enddo
    write(6,50)(x(i),i=1,3)
50  format(' M(C)=' ,1P,E14.7,' M(CR)=' ,E14.7,' M(FE)=' ,E14.7)
900  continue
    end

    double precision function delta(i,j)
    delta=0.d0
    if(i.eq.j) delta=1.d0
    return
    end

```



```

write(10,*) ' '
write(10,*) ' '

C...get the interstitial C and/or N's index
ni = 0
do i=1,icom
  if(names(i)(1:lens(names(i))).eq.'C'.or.
+   names(i)(1:lens(names(i))).eq.'N') then
    ni = ni + 1
    ii(ni)=i
  endif
  if(names(i)(1:lens(names(i))).eq.'FE') imajor=i
end do

C...get number of phases in the system

call tqgpn(iph, iwsg)
write(6,*) 'This system has', iph, ' phases:'
write(10,*) 'This system has', iph, ' phases:'

C...get names and status of the phases in the system
do 10 i=1, iph
  call tqgpn(i, name, iwsg)
  pstat=tqgsp(i, sname, an, iwsg)
  write(6,*) i, ' ', name, ' ', sname, ' ', an
  write(10,*) i, ' ', name, ' ', sname, ' ', an
  if(index(name,'FCC').gt.0.and.
+   (index(name,'#').le.0.or.index(name,'#1').gt.0)) then
    im=i
  elseif(index(name,'BCC').gt.0.and.
+   (index(name,'#').le.0.or.index(name,'#1').gt.0)) then
    ip=i
  endif
10 continue

C.. set mode of calculation
C...mode = 1 or -1 means mole-fraction
C...mode = 2 or -2 means mass-fraction
C...mode = 3 or -3 means u-fraction
C...if mode is negative, local equilibrium is not calculated.

mode=3

C... set matrix composition

xm(imajor)=1.0d0
do i=1,icom
  if(i.ne.imajor) then
    xm(i)=1.d-2
    isint=0
    if(abs(mode).eq.3) then
      do j=1,ni
        if(i.eq.ii(j)) isint=1
      end do
    endif
    if(isint.eq.0) xm(imajor)=xm(imajor)-xm(i)
  endif
enddo

write(6,*) ' '
write(10,*) ' '
write(6,*) 'Matrix composition: '
write(10,*) 'Matrix composition: '
write(6,*) ' '
write(10,*) ' '

do i=1,icom
  if(i.ne.imajor) then
    if(abs(mode).eq.1) then
      write(6,40) names(i), xm(i)
      write(10,40) names(i), xm(i)
    elseif(abs(mode).eq.2) then
      write(6,41) names(i), xm(i)
      write(10,41) names(i), xm(i)
    elseif(abs(mode).eq.3) then

```

```

        write(6,42) names(i), xm(i)
        write(10,42) names(i), xm(i)
    endif
end do

40  format(' X(',A2,')=',1PE12.5)
41  format(' W(',A2,')=',1PE12.5)
42  format(' U(',A2,')=',1PE12.5)

write(6,45) 'T','DF','XP','XIM','XIP','MU(C)'
write(10,45) 'T','DF','XP','XIM','XIP','MU(C)'
45  format(' '/6(1x,A12)/' ')
50  format(6(1x,1PE12.5))

do i=0,300,10
    temp=1173.d0-i
    call tqgdf2(mode,im,ip,ni,ii,xm,temp,df,xp,xim,xip,amui,iwsg)
    if(.not.tqglerr(ierr)) then
        write(6,50) temp, df, (xp(ii(j)),j=1,ni), (xim(ii(j)),j=1,ni),
+           (xip(ii(j)),j=1,ni), (amui(ii(j)),j=1,ni)
        write(10,50) temp, df, (xp(ii(j)),j=1,ni), (xim(ii(j)),j=1,ni),
+           (xip(ii(j)),j=1,ni), (amui(ii(j)),j=1,ni)
    else
        write(6,*) temp, 'calculation failed'
    endif
end do
close(10)
end

```

5.12 Example 12

```

C/*****
C/*
C/* This is a part of the source code samples demonstrating the use
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational
C/* Thermodynamics
C/*
C/* This sample program demonstrates how to use subroutines getting
C/* system data from a database and how to restart new calculation
C/* on a different system in the same application program
C/*
C/* Coded by:
C/*
C/* Qing Chen
C/* Thermo-Calc Software AB
C/* Bjornnasvagen 21
C/* S-113 47, Stockholm, Sweden
C/*
C/*****

