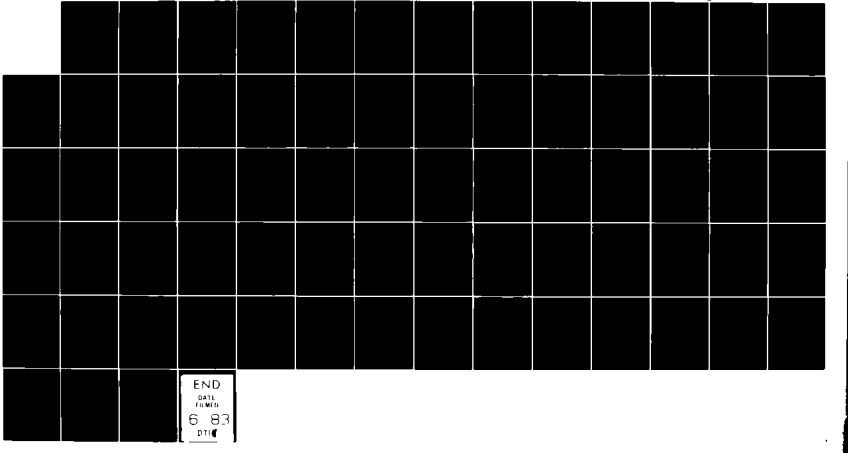
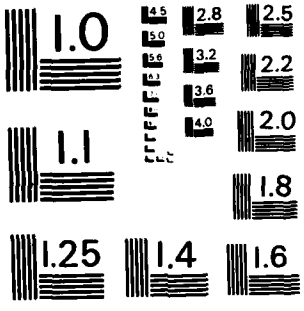
AD-A128 259 ADAPTATION OF COMPUTER PROGRAM FOR POURBAIX DIAGRAMS TO PERSONAL COMPUTERS(U) FLORIDA UNIV GAINESVILLE DEPT OF MATERIALS SCIENCE AND ENGINEERING E D VERINK APR 83 1/

UNCLASSIFIED N00014-82-C-0718 F/G 9/2 NL



END
DATE FILMED
6 83
DTIC



Final Report

Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217

ADAPTATION OF COMPUTER PROGRAM FOR POURBAIX DIAGRAMS TO PERSONAL COMPUTERS

Contract No. N00014-82-C-0718

Submitted by

Ellis D. Verink, Jr.

Department of Materials Science and Engineering
University of Florida
Gainesville, Florida 32611
(904) 392-1454
April 1983

83 04 27 022

Final Report
ONR N00014-82-C0718

Since publication of Pourbaix's classic treatise "Atlas of Electrochemical Equilibria in Aqueous Solutions" (Pergamon Press, Oxford, *CEBELCOR, Brussels*, 1966), there has been interest in computerizing the calculation of Pourbaix diagrams in order to avoid the tedious task of calculating these diagrams "long-hand" for complex systems. Reference 1 reports a computerized method developed at University of Florida and currently in use in a number of laboratories which have access to main-frame computing facilities. On the assumption that personal computers can be made more readily available to scientists and engineers, there was interest in adopting the existing program for use on an Apple II personal computer.

ONR Contract #N00014-82-C0718 provided support for this effort. The new program developed at University of Florida has the following capabilities and advantages:

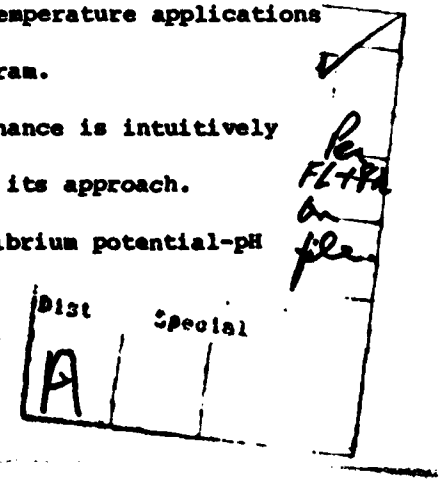
(1) The program is written in UCSD-Apple Pascal which makes it adaptable to many microcomputer systems.

(2) The method used to balance the equilibrium chemical equations allows the consideration of complicated metal-ion-water systems.

(3) The program can be easily modified for high temperature applications through the inclusion of an available subprogram.

(4) The method used to determine areas of predominance is intuitively obvious, rigorously correct, and continuum in its approach.

(5) The program can be applied to determine equilibrium potential-pH diagrams for even very complex systems.



The Program

The Fe-Cl-H_2O system provides a simple example of the application of the program. The metallic species which will be considered are shown in Table 1. Later, a more complex example will be given.

Table 1. DATA EMPLOYED IN COMPUTERIZED CALCULATION OF THE POTENTIAL vs pH DIAGRAM FOR THE Fe-Cl-H_2O SYSTEM AT 25°C

Species considered		Gibb's free energy of formation (in cals/M)	Assumed thermodynamic activity
Chemical notation	Computer notation		
Cl-	CL-	-31372.0	1.0×10^{-1}
Fe	FE	0.0	1.0
Fe_3O_4	FE304	-242700.0	1.0
Fe_2O_3	FE203	-177400.0	1.0
$HFeO_2^-$	HFEO2-	-90300.0	1.0×10^{-6}
$FeOH^{2+}$	FEOH++	-54830.0	1.0×10^{-6}
$Fe(OH)_2^-$	FE(OH)2+	-104700.0	1.0×10^{-6}
$FeCl_2(Aq)$	FECL2(AQ)	-81590.0	1.0×10^{-6}
$FeCl^{2+}$	FECL++	-34400.0	1.0×10^{-6}

Since computer line printers display only capital letters, it is necessary to develop a suitable convention for writing chemical equations entirely on one line (no subscripts) and in capital letters. Any reasonable, internally consistent, convention may be employed. Further reference to Table 1 will supply insight into the type of convention employed by the authors in this example. In order to avoid confusion in reading the balanced chemical equations, the computer places an asterisk between all coefficients and the chemical species to which they refer.

Input of Data

Data is entered into the microcomputer interactively. The user is prompted for each piece of data to be entered. In order to prevent crashing, all data is read as characters and converted to integer or real values as required.

The first piece of data to be entered is the temperature, in degrees Kelvin, to be used in Nernst equation calculations. Temperature corrections of free energy data may be done using a separate subprogram based on the methods of Criss & Cobble. (ref. 2)

The number of non-metallic ionic species to be considered is entered. In this example the number 1 is entered, since only Cl^- is considered. Pertinent ionic-species data in the form of each species' name, free energy of formation, activity, electrical charge and elemental composition is then input.

Immediately after the ionic species data the number of metallic species to be considered is entered (for the Fe-Cl-H_2O system this number is 8, since Cl^- would not be counted as a metallic species). Other pertinent data similar to that listed above for the ionic species is then input for each metallic species.

When data entry is completed the user is asked if the data requires correction. The user is able to select and modify any piece of data.

The data is written to the terminal or printer as specified by the user. The tabular form allows checking for accuracy and the user is again offered the opportunity to modify the entered data.

Chemical equation balancing and Equilibrium Line Determination

When considering the possible formation of N different metallic species

$$\frac{N\ (N - 1)}{2}$$

chemical equilibrium equations exist. The program balances chemical equations between all of the possible pairs of metallic species. The program takes all possible pairs of metallic species and balances the equations in terms of water, hydrogen ions, electrons, etc. The balanced chemical equations are used by the computer to determine and tabulate the equilibrium line equations by means of Nernst equation calculations. (For a simple explanation of the calculations involved, consult Ref. 3)

Areas of Predominance

Determining the areas of predominance on a potential-pH diagram requires a proper understanding of the information conveyed by each line on the diagram. Each line represents equilibrium coexistence between two metallic species (with certain assumptions regarding ionic activities). When any line is crossed one metallic species becomes more energetically favorable than another. Each line, therefore, delineates a boundary between two adjacent areas on the diagram. Each area represents the region (in potential vs pH coordinates) in which a particular species is thermodynamically stable and dominant. By simultaneously considering all of the lines which correspond to equilibria involving a given species, an area is traced out in which the species in question is "not excluded". This then is the area of predominance for that species.

When each species is considered (one at a time) in the above manner, a series of separate areas of predominance results. When these separate areas are fitted together, they form the finished theoretical potential-pH diagram without any overlaps.

A mathematical analog can be drawn by considering the equilibrium lines as inequalities. For example, in considering the equilibrium between Fe and Fe_3O_4, Fe is the predominant species at values of potential and pH less than those described by the equilibrium coexistence line between Fe and Fe_3O_4 (hence the term "inequality"). Thus a given species will exist whenever the potential and pH satisfy all of the line inequalities pertaining to the species. If no potential and pH simultaneously satisfies all of the inequalities of a given species, the species has no area of predominance.

The field of operations research has been developed by industrial and systems engineers to deal with such mathematical analogs. In operations research the area which mutually satisfies a set of linear inequalities is termed a "convex polygon".* Methods of determining convex polygons, therefore, lend themselves to determining areas of predominance on potential-pH diagrams.

Computer determination of areas of predominance

The equation balancing portion of the program is designed to write all electron-transfer equations as oxidation reactions. This results in

---

* Convex polygon-an area such that if a line segment is drawn between any two arbitrarily selected points in the polygon all points on the line segment will also lie within the polygon. (Ref. 3)

two important consequences. First, for all constant-pH lines the metallic species on the lower pH (left) side of the line will appear on the left side of the chemical equation. Second, for all non-vertical lines the metallic species which lie on the lower potential sides of the lines will also be on the left side of the chemical equation. By noting whether the various lines are vertical or nonvertical and on which side of the chemical equations the species occur, lines can be sorted into the following four categories: (1) minimum pH lines; (2) maximum pH lines; (3) potential, upper bound lines; and (4) potential, lower bound lines. The program uses this sorting technique to determine the convex polygon for each metallic species which simultaneously satisfies all the equilibrium line equations of each individual species. The resulting polygons are displayed to the user by the computer as a set of points which constitute the corners of each convex polygon. Plotting the determined points (expressed in potential and pH coordinates) and connecting them with straight lines results in a diagram like the one shown in Fig. 1 for Fe-Cl-H_2O.

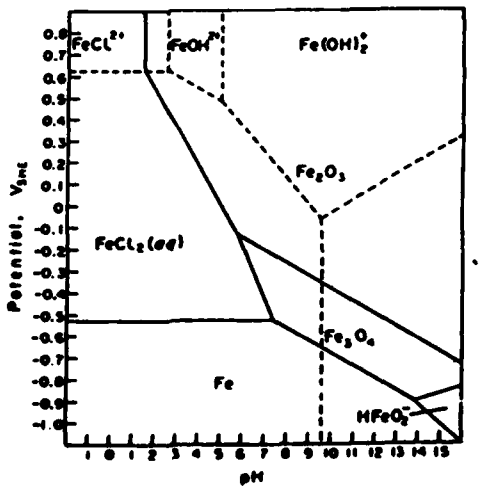


FIG. 1. Potential vs pH diagram for the Fe-Cl-H_2O system at 25°C as determined by computer considering the species shown in Table 1.

Determination of more Complex Systems

More complex metal-ion-water systems involving ionic species which

Calculation of potential-pH diagrams of metal-ion-water systems by computer

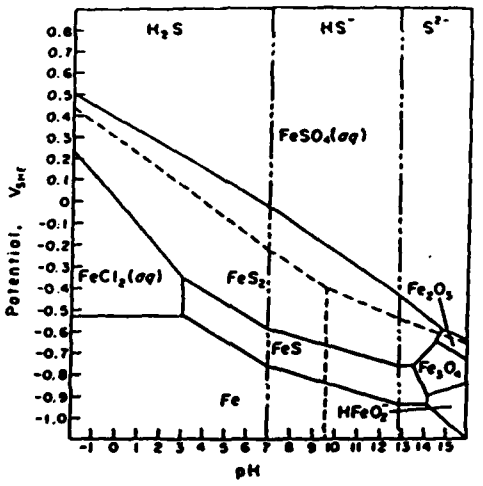


FIG. 2. Potential vs pH diagram for the Fe-Cl-S-H_2O system at 25°C as determined by the computer considering the species shown in Tables 1 and 2.

have their own potential-pH equilibria (e.g. H_2S, HS^-, S^{2-}) appear to violate the previous theory for determining regions of predominance. Such is not the case when the program is properly employed.

Addition of the species shown in Table 2 to those in Table 1 results in the computer determined diagram shown in Fig. 2. Note even though the region of predominance for FeS does not, as a whole, constitute a convex polygon, the portions of the region which lie in each of the individual areas of predominance for H_2S, HS^- and S^{2-} respectively, are in each case convex. Thus when it becomes necessary to formulate such a diagram it requires the initial computer calculation of more than

one diagram. In this case three diagrams were determined considering individually H_2S, HS^- and S^{2-} The appropriate portion of each of the three diagrams was then used to construct the final diagram

Table 2. DATA EMPLOYED IN COMPUTERIZED CALCULATION OF THE POTENTIAL vs pH DIAGRAM FOR THE Fe-Cl-S SYSTEM AT 25°C

Species considered		Gibb's free energy of formation (in cals/M)	Assumed thermodynamic activity
Chemical notation	Computer notation		
H_2S	H2S	-6660.0	1.0×10^{-1}
HS^-	HS-	2880.0	1.0×10^{-1}
S^{2-}	SS--	20500.0	1.0×10^{-1}
FeS (Solid)	FES (SOL)	-24000.0	1.0
FeS_2 (Solid)	FES2 (SOL)	-39900.0	1.0
$FeSO_4$ (Aq)	FESO4 (AQ)	-196820.0	1.0×10^{-6}

shown in Fig. 2 taking into account the domain of stability of each ionic species. (For simplicity the diagrams are drawn with straight-line segments meeting at the "equal dominance" lines for adjacent ionic species. Actually at such lines (e.g. HS^-/S^{2-}) the activities of both species are equal at the equal dominance line but one species rapidly predominates as one moves away from the line.) Using this technique even more complicated systems can be determined.

Summary of Program Capabilities

The interactive microcomputer version of the pH potential program is largely a translation of the WATFIV version and takes advantage of UCSD Pascal features.

The described computer program is arbitarily designed to consider a maximum of 35 metallic species and 10 non-metallic ionic species simultaneously. Redimensioning of the program matrices will allow for any desired increase or decrease in the program's capacity.

The structure of the program and the UCSD operating system facilitate the addition of extensions to the program through the independent development of new modules. It is hoped that support can be obtained for development of additional modules including a data maintenance routine that will record on disk a library of data for selected metal-ion-water systems and an accompanying graphics package that will permit interactive terminal and printer output of generated pH potential diagrams.

REFERENCES

1. M. H. Froning, M. E. Shanley, and E. D. Verink. "An Improved Method for Calculation of Potential -pH Diagrams of Metal-Ion-Water Systems by Computer", Corrosion Science, 1976, Vol. 16, pp. 371-377.
2. C. M. Criss and J. W. Cobble. J. Am. Chem. Soc., 86, 5394 (1964).
3. E. D. Verink, "Simplified Procedure for Constructing Pourbaix Diagrams" JEMMSE, Vol. 1, No. 3, Fall 1979, pp. 535-560.
4. H. M. Wagner, Principles of Operations Research, pg. 83, Prentice-Hall, N.J. (1969).

Attachments Include:

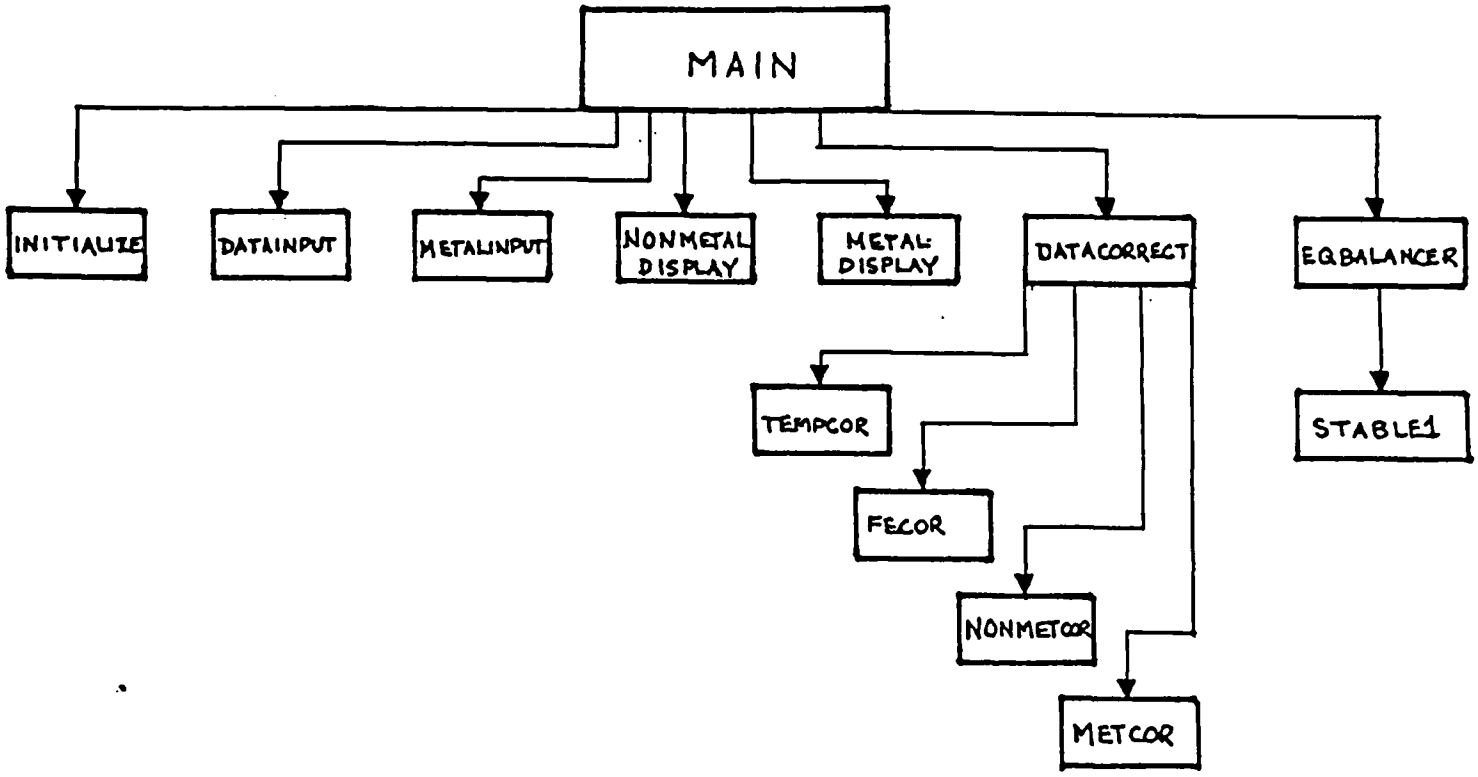
Flow chart

Modular Documentation

The Main Program with Accompanying Units

User's Guide

Example of Output



FLOW DIAGRAM OF PRIMARY PROGRAM MODULES

PH - POTENTIAL PROGRAM

MODULAR DOCUMENTATION

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Metal input (* part of unit in data*)

B. MODULE FUNCTION - Prompts user for metallic species data

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal:Number of metallic species (NP)
Name of metallic specie (SP [M, HH])
Free energy of formation (F [M])
Ionic activity (AM [M])
Electrical charge (Z [M])
Number of metal atoms (Metal [M])
Number of oxygen atoms (Oxygen [M])
Number of hydrogen atoms (Hydro [M])
Number of characteristic atoms for each ionic species (ION [I, M])

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

E. SUMMARY OF PROCESS - Sequentially prompts user for metallic species data. All real values read as characters by the procedure inreal, and are converted to real by CHAR to REAL. Integer values are also read as characters and are converted to integer by CHAR to INT. Name of species is read in by the procedure GETMET.

F. CALLS THESE MODULES - Procedures:INCHAR, INREAL, GETMET, LF.
Unit:Convert

G. CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Data input (* part of unit in data *)

B. MODULE FUNCTION - Prompts user for non-metallic ionic species data

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal:Temperature (T)
Free energy of formation of H_2O (FEH_2O)
Number of non-metallic ions (NION)
Name of species (Name [K, HH])
Free energy of formation (FION [1])
Ionic activity (AION [1])
Electrical charge (ZION [1])
Number of characteristic atoms (BION [1])
Number of Oxygen atoms (OXY [1])
Number of hydrogen atoms (HYD [1])

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

E. SUMMARY OF PROCESS - Sequentially prompts user for non-metallic ionic species data. All real values are read as characters by the procedure inreal and converted to real by CHAR to REAL. Integer values are also read as characters and are converted to integers by CHAR to INT. Name of species is read by the procedure GETNONMET.

F. CALLS THESE MODULES - Procedures:INCHAR, INREAL, GETNONMET, LF. Unit:Convert.

G. CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Initialize

B. MODULE FUNCTION - Initialization of index values, ionic species values, and species name array

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

E. SUMMARY OF PROCESS - Initializes program counter index to 1. Initializes FION [1], OXY [1], to 0. Initializes S, Name, X, SP to all blanks ("b").

F. CALLS THESE MODULES - None

G. CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Main program

B. MODULE FUNCTION - Controls execution of program

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Interactively prompts user for display option (printer or terminal)

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

E. SUMMARY OF PROCESS - Sequentially calls primary program modules for:
1. input of data
2. correction of data
3. display of data
4. correction of data
5. equation balancing

F. CALLS THESE MODULES - Initialize, data input, metal input, nonmetal display, metal display, data correct, Eq balancer

G. CALLED BY THESE MODULES - None

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Stable 1 (* part of unit stable *)

B. MODULE FUNCTION - Determining areas of predominance

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Listing of areas of predominance, line segment data. Listing of minimum and maximum constant pH line data.

E. SUMMARY OF PROCESS - Determine minimum pH line for metallic species (Stable 3). Determines maximum constant pH line for metallic species (Stable 4). Calculates upper bound lines to determine the line which supplies minimum potential for upper bound of stability region at minimum pH (UPBNDLNS). Calculates intersection points between previously determined upper bound lines and remaining upper bound lines (INTPTSHI). Calculates lower bound lines (LOWBNDLNS) and intersection points for lower bound lines (INTPTSLO). Tests upper and lower bounds to determine whether they intersect within the minimum and maximum pH lines. If they intersect, the points are determined and data is printed. Orders pH values at which either two upper bound lines or two lower bound lines intersect from PHMIN to PHMAX. Finds point where upper bound exceeds lower bound. Determines intersection point on maximum pH side. Prints data.

F. CALLS THESE MODULES - Procedures:Start, Set flags, Stable 2, Stable 3, Stable 4, UPBNDLNS, INTPTS H1, TEMP 1, LOWBNDLNS, INTPTSLO, TEMP 2, Stable 5, Stable 6, Stable 7, Stable 8, Stable 9, Stable 10, Stable 11.

G. CALLED BY THESE MODULES - Eq Balancer

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Eq Balancer

B. MODULE FUNCTION - Balances equations between all possible pairs of metallic species.

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal or Printer: Balanced equations. Equilibrium line equation.

E. SUMMARY OF PROCESS - Balances chemical equations between all possible pairs of metallic species. Takes all pairs of metallic species and balances them in terms of water, hydrogen ions, electrons, etc. NERNST equation calculations are then used to tabulate the equilibrium line equations. Procedure diagram then calls stable 1.

F. CALLS THESE MODULES - Procedures:EQB1, SUMNUM, Update, LOGSUM, FINDSIGN, EQB4, CALCEQEQ,
Diagram, Stable 1.

G. CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Nonmetcor (* part of unit in data *)

B. MODULE FUNCTION - Correction of non-metallic species data entered.

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal: Parameter to be modified. (Name, FION, AION, ZION, BION, OXY, HYD). New value for parameter specified.

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal: Menu of parameters. Old value of parameter.

E. SUMMARY OF PROCESS - Asks which ion to modify. Displays menu of options to be modified.

1. ion name
2. free energy of formation
3. activity
4. charge
5. number of characteristic atoms
6. number of oxygen atoms
7. number of hydrogen atoms

Writes old value of parameter, prompts for and reads new value.

F.CALLS THESE MODULES - Procedures:NMETC1, NMETC2, NMETC3, NMETC4, NMETC5, NMETC6, NMETC7,
6ETNONMET, INREAL, INCHAR.
Unit:Convert .

G.CALLED BY THESE MODULES - Data correct

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Metcor

B. MODULE FUNCTION - Correction of metallic species data entered.

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal:species and parameter to be modified (SP, F, AM, Z, Metal, Oxygen, Hydro, Ion).

New value for parameter specified.

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal:Menu of parameters. Old value of parameter.

E. SUMMARY OF PROCESS - Asks for species to be modified. Displays menu of options.

1. specie name
2. free energy of formation
3. activity
4. charge
5. number of metal atoms
6. number of oxygen atoms
7. number of hydrogen atoms
8. number of characteristic atoms

Writes old value of parameter, prompts for and reads new value.

F.CALLS THESE MODULES - Procedures:METC1, METC2, METC3, METC4, METC5, METC6, METC7, METC8,
GETMET, INCHAR, INREAL.
Unit:Convert

G.CALLED BY THESE MODULES - Data correct.

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Fecor (* part of unit in data *)

B. MODULE FUNCTION - Correction of free energy of formation of H_2O entered.

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal: FEH_2O

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Old FEH_2O value

E. SUMMARY OF PROCESS - Prints current value of FEH_2O. Prompts for and reads new value of FEH_2O.

F. CALLS THESE MODULES - Procedure:Inreal
Unit:Convert

G. CALLED BY THESE MODULES - Data correct

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Tempcor

B. MODULE FUNCTION - Correction of temperature entered

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal: T

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Old T value

E. SUMMARY OF PROCESS - Prints out current value of temperature.
Prompts for and reads new value of temperature.

F. CALLS THESE MODULES - Procedure:Inreal
Unit:Convert

G. CALLED BY THESE MODULES - Data correct

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Data correct (* part of unit in data *)

B. MODULE FUNCTION - User correction of data entered

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Data type to be modified (Opt.)