PROGRAM TQEX12

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NWG=80000,NC=20,ZERO=0.0D0)
DIMENSION IWSG(NWG)
CHARACTER*24 TDB(100),ENAME(100),DBNAME
CHARACTER*24 PNAME(100),NAME
CHARACTER*2 EL1,EL2
CHARACTER*8 SNAME
CHARACTER*1 YES
DOUBLE PRECISION N,P,T,X2

LOGICAL TQG2ERR

WRITE(*,1)
1 FORMAT(
&' '/
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&' '/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This sample program demonstrates how to use subroutines getting'/
&'system data from a database and how to restart new calculation'/
&'on a different system in the same application program.'/
&' '/
&'Qing Chen'/
&'Thermo-Calc Software AB'/
&'Bjornnasvagen 21, S-113 47, Stockholm, Sweden'/)

C...initializing
CALL TQINI(NWG,IWSG)
10 FORMAT(1X,A,$)
20 FORMAT(A)
60 CALL TQREJSY(IWSG)
IF(TQG2ERR(IERR)) THEN
WRITE(6,*) 'ERROR IN REJECTING DEFINED SYSTEM!'
STOP
ENDIF
CALL TQGDBN(TDB,ND,IWSG)
IF(TQG2ERR(IERR)) THEN
WRITE(6,*) 'ERROR IN GETTING DATABASE LIST!'
STOP
ELSE
WRITE(6,*) 'AVAILABLE DATABASE: '
WRITE(6,*) (TDB(I),I=1,ND)
ENDIF
100 WRITE(6,10) 'Select a database from above list: '

```

```

READ(5,20) DBNAME
CALL TQOPDB(DBNAME,IWSG)
IF(TQG2ERR(IERR)) THEN
    CALL TQRSERR
    WRITE(6,*) 'THERE IS NO SUCH DATABASE! TRY AGAIN.'
    GOTO 100
ENDIF
CALL TQLIDE(ENAME,NE,IWSG)
IF(TQG2ERR(IERR)) THEN
    WRITE(6,*) 'ERROR IN LISTING DATABASE ELEMENTS!'
    STOP
ELSE
    WRITE(6,*) 'AVAILABLE DATABASE ELEMENTS: '
    WRITE(6,*) (ENAME(I),I=1,NE)
ENDIF
110 WRITE(6,10) 'Select first element from above list: '
    READ(5,20) EL1
    CALL TQDEFEL(EL1,IWSG)
    IF(TQG2ERR(IERR)) THEN
        CALL TQRSERR
        WRITE(6,*) 'THERE IS NO SUCH ELEMENT! TRY AGAIN.'
        GOTO 100
    ENDIF
120 WRITE(6,10) 'Select second element from above list: '
    READ(5,20) EL2
    CALL TQDEFEL(EL2,IWSG)
    IF(TQG2ERR(IERR)) THEN
        CALL TQRSERR
        WRITE(6,*) 'THERE IS NO SUCH ELEMENT! TRY AGAIN.'
        GOTO 120
    ENDIF
    CALL TQLISPH(PNAME,NP,IWSG)
    IF(TQG2ERR(IERR)) THEN
        WRITE(6,*) 'CANNOT LIST ALL SYSTEM PHASES IN THE DATABASE!'
        STOP
    ELSE
        WRITE(6,*) 'All phases for the system in the database: '
        WRITE(6,*) (PNAME(I),I=1,NP)
    ENDIF
    CALL TQLISSF(PNAME,NP,IWSG)
    IF(TQG2ERR(IERR)) THEN
        WRITE(6,*) 'CANNOT LIST SELECTED PHASES IN THE DATABASE!'
        STOP
    ELSE
        WRITE(6,*) 'Selected system phases: '
        WRITE(6,*) (PNAME(I),I=1,NP)
    ENDIF
130 WRITE(6,*) ''
    WRITE(6,10) 'Reject any phase(s)? /NONE/: '
    READ(5,20) NAME
    IF (LEN_TRIM(NAME) .GT. 0 .AND. NAME(1:4) .NE. 'NONE' .AND.
    & NAME(1:4) .NE. 'none') THEN
        CALL TQREJPH(NAME,IWSG)
        IF(TQG2ERR(IERR)) THEN
            CALL TQRSERR
            WRITE(6,*) ' CANNOT REJECT PHASE ',NAME
        ENDIF
        CALL TQLISSF(PNAME,NP,IWSG)
        IF(TQG2ERR(IERR)) THEN
            WRITE(6,*) 'CANNOT LIST PHASES IN THE DATABASE!'
            STOP
        ELSE
            WRITE(6,*) 'Selected system phases: '
            WRITE(6,*) (PNAME(I),I=1,NP)
        ENDIF
        IF (INDEX(NAME, '*') .LE. 0) GOTO 130
    ENDIF
140 WRITE(6,*) ''
    WRITE(6,10) 'Restore any phase? /NONE/: '
    READ(5,20) NAME
    IF (LEN_TRIM(NAME) .GT. 0 .AND. NAME(1:4) .NE. 'NONE' .AND.
    & NAME(1:4) .NE. 'none') THEN
        CALL TQRESPPH(NAME,IWSG)
        IF(TQG2ERR(IERR)) THEN
            CALL TQRSERR
            WRITE(6,*) ' CANNOT RESTORE PHASE ',NAME
        ENDIF

```