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

E. SUMMARY OF PROCESS - Prompts user for type of data that requires modification. The options are:

1. temperature
2. free energy of H_2O
3. non-metallic species data
4. metallic species data
5. data is now correct

Calls appropriate procedure.

F.CALLS THESE MODULES - Procedures: Tempcor, Fecor, Nonmetcor, Metcor

G.CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Metal display

B. MODULE FUNCTION - Displays metallic species data

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal or Printer: SP, F, AM, Z, Metal, Oxygen, Hydro, Ion

E. SUMMARY OF PROCESS - Displays metallic species data in tabular form to terminal or printer as specified by user.

F. CALLS THESE MODULES - Procedures: Dis 1 MET, Dis 2 MET, Dis 3 MET, Draw.

G. CALLED BY THESE MODULES - Main

MODULAR DESCRIPTION FORM

A. MODULAR NAME - Nonmetal display

B. MODULE FUNCTION - Displays non-metallic species data

C. INPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - None

D. OUTPUTS (ARGUMENTS, COMMON, FILE & TERMINAL) - Terminal or Printer: T, FEH2O, Name, FION, AION, ZION, BION OXY, HYD

E. SUMMARY OF PROCESS - Displays non-metallic species data in tabular form to terminal or printer as specified by user.

F. CALLS THESE MODULES - Procedures: ions out 1, ions out 2, draw.

G. CALLED BY THESE MODULES - Main

```
(*$S+*)
PROGRAM PHPOTENTIAL(INPUT,OUTPUT);

USES TRANSCEND,GETVALUE,CONVERT,INDATA,STABLE;
TYPE
  AONAME   =ARRAY[1..15,1..15] OF CHAR;
  MATRIX80=ARRAY[1..80] OF CHAR;
  AOINT10 =ARRAY[1..10] OF INTEGER;
  MAR2X595=ARRAY[1..2] OF ARRAY[1..595] OF REAL;
  AOCH15   =ARRAY[1..15] OF CHAR;

VAR
  C,COEF          :AOI15;
  JINDEX,INDEX    :INTEGER;
  MUN,A,NPP       :INTEGER;
  NUMB,NUM,NUME   :INTEGER;
  II,HH,KK        :INTEGER;
  J,K,L,I,M       :INTEGER;
  IONS,NONION     :INTEGER;
  SAVE,TEMP,S     :AOCH15;
  MAT             :MATRIX80;
  X               :AONAME;
  NUMF,SUMAIO     :REAL;
  ACT,ACT1,ACT2   :REAL;
  DELG,PH         :REAL;
  HE,ZAP1,ZAP2    :REAL;
  EN,ZAP3,ZAP4    :REAL;
  OPT1,NOPT       :CHAR;
  AB              :BOOLEAN;

PROCEDURE INITIALIZE;
  BEGIN
    INDEX:=1;
    FION[1]:=0.0;
    AION[1]:=1.0;
    ZION[1]:=0;
    BION[1]:=0;
    OXY[1]:=0;
    HYD[1]:=0;
    FOR I:=1 TO 15 DO
      BEGIN
        S[I]:=' ';
        FOR J:=1 TO 15 DO
          BEGIN
            NAME[I,J]:=' ';
            X[I,J]:=' ';
          END;
      END;
    FOR I:=1 TO 35 DO
      FOR J:=1 TO 15 DO
        SP[I,J]:=' ';
  END;


PROCEDURE FINDSIGN;
  BEGIN
    FOR J:=1 TO NUMB DO
      IF COEF[J]<>0 THEN
        BEGIN
          I:=I+1;
          S[I]:='+';
          IF I=1 THEN S[I]:=' ';
          IF COEF[J]<0 THEN S[I]:='-';
          IF J=(NION+3) THEN
            BEGIN
              S[I]:='=';
              C[I]:=COEF[J];
            END
          ELSE C[I]:=ABS(COEF[J]);
          FOR HH:=1 TO 15 DO X[I,HH]:=NAME[J,HH];
        END;
  END;
```

```
PROCEDURE UPDATE;
  BEGIN
    FOR I:=1 TO NUMB DO
      COEF[I]:=-COEF[I];
    J:=COEF[1];
    FOR HH:=1 TO 15 DO
      BEGIN
        NAME[1,HH]:=NAME[NION+3,HH];
        NAME[NION+3,HH]:=TEMP[HH];
      END;
    COEF[1]:=-COEF[NION+3];
    COEF[NION+3]:=-J;
    ACT:=ACT1;
    ACT1:=ACT2;
    ACT2:=ACT;
    LINE[1,MI]:=A;
    LINE[2,MI]:=M;
    DELG:=-DELG;
  END;

PROCEDURE SUMNUM;
  BEGIN
    MUN:=0;
    NUM:=0;
    NUMF:=0;
    FOR I:=1 TO NION DO
      BEGIN
        NUM:=NUM+COEF[I+2]*OXY[I];
        MUN:=MUN+COEF[I+2]*HYD[I];
        NUMF:=NUMF+COEF[I+2]*FION[I];
      END;
    COEF[2]:=OXYGEN[A]*METAL[M]-OXYGEN[M]*METAL[A]-NUM;
  COEF[NION+4]:=COEF[2]*2-HYDRO[A]*COEF[NION+3]+HYDRO[M]*COEF[1]+MUN;
  DELG:=COEF[NION+3]*F[A]-COEF[1]*F[M]-COEF[2]*FEH2O-NUMF;
  NUME:=0;
  FOR I:=1 TO NION DO
    NUME:=NUME+COEF[I+2]*ZION[I];
  COEF[NION+5]:=COEF[NION+4]-NUME-COEF[1]*Z[M]+COEF[NION+3]*Z[A];
END;

PROCEDURE EQWRITE;
  BEGIN
    KK:=1;
    WHILE MAT[KK]<>'#' DO
      BEGIN
        WRITE(FF,MAT[KK]);
        KK:=KK+1;
        IF MAT[KK]='=' THEN
          BEGIN
            WRITELN(FF,MAT[KK]);
            KK:=KK+1;
            WRITE(FF,'        !                                !');
            WRITE(FF,'       ');
          END;
      END;
    LNFD(1);
    WRITELN(FF,'        !                                !');
  END;


PROCEDURE SUB10;
  BEGIN
    LTYPE[MI]:=0;
    EQUAT[1,MI]:=0;
    EQUAT[2,MI]:=0;
    WRITE(FF,' ',MI:3,'    !     NO LINE GENERATED');
    WRITE(FF,'     !     ');
    EQWRITE;
  END;

PROCEDURE SUB11;
  BEGIN
    ZAP1:=DELG/(2.303*1.987*T);
    ZAP2:=COEF[NION+3]*LOG(ACT2);
    ZAP3:=COEF[1]*LOG(ACT1);
    PH:=(ZAP1+ZAP2-ZAP3-SUMAIO)/COEF[NION+4];
    LTYPE[MI]:=1;
    EQUAT[1,MI]:=PH;
    EQUAT[2,MI]:=0;
    WRITE(FF,' ',MI:3,'    !        PH=',PH:8:2);
    WRITE(FF,'        !   ');
    EQWRITE;
  END;
```

```
PROCEDURE CALCEQEQ;
   BEGIN
     IF(COEF[NION+5]=0) AND (COEF[NION+4]=0) THEN SUB10
     ELSE
       IF COEF[NION+5]=0 THEN SUB11
       ELSE
         BEGIN
           ZAP1:=1.9844*T/COEF[NION+5];
           ZAP2:=DELG/(COEF[NION+5]*23060.0);
           ZAP3:=COEF[NION+3]*LOG(ACT2);
           ZAP4:=COEF[1]*LOG(ACT1);
           HE:=-(COEF[NION+4]*ZAP1*0.0001);
           EN:=ZAP2+ZAP1*(ZAP3-ZAP4-SUMAIO)*0.0001;
           LTYPE[MI]:=2;
           EQUAT[1,MI]:=EN;
           EQUAT[2,MI]:=HE;
           IF HE=0 THEN
             BEGIN
               WRITE(FF,' ',MI:3,'    !  E=',EN:8:4);
               WRITE(FF,'                 !  ');
               EQWRITE;
             END
           ELSE
             IF HE<0 THEN
               BEGIN
                 HE:=-HE;
                 WRITE(FF,' ',MI:3,'    !  E=',EN:8:4);
                 WRITE(FF,' -',HE:7:4,'PH  !  ');
                 EQWRITE;
               END
             ELSE
               BEGIN
                 WRITE(FF,' ',MI:3,'    !  E=',EN:8:4);
                 WRITE(FF,' +',HE:7:4,'PH  !  ');
                 EQWRITE;
               END;
         END;
  FOR KK:=1 TO 80 DO MAT[KK]:=' ';
  FOR KK:=1 TO 15 DO NAME[1,KK]:=TEMP[KK];
  END;

PROCEDURE EQB4;
  BEGIN
    FOR KK:=1 TO 80 DO MAT[KK]:=' ';
    II:=1;
    FOR J:=1 TO I DO
      BEGIN
        IF J<>1 THEN
          BEGIN
            MAT[II]:=S[J];
            II:=II+1;
            MAT[II]:=' ';
            II:=II+1;
          END;
        IF C[J]<>1 THEN
          BEGIN
            GETVAL(C[J],SAVE);
            FOR KK:=1 TO 15 DO
              IF SAVE[KK]<>' ' THEN
                BEGIN
                  MAT[II]:=SAVE[KK];
                  II:=II+1;
                END;
            MAT[II]:='*';
            II:=II+1;
          END;
        FOR KK:=1 TO 15 DO
          SAVE[KK]:=X[J,KK];
        FOR KK:=1 TO 15 DO
          IF SAVE[KK]<>' ' THEN
            BEGIN
              MAT[II]:=SAVE[KK];
              II:=II+1;
            END;
        MAT[II]:=' ';
        II:=II+1;
        II:=II+1;
    END;
    MAT[II]:='@';
  END;
```

```
PROCEDURE LOGSUM;
  BEGIN
    I:=0;
    SUMAIO:=0.0;
    FOR KK:=1 TO NION DO
      BEGIN
        SUMAIO:=SUMAIO+COEF[KK+2]*LOG(AION[KK]);
      END;
  END;

PROCEDURE EQB1;
  BEGIN
    MI:=MI+1;
    IONLIN[MI]:=0;
    IF(AM[M]<>1) OR (AM[A]<>1) THEN IONLIN[MI]:=2;
    IF(AM[M]<>1) AND (AM[A]<>1) THEN IONLIN[MI]:=1;
    IF MI=INDEX THEN
      BEGIN
        IF INDEX<>1 THEN DRAWER(79);
        INDEX:=INDEX+70;
        LNFD(15);
        WRITELN(CHR(12));
        DRAWER(79);
        WRITELN(FF,'        !                                    !');
        WRITE(FF,'NUMBER !        EQUILIBRIUM        !');
        WRITELN(FF,'      BALANCED CHEMICAL EQUATION');
        WRITELN(FF,'        !                                    !');
        DRAWER(79);
      END;
    ACT1:=AM[M];
    ACT2:=AM[A];
    FOR HH:=1 TO 15 DO NAME[NION+3,HH]:=SP[A,HH];
    COEF[NION+3]:=METAL[M];
    COEF[1]:=METAL[A];
    FOR I:=1 TO NION DO
      COEF[I+2]:=COEF[NION+3]*ION[I,A]-COEF[1]*ION[I,M];
    SUMNUM;
    NUMB:=NION+5;
    FOR HH:=1 TO 15 DO TEMP[HH]:=NAME[1,HH];
    LINE[1,MI]:=M;
    LINE[2,MI]:=A;
    IF ((COEF[NION+5]=0)AND(COEF[NION+4]<0))OR(COEF[NION+5]<0) THEN
      UPDATE;
  END;

PROCEDURE EQBALANCER;
  BEGIN
    MI:=0;
    NAME[2,1]:='H';
    NAME[2,2]:='2';
    NAME[2,3]:='O';
    NAME[NION+4,1]:='H';
    NAME[NION+4,2]:='+';
    NAME[NION+5,1]:='E';
    NAME[NION+5,2]:='-';
    NPP:=NP-1;
    FOR M:=1 TO NPP DO
      BEGIN
        FOR HH:=1 TO 15 DO NAME[1,HH]:=SP[M,HH];
        K:=M+1;
        FOR A:=K TO NP DO
          BEGIN
            EQB1;
            LOGSUM;
            FINDSIGN;
            EQB4;
            CALCEQEQ;
          END;
      END;
  END;
```

```
PROCEDURE DIAGRAM;
  BEGIN
    DRAWER(78);
    IONS:=0;
    NONION:=0;
    FOR I:=1 TO NP DO
      IF AM[I]=1 THEN NONION:=NONION+1
      ELSE IONS:=IONS+1;
    IF NONION=0 THEN
      BEGIN
        LNFD(10);
        WRITELN(FF,' ** NO SOLID SPECIES DIAGRAM **');
      END
    ELSE
      BEGIN
        IONDIG:=0;
        STABLE1;
      END;

    IF IONS<=1 THEN
      BEGIN
        LNFD(10);
        WRITELN(FF,' *** NO DISSOLVED SPECIES DIAGRAM ***');
      END
    ELSE
      BEGIN
        IONDIG:=1;
        STABLE1;
      END;
  END;

PROCEDURE OUTMENU;
  BEGIN
    WHILE (OPT1<>'T') AND (OPT1<>'P') DO
      BEGIN
        WRITELN(CHR(12));
        IF AB=FALSE
          THEN WRITELN('           DATA OUTPUT OPTIONS  ')
          ELSE WRITELN('          EQUATION LISTING OPTIONS');
        LF(3);WRITELN('     P - PRINTER');
        LF(1);WRITELN('     T - TERMINAL');
        LF(2);WRITE('      ENTER OPTION:');
        READLN(OPT1);
        WRITELN(CHR(12));WRITELN;
      END;
    IF OPT1='P' THEN FILENAME:='PRINTER:'
    ELSE FILENAME:='CONSOLE:';
  END;
```

```
(*$N+*)
BEGIN
  REPEAT
    WRITELN(CHR(12));
    INITIALIZE;
    DATAINPUT;
    METALINPUT;
    WRITE(' MODIFY DATA ? [Y/N]: ');
    READLN(OPT1);
    IF OPT1='Y' THEN DATACORRECT;
    WRITELN(CHR(12));
      REPEAT
        IF NOPT='Y' THEN DATACORRECT;
          REPEAT
            LF(1);WRITE(' DISPLAY TABULATED DATA ? [Y/N]: ');
            READLN(OPT1);
            AB:=FALSE;
            IF OPT1='Y' THEN
              BEGIN
                OUTMENU;
                REWRITE(FFF,FILENAME);
                NONMETALDISPLAY;
                METALDISPLAY;
                CLOSE(FFF);
                OPT1:=' ';
                WHILE (OPT1<>'Y') AND (OPT1<>'N') DO
                  BEGIN
                    WRITE(' MODIFY DATA ? [Y/N]: ');
                    READLN(OPT1);
                  END;
                IF OPT1='Y' THEN DATACORRECT;
              END;
            AB:=TRUE;
            OUTMENU;
            REWRITE(FF,FILENAME);
            EQBALANCER;
            DIAGRAM;
            CLOSE(FF);
            LF(2);
            WRITELN(' EXECUTION COMPLETE....');
            LF(3);
            WRITE(' EXECUTE AGAIN ? [Y/N]: ');
            READLN(NOPT);
            IF NOPT='N' THEN EXIT(PROGRAM);
            INDEX:=1;
            WRITELN(CHR(12));
            LF(2);
            WRITE(' REPEAT WITH SAME DATA ? [Y/N]: ');
            READLN(NOPT);
          UNTIL NOPT='N';
        LF(2);
        WRITE(' MODIFY CURRENT DATA SET ? [Y/N]: ');
        READLN(NOPT);
      UNTIL NOPT='N';
  UNTIL NOPT='X';
END.
```

```
(*$S+*)
UNIT GETVALUE; INTRINSIC CODE 23 DATA 24;

 INTERFACE
   TYPE
     AO15=ARRAY[1..15] OF CHAR;
     AOI15=ARRAY[1..15] OF INTEGER;

   PROCEDURE GETVAL(X:INTEGER;
          VAR    CCHAR:AO15);

 IMPLEMENTATION

     VAR HH,I  :INTEGER;
         INT   :AOI15;
         VAL   :AOI15;
     PROCEDURE GETVAL;
       BEGIN
         FOR I:=1 TO 15 DO
           BEGIN
             INT[I]:=0;
             VAL[I]:=0;
             CCHAR[I]:=' ';
           END;
         HH:=1;
         WHILE X>= 1 DO
           BEGIN
             INT[HH]:=X;
             X:=X DIV 10;
             HH:=HH+1;
           END;
         VAL[HH]:=INT[HH];
         FOR I:=1 TO (HH-1) DO
           VAL[HH+1-I]:=INT[I]-10*INT[I+1];
         FOR I:=1 TO HH DO
           CCHAR[I]:=CHR(VAL[I]+ORD('0'));
         FOR I:=1 TO (HH-1) DO
           CCHAR[I]:=CCHAR[I+1];