```

ENDIF
CALL TQLISSF(PNAME,NP,IWSG)
IF(TQG2ERR(IERR)) THEN
  WRITE(6,*) 'CANNOT LIST PHASES IN THE DATABASE!'
  STOP
ELSE
  WRITE(6,*) 'Selected system phases: '
  WRITE(6,*) (PNAME(I),I=1,NP)
ENDIF
IF(INDEX(NAME,'*').LE.0) GOTO 140
ENDIF

CALL TQGDAT(IWSG)
IF(TQG2ERR(IERR)) THEN
  WRITE(6,*) 'CANNOT GET DATA FROM THE DATABASE! '
  STOP
ENDIF

P=1.D5
T=1000.D0
N=1.0D0
150 WRITE(6,*)''
WRITE(6,10)'Mole fraction of the second element: '
READ(5,*,ERR=910) X2
IF(X2.GT.1.0) THEN
  WRITE(6,*)'MOLE FRACTION SHOULD BE LESS THAN 1.0!'
  GOTO 150
ENDIF
CALL TQSETC('P',-1,-1,P,NCP,IWSG)
CALL TQSETC('T',-1,-1,T,NCT,IWSG)
CALL TQSETC('N',0,0,N,NCN,IWSG)
CALL TQGSCI(IEL2,EL2,IWSG)
CALL TQSETC('X',-1,IEL2,X2,NCX,IWSG)
CALL TQCEG(IWSG)
WRITE(6,*)' '
CALL TQSIO('OUTPUT',6)
CALL TQLE(IWSG)
WRITE(6,*)' '
WRITE(6,10)'MEW CALCULATION ON A NEW SYSTEM? /Y OR N/: '
READ(5,20)YES
IF(YES.EQ.'Y'.OR.YES.EQ.'y') GOTO 60
GOTO 990
910 WRITE(6,*)'NOT A NUMERICAL VALUE!'
990 CONTINUE
END

```

5.13 Example 13

```
C/*****
C/*
C/* This is a part of the source code samples demonstrating the use
C/* of the Thermodynamic calculation interface (TQ) for Thermo-Calc
C/*
C/* Copyright (1996, 1999, 2000) Foundation for Computational
C/* Thermodynamics
C/*
C/* This sample program demonstrates that the number of phases can
C/* increase due to the use of global minimization for equilibrium
C/* calculation during which additional composition set(s) can be
C/* added automatically if a miscibility gap is detected.
C/*
C/* Coded by:
C/*
C/* Qing Chen
C/* Thermo-Calc Software AB
C/* Bjornnasvagen 21
C/* S-113 47, Stockholm, Sweden
C/*
C/*****

    program tqex13

        implicit double precision (a-h,o-z)
        double precision P,N
        parameter (nwsg=80000)
        dimension iwsg(nwsg)
        character*32 phas1,pname(200)
        logical tqg2err