         CCHAR[HH]:=' ';
       END;
       BEGIN
       END.
```

```
(*$S+*)
UNIT CONVERT, INTRINSIC CODE 16 DATA 17;
  INTERFACE
    TYPE
      AOCH3=ARRAY[1..3] OF CHAR;
      AOCH11=ARRAY[1..11] OF CHAR;
    VAR
      NUMBER  :INTEGER;
      CH3     :AOCH3;
      ERROR   :BOOLEAN;
      CH11    :AOCH11;
      RNUMBER:REAL;
      PROCEDURE CHARTOINT;
      PROCEDURE CHARTOREAL;
  IMPLEMENTATION
      VAR I     :INTEGER;
          BLANK:BOOLEAN;
          POWER:REAL;
  PROCEDURE CHARTOINT;
    BEGIN
      I:=3;
      NUMBER:=0;
      BLANK:=FALSE;
      POWER:=1.0;
      ERROR:=FALSE;
      WHILE (CH3[I]=' ') AND (I>1) DO I:=I-1;
      IF (I=1) AND (CH3[I]=' ')
        THEN
          BEGIN
            WRITELN;
            WRITELN('NUMBER EXPECTED');
            ERROR:=TRUE;
          END
        ELSE
          BEGIN
            WHILE (ERROR=FALSE) AND (I>0) AND (BLANK=FALSE) DO
              IF CH3[I] IN ['0'..'9','-'] THEN
                BEGIN
                  IF CH3[I]='-' THEN
                      NUMBER:=-NUMBER
                  ELSE NUMBER:=NUMBER+(ORD(CH3[I])-ORD('0'))*TRUNC(POWER)
                  POWER:=POWER*10.0;
                  I:=I-1;
                END
              ELSE
                IF CH3[I]=' ' THEN
                  BLANK:=TRUE
                ELSE
                  BEGIN
                    WRITELN;
                    WRITELN(' IMPROPER NUMBER');
                    ERROR:=TRUE;
                  END;
          END;
    END;
```

```
PROCEDURE CHARTOREAL;
  BEGIN
    I:=11;
    RNUMBER:=0;
    BLANK:=FALSE;
    POWER:=1.0;
    ERROR:=FALSE;
    WHILE (CH11[I]=' ') AND (I>1) DO I:=I-1;
    IF (I=1) AND (CH11[I]=' ') THEN
      BEGIN
        WRITELN;
        WRITELN(' NUMBER EXPECTED');
        ERROR:=TRUE;
      END
    ELSE
      BEGIN
        WHILE (ERROR=FALSE) AND (I>0) AND (BLANK=FALSE) DO
          IF CH11[I] IN ['0'..'9','-','.'] THEN
            IF CH11[I]='.' THEN
              BEGIN
                RNUMBER:=RNUMBER/POWER;
                POWER:=1.0;
                I:=I-1;
              END
            ELSE
              BEGIN
                IF CH11[I]='-' THEN
                  RNUMBER:=-RNUMBER
                ELSE
                  BEGIN
                    RNUMBER:=RNUMBER+(ORD(CH11[I])-ORD('0'))*POWER;
                    POWER:=POWER*10.0;
                  END;
                I:=I-1;
              END
          ELSE
            IF CH11[I]=' ' THEN
              BLANK:=TRUE
            ELSE
              BEGIN
                WRITELN;
                WRITELN(' IMPROPER NUMBER');
                ERROR:=TRUE;
              END;
      END;
  END;
BEGIN
END.
```

```
UNIT STABLE; INTRINSIC CODE 26 DATA 27;

  INTERFACE
     USES CONVERT,INDATA;
     TYPE
          AOINT2X595=ARRAY[1..2,1..595]OF INTEGER;
          AOINT595  =ARRAY[1..595] OF INTEGER;
          AOR2X595  =ARRAY[1..2,1..595]OF REAL;
     VAR  FF:TEXT;
          FILENAME:STRING;
            MI     :INTEGER;
            LINE   :AOINT2X595;
            LTYPE  :AOINT595;
            EQUAT  :AOR2X595;
           IONDIG  :INTEGER;
           IONLIN  :AOINT595;

    PROCEDURE STABLE1;
    PROCEDURE LNFD(QQ:INTEGER);
    PROCEDURE DRAWER(QQ:INTEGER);

  IMPLEMENTATION
    VAR
      NUMB1,LINNO,LINNN,LTYPE1              :ARRAY[1..34] OF INTEGER;
      I1,L1,L2,INDEX,JINDEX,I,N,NN,PASS     :INTEGER;
      PHMIND,PHMAXD,K,JJ,HH,J               :INTEGER;
      LINE1                                 :ARRAY[1..2,1..34] OF INTEGER;
      EQUAT1                                :ARRAY[1..2,1..34] OF REAL;
      LOWER,UPPER                           :ARRAY[1..4,1..34] OF REAL;
      PHTEST                                :ARRAY[1..35] OF REAL;
      INTERC,EMINU,EMINL,PH1,PH2            :REAL;
      ABC,POT1,POT2,POT11,POT22,E1,E2       :REAL;
      PHMIN,PHMAX,EMAXU,EMAXL,PHINT         :REAL;
      FLAG1,FLAG3,FLAG4,FLAG6,FLAGGER       :BOOLEAN;
      FLAG5,FLAG7,FLAG8,FLAG9,FLAG10        :BOOLEAN;

    PROCEDURE LNFD;
    VAR Q:INTEGER;
      BEGIN
        FOR Q:=1 TO QQ DO WRITELN(FF);
      END;

    PROCEDURE DRAWER;
    VAR Q:INTEGER;
      BEGIN
        FOR Q:=1 TO QQ DO WRITE(FF,'-');
        WRITELN(FF);
      END;

    PROCEDURE RITER(ARG1,ARG2:REAL;
                    ARG3,ARG4:REAL;
                         ARG5:INTEGER);
      BEGIN
        WRITE(FF,'                    !   ',ARG1:7:3,'   ! ',ARG2:6:2);
        WRITE(FF,' !   ',ARG3:7:3,'   ! ');
        WRITELN(FF,ARG4:6:2,' !   ',ARG5:3,'    !');
      END;

    PROCEDURE SUB1;
      BEGIN
        IF (POT11(POT1) AND (PHMIND()0) THEN
          RITER(POT1,UPPER[3,1],POT11,LOWER[3,1],NUMB1[PHMIND]);
        IF (POT11(POT1) AND (PHMIND=0) THEN
          BEGIN
            WRITE(FF,'                    !  '',POT1:7:3,'   ! ');
            WRITE(FF,UPPER[3,1]:6:2,' !    ',POT11:7:3);
            WRITELN(FF,'   ! ',LOWER[3,1]:6:2,' !           !');
          END;
        IF (POT22(POT2) AND (PHMAXD()0) THEN
          RITER(POT2,UPPER[4,N],POT22,LOWER[4,NN],NUMB1[PHMAXD]);
        IF (POT22(POT2) AND (PHMAXD=0) THEN
          BEGIN
            WRITE(FF,'                    !   ',POT2:7:3,'   ! ');
            WRITE(FF,UPPER[4,N]:6:2,' !    ',POT22:7:3);
            WRITELN(FF,'   ! ',LOWER[4,NN]:6:2,' !           !');
          END;
      END;
```

```
PROCEDURE STABLE11;
  BEGIN
    FOR I:=1 TO N DO
      BEGIN
        POT1:=UPPER[1,I]+UPPER[2,I]*UPPER[3,I];
        POT2:=UPPER[1,I]+UPPER[2,I]*UPPER[4,I];
        RITER(POT1,UPPER[3,I],POT2,UPPER[4,I],LINNO[I]);
      END;
    FOR I:=1 TO NN DO
      BEGIN
        POT1:=LOWER[1,I]+LOWER[2,I]*LOWER[3,I];
        POT2:=LOWER[1,I]+LOWER[2,I]*LOWER[4,I];
        RITER(POT1,LOWER[3,I],POT2,LOWER[4,I],LINNN[I]);
      END;
    POT1:=UPPER[1,1]+UPPER[2,1]*UPPER[3,1];
    POT11:=LOWER[1,1]+LOWER[2,1]*LOWER[3,1];
    POT11:=POT11+0.0001;
    POT2:=UPPER[1,N]+UPPER[2,N]*UPPER[4,N];
    POT22:=LOWER[1,NN]+LOWER[2,NN]*LOWER[4,NN];
    POT22:=POT22+0.0001;
    SUB1;
    FLAGGER:=TRUE;
  END;

PROCEDURE STABLE10;
  BEGIN
    LNFD(2);
    WRITE(FF,'                    ');
    DRAWER(57);
    WRITE(FF,'                   ! POTENTIAL 1 !  PH 1  !');
    WRITELN(FF,' POTENTIAL 2 !  PH 2  ! LINE NO. !');
    WRITE(FF,'                    ');
    DRAWER(57);
  END;

PROCEDURE STABLE9;
  BEGIN
    FLAG10:=FALSE;
    IF EMAXL)=EMAXU THEN
      BEGIN
        WHILE FLAG10=FALSE DO
          BEGIN
            IF (UPPER[4,N])PHTEST[K])AND(N)1) THEN N:=N-1;
            IF (LOWER[4,NN])PHTEST[K])AND(NN)1) THEN NN:=NN-1;
            K:=K-1;
            EMAXL:=LOWER[1,NN]+LOWER[2,NN]*PHTEST[K];
            EMAXU:=UPPER[1,N]+UPPER[2,N]*PHTEST[K];
            IF EMAXU)=EMAXL THEN FLAG10:=TRUE;
          END;
        PHINT:=(UPPER[1,N]-LOWER[1,NN])/(LOWER[2,NN]-UPPER[2,N]);
        LOWER[4,NN]:=PHINT;
        UPPER[4,N]:=PHINT;
      END;
  END;

PROCEDURE STABLE8;
  BEGIN
    IF FLAG8=FALSE THEN
      BEGIN
        PHINT:=(UPPER[1,1]-LOWER[1,1])/(LOWER[2,1]-UPPER[2,1]);
        LOWER[3,1]:=PHINT;
        UPPER[3,1]:=PHINT;
      END;
  END;

PROCEDURE SUB4(L:INTEGER);
  BEGIN
    LOWER[1,L]:=LOWER[1,L+1];
    LOWER[2,L]:=LOWER[2,L+1];
    LOWER[3,L]:=LOWER[3,L+1];
    LOWER[4,L]:=LOWER[4,L+1];
    LINNN[L]:=LINNN[L+1];
  END;

PROCEDURE SUB5(L:INTEGER);
  BEGIN
    UPPER[1,L]:=UPPER[1,L+1];
    UPPER[2,L]:=UPPER[2,L+1];
    UPPER[3,L]:=UPPER[3,L+1];
    UPPER[4,L]:=UPPER[4,L+1];
    LINNO[L]:=LINNO[L+1];
  END;
```

```
PROCEDURE STABLE7;
VAR L:INTEGER;
  BEGIN
    FLAG8:=FALSE;
    IF EMINL<EMINU THEN FLAG8:=TRUE;
    IF FLAG8=FALSE THEN
      BEGIN
        I:=1;
        FLAG9:=FALSE;
        WHILE FLAG9=FALSE DO
          BEGIN
            I:=I+1;
            ABC:=PHTEST[I];
            IF (ABC<0.000000001) AND (ABC)-0.00000001) THEN ABC:=0.0;
            EMINL:=LOWER[1,1]+LOWER[2,1]*ABC;
            EMINU:=UPPER[1,1]+UPPER[2,1]*ABC;
            IF(EMINU<=EMINL) AND (I=K) THEN FLAG7:=TRUE;
            IF (FLAG7=FALSE) THEN
              IF EMINL<EMINU THEN FLAG9:=TRUE;
            IF (FLAG7=FALSE) AND (FLAG9=FALSE) THEN
              BEGIN
                IF LOWER[4,1]<=PHTEST[I] THEN LINNN[1]:=0;
                IF UPPER[4,1]<=PHTEST[I] THEN LINNO[1]:=0;
                IF LINNN[1]=0 THEN
                  BEGIN
                    NN:=NN-1;
                    FOR L:=1 TO NN DO SUB4(L);
                  END;
                IF LINNO[1]=0 THEN
                  BEGIN
                    N:=N-1;
                    FOR L:=1 TO N DO SUB5(L);
                  END;
              END;
            IF I=35 THEN FLAG9:=TRUE;
          END;
      END;
  END;

PROCEDURE SUB3;
  BEGIN
    PHTEST[K]:=UPPER[4,I];
    IF (I=N) AND (JJ=NN) THEN FLAG6:=TRUE;
    IF FLAG6=FALSE THEN
      BEGIN
        IF I<N THEN I:=I+1;
        IF JJ<NN THEN JJ:=JJ+1;
        K:=K+1;
      END;
  END;


PROCEDURE STABLE6;
  BEGIN
    IF FLAG5=FALSE THEN
      BEGIN
        I:=1;
        JJ:=1;
        K:=2;
        PHTEST[1]:=PHMIN;
        FLAG6:=FALSE;
        WHILE FLAG6=FALSE DO
          BEGIN
            IF UPPER[4,I]<LOWER[4,JJ] THEN
              BEGIN
                PHTEST[K]:=UPPER[4,I];
                K:=K+1;
                I:=I+1;
              END
            ELSE
              IF LOWER[4,JJ]<UPPER[4,I] THEN
                BEGIN
                  PHTEST[K]:=LOWER[4,JJ];
                  K:=K+1;
                  JJ:=JJ+1;
                END
              ELSE
                IF UPPER[4,I]=LOWER[4,JJ] THEN SUB3;
          END;
      END;
  END;
```

```
PROCEDURE STABLE5;
  BEGIN
    EMINL:=LOWER[1,1]+LOWER[2,1]*PHMIN;
    EMINU:=UPPER[1,1]+UPPER[2,1]*PHMIN;
    EMAXL:=LOWER[1,NN]+LOWER[2,NN]*PHMAX;
    EMAXU:=UPPER[1,N]+UPPER[2,N]*PHMAX;
    IF (EMINL<EMINU) AND (EMAXL<>EMAXU) THEN FLAG5:=TRUE;
  END;

PROCEDURE SUB8;
  BEGIN
    LNFD(2);
    WRITE(FF,'                    ');
    DRAWER(55);
    WRITE(FF,'                   ! POTENTIAL 1 !  PH 1  !');
    WRITELN(FF,' POTENTIAL 2 !  PH 2  ! LINE NO. !');
    WRITE(FF,'                    ');
    DRAWER(55);
    IF N<>0 THEN
      FOR I:=1 TO N DO
        BEGIN
          POT1:=UPPER[1,I]+UPPER[2,I]*UPPER[3,I];
          POT2:=UPPER[1,I]+UPPER[2,I]*UPPER[4,I];
        RITER(POT1,UPPER[3,I],POT2,UPPER[4,I],LINNO[I]);
        END
    ELSE
      FOR I:=1 TO NN DO
        BEGIN
          POT1:=LOWER[1,I]+LOWER[2,I]*LOWER[3,I];
          POT2:=LOWER[1,I]+LOWER[2,I]*LOWER[4,I];
         RITER(POT1,LOWER[3,I],POT2,LOWER[4,I],LINNN[I]);
        END;
  END;

PROCEDURE INTERIMWRITER;
  BEGIN
    IF (N=0) OR (NN=0) THEN
      BEGIN
        IF N=0 THEN
          BEGIN
            LNFD(2);
            WRITELN(FF,'                NO UPPER LIMIT');
          END;
        IF NN=0 THEN
          BEGIN
            LNFD(2);
            WRITELN(FF,'                NO LOWER LIMIT');
          END;
        IF (N<>0) OR (NN<>0) THEN
          SUB8;
        IF PHMIN>-2.0 THEN
          BEGIN
            LNFD(1);
            WRITE(FF,'                   ');
            WRITE(FF,'MINIMUM PH = ',PHMIN:6:2,'  LINE NO. ');
            WRITELN(FF,NUMB1[PHMIND]:4);
          END;
        IF PHMIN<=-2.0 THEN
           WRITELN(FF,'                MINIMUM PH = ',PHMIN:6:2);
        IF PHMAX<16.0 THEN
          BEGIN
            WRITE(FF,'                MAXIMUM PH = ',PHMAX:6:2);
            WRITELN('PHMAXD= ',PHMAXD:4);
            WRITELN(FF,'  LINE NO. ',NUMB1[PHMAXD]:4);
          END;
        IF PHMAX>=16.0 THEN
          WRITELN(FF,'                MAXIMUM PH = ',PHMAX:6:2);
        FLAGGER:=TRUE;
      END;
  END;
```

```
PROCEDURE TEMP2;
  BEGIN
    FLAG3:=FALSE;
    NN:=NN+1;
    LOWER[1,NN]:=EQUAT1[1,L1];
    LOWER[2,NN]:=EQUAT1[2,L1];
    LOWER[3,NN]:=PH1;
    LINNN[NN]:=NUMB1[L1];
    LOWER[4,NN]:=PHMAX;
    IF PASS<1 THEN FLAG3:=TRUE;
    IF FLAG3=FALSE THEN
      BEGIN
        LOWER[4,NN]:=PH2;
        IF PH2>=PHMAX THEN LOWER[4,NN]:=PHMAX;
        IF PH2>=PHMAX THEN FLAG3:=TRUE;
        IF FLAG3=FALSE THEN
          BEGIN
            PH1:=PH2;
            E1:=EQUAT1[1,L2]+EQUAT1[2,L2]*PH1;
            L1:=L2;
          END;