C
        WRITE(*,1)
1      FORMAT(
&' '/
&'This is a part of the source code samples demonstrating the use'/
&'of the Thermodynamic calculation interface (TQ) for Thermo-Calc'/
&' '/
&'Copyright (1996, 1999, 2000) Foundation for Computational '/
&'Thermodynamics'/
&' '/
&'This sample program demonstrates that the number of phases can'/
&'increase due to the use of global minimization for equilibrium'/
&'calculation during which additional composition set(s) can be'/
&'added automatically if a miscibility gap is detected.'/
&' '/
&'Qing Chen'/
&'Thermo-Calc Software AB'/
&'Bjornnasvagen 21, S-113 47, Stockholm, Sweden'/)

c... initiate workspace

        call tqini(nwsg,iwsg)
        if(tqg2err(ierr)) then
            print *, 'Cannot initialize the Thermo-Calc workspace!'
            stop
        endif

c... open database PTERN

        call tqopdb('PTERN',iwsg)
        if(tqg2err(ierr)) then
            print *, 'Database or database license not available!'
            stop
        endif

c... define element Fe

        call tqdefel('Fe',iwsg)
        if(tqg2err(ierr)) then
            print *, 'Error in defining element Fe!'

```

```

        stop
    endif

c... define element Cr

    call tqdefel('Cr',iwsg)
    if(tgg2err(ierr)) then
        print *, 'Error in defining element Cr!'
        stop
    endif

c... get data from the database

    call tqgdat(iwsg)
    if(tgg2err(ierr)) then
        print *, 'Error in getting data from the database!'
        stop
    endif

c... print phase names and their index
c... note no 2nd composition set added manually for BCC phase

    call prphase(iwsg)

c... set conditions

    P=101325.d0
    N=1.0D0
    xFCR=0.5d0
    call tqsetc('P',-1,-1, P ,numcon,iwsg)
    call tqsetc('N', 0, 0, N ,numcon,iwsg)
    call tqgsci(icmp2,'CR',iwsg)
    call tqsetc('X',-1,icmp2,xFCR,numcon,iwsg)

c... make calculations at 1000 K, where there should
c... be no miscibility gap exsiting. As a result, normal
c... calculation and global one should give same result.

    T=1000.D0
    call tqsetc('T',-1,-1, T ,numcon,iwsg)
    call tqce(' ', 0, 0, 0.d0,iwsg)
    if(tgg2err(ierr)) then
        print *, 'Error in calculating phase equilibrium!'
        stop
    endif

c... open the screen output unit and list equilibrium result
c... correct equilibbrium obtained.

    call tqsisio('output',6)
    call tqle(iwsg)

c... print phase names and their index
c... nothing changed

    call prphase(iwsg)

c... try global minimization

    call tqceg(iwsg)
    if(tgg2err(ierr)) then
        print *, 'Error in calculating phase equilibrium!'
        stop
    endif

c... list equilibrium result, which is the same as before

    call tqle(iwsg)

c... print phase names and their index. nothing changed

    call prphase(iwsg)

c... make calculations at 500 K, where BCC should split into
c... two BCC phases, one rich in Fe, and the other in Cr.
c... normal calculation will end up with a metastable equilibrium

```

```

c... because no 2nd composition set predefined.
c... global minimization algorithm can automatically generate
c... this 2nd composition set if found necessary and hence
c... guarantee that the stable equilibrium will be obtained.

      T=500.D0
      call tqsetc('T',-1,-1, T ,numcon,iwsg)
      call tqce(' ', 0, 0, 0.d0, iwsg)

c... list result. a metastable equilibrium instead stable equilibrium
c... is obtained

      call tqle(iwsg)

c... print phase names and their index. nothing changed

      call prphase(iwsg)

c... try global and list calculation result. the stable
c... equilibrium is obtained.

      call tqceg(iwsg)
      call tqle(iwsg)

c... print phase names and their index.
c... number of system phases increased by one due to the
c... appearance of BCC#2. note his "new" phase is added
c... to the bottom of the list and the index of "old" phases
c... is unaltered.

      call prphase(iwsg)

900  continue
      end

      subroutine prphase(iwsg)
      implicit double precision (a-h,o-z)
      dimension iwsg(*)
      character*32 phas1

      print *,''
      call tqgnp(npha,iwsg)
      print *, 'number of phases : ', npha
      do i=1,npha
         call tqgpn(i,phas1,iwsg)
         print *, 'index of ', phas1,' is ', i
      enddo

      return
      end

```