      END;
  END;

PROCEDURE TEMP1;
  BEGIN
    FLAG1:=FALSE;
    N:=N+1;
    UPPER[1,N]:=EQUAT1[1,L1];
    UPPER[2,N]:=EQUAT1[2,L1];
    UPPER[3,N]:=PH1;
    LINNO[N]:=NUMB1[L1];
    UPPER[4,N]:=PHMAX;
    IF PASS<1 THEN FLAG1:=TRUE;
    IF FLAG1=FALSE THEN
      BEGIN
        UPPER[4,N]:=PH2;
        IF PH2>=PHMAX THEN UPPER[4,N]:=PHMAX;
        IF PH2>=PHMAX THEN FLAG1:=TRUE;
        IF FLAG1=FALSE THEN
          BEGIN
            PH1:=PH2;
            E1:=EQUAT1[1,L2]+EQUAT1[2,L2]*PH1;
            L1:=L2;
          END;
      END;
  END;

PROCEDURE LOWBNDLNS;
  BEGIN
    FOR I:=1 TO I1 DO
      IF(LINE1[2,I]=J) AND (LTYPE1[I]=2) THEN
        BEGIN
          PASS:=PASS+1;
          IF PASS=1 THEN
            BEGIN
              E1:=EQUAT1[1,I]+EQUAT1[2,I]*PH1;
              L1:=I;
            END
          ELSE
            BEGIN
              E2:=EQUAT1[1,I]+EQUAT1[2,I]*PH1;
              IF E2>=E1 THEN
                BEGIN
                  E1:=E2;
                  L1:=I;
                END;
            END;
        END;
  END;
```

```
PROCEDURE INTPTSLO;
  BEGIN
    PASS:=0;
    FOR I:=1 TO I1 DO
      BEGIN
        IF (LINE1[2,I]=J) AND (LTYPE1[I]=2) AND
          (EQUAT1[2,I]>EQUAT1[2,L1]) THEN
            BEGIN
              PASS:=PASS+1;
              IF PASS <= 1 THEN
                BEGIN
                  PH2:=EQUAT1[1,L1]-EQUAT1[1,I];
                  PH2:=PH2/(EQUAT1[2,I]-EQUAT1[2,L1]);
                  L2:=I;
                END
              ELSE
                BEGIN
                  INTERC:=EQUAT1[1,L1]-EQUAT1[1,I];
                  INTERC:=INTERC/(EQUAT1[2,I]-EQUAT1[2,L1]);
                  IF (INTERC<=PH2) AND (INTERC>=PH1) THEN
                    BEGIN
                      L2:=I;
                      PH2:=INTERC
                    END;
                END;
            END;
      END;
    TEMP2;
  END;

PROCEDURE INTPTSHI;
  BEGIN
    PASS:=0;
    FOR I:=1 TO I1 DO
      IF (LINE1[1,I]=J) AND (LTYPE1[I]=2) THEN
        IF EQUAT1[2,I]<EQUAT1[2,L1] THEN
          BEGIN
            PASS:=PASS+1;
            IF PASS<=1 THEN
              BEGIN
                PH2:=EQUAT1[1,L1]-EQUAT1[1,I];
                PH2:=PH2/(EQUAT1[2,I]-EQUAT1[2,L1]);
                L2:=I;
              END
            ELSE
              BEGIN
                INTERC:=EQUAT1[1,L1]-EQUAT1[1,I];
                INTERC:=INTERC/(EQUAT1[2,I]-EQUAT1[2,L1]);
                IF INTERC<=PH2 THEN
                  BEGIN
                    L2:=I;
                    PH2:=INTERC;
                  END;
              END;
          END;
    TEMP1;
  END;

PROCEDURE UPBNDLNS;
  BEGIN
    PASS:=0;
    FOR I:=1 TO I1 DO
      IF(LINE1[1,I]=J) AND (LTYPE1[I]=2) THEN
        BEGIN
          PASS:=PASS+1;
          IF PASS=1 THEN
            BEGIN
              E1:=EQUAT1[1,I]+EQUAT1[2,I]*PH1;
              L1:=I;
            END
          ELSE
            BEGIN
              E2:=EQUAT1[1,I]+EQUAT1[2,I]*PH1;
              IF E2<=E1 THEN
                BEGIN
                  E1:=E2;
                  L1:=I;
                END;
            END;
        END;
  END;
```

```
PROCEDURE STABLE4;
  BEGIN
    PHMAX:=16.0;
    PHMAXD:=0;
    FOR I:=1 TO I1 DO
      IF LINE1[1,I]=J THEN
        IF LTYPE1[I]=1 THEN
        IF EQUAT1[1,I]<PHMAX THEN
            BEGIN
              PHMAX:=EQUAT1[1,I];
              PHMAXD:=I;
            END;
  END;

PROCEDURE STABLE3;
  BEGIN
    PHMIN:=-2.0;
    PHMIND:=0;
    FOR I:=1 TO I1 DO
      IF (LINE1[2,I]=J) THEN
        IF (LTYPE1[I]=1) THEN
          IF (EQUAT1[1,I]>PHMIN) THEN
            BEGIN
              PHMIN:=EQUAT1[1,I];
              PHMIND:=I;
            END;
  END;

PROCEDURE STABLE2;
  BEGIN
    I1:=I1+1;
    LINE1[1,I1]:=LINE[1,I];
    LINE1[2,I1]:=LINE[2,I];
    LTYPE1[I1]:=LTYPE[I];
    EQUAT1[1,I1]:=EQUAT[1,I];
    EQUAT1[2,I1]:=EQUAT[2,I];
    NUMB1[I1]:=I;
  END;

PROCEDURE START;
  BEGIN
    INDEX:=1;
    JINDEX:=0;
  END;

PROCEDURE SETFLAGS;
  BEGIN
    FLAGGER:=FALSE;
    FLAG1:=FALSE; FLAG3:=FALSE;
    FLAG4:=FALSE; FLAG5:=FALSE;
    FLAG6:=FALSE; FLAG7:=FALSE;
    FLAG8:=FALSE; FLAG9:=FALSE;
    FLAG10:=FALSE;
  END;
```

```
PROCEDURE STABLE1;
  BEGIN
    START;
    FOR J:=1 TO NP DO
    BEGIN
      SETFLAGS;
      IF (IONDIC()1) OR (AM[J]()1.0) THEN
        BEGIN
          JINDEX:=JINDEX+1;
          N:=0;
          NN:=0;
          I1:=0;
          IF JINDEX=INDEX THEN WRITELN(CHR(12));
          IF JINDEX=INDEX THEN INDEX:=INDEX+5;
          LNFD(4);
  WRITELN(FF,'*************************************************');
          WRITE(FF,'*   AREA OF PREDOMINANCE FOR: ');
          FOR HH:=1 TO 15 DO WRITE(FF,SP[J,HH]);
          WRITE(FF,'  *');
          IF IONDIG=1 THEN WRITE(FF,' DISSOLVED SPECIES DIAGRAM');
          LNFD(1);
  WRITELN(FF,'*************************************************');
          FOR I:=1 TO MI DO
            IF (LINE[1,I]=J) OR (LINE[2,I]=J) THEN
              IF (IONDIC()1) OR (IONLIN[I]=1) THEN
                STABLE2;
          STABLE3;
          IF PHMIN(16 THEN
            BEGIN
              STABLE4;
              IF PHMAX)= PHMIN THEN
                BEGIN
                  PH1:=PHMIN;
                  PH2:=PHMAX;
                  UPBNDLNS;
                  IF PASS=0 THEN FLAG1:=TRUE;
                  IF FLAG1=FALSE THEN
                    BEGIN
                      IF PASS=1 THEN TEMP1;
                      WHILE FLAG1=FALSE DO INTPTSHI;
                    END;
                  PH1:=PHMIN;
                  PH2:=PHMAX;
                  PASS:=0;
                  LOWBNDLNS;
                  IF PASS=0 THEN FLAG3:=TRUE;
                  IF FLAG3=FALSE THEN
                    BEGIN
                      IF PASS=1 THEN TEMP2;
                      WHILE FLAG3=FALSE DO INTPTSLO;
                    END;
                  INTERIMWRITER;
                  IF FLAGGER=FALSE THEN
                    BEGIN
                      STABLE5;
                      STABLE6;
                      IF FLAG5=FALSE THEN STABLE7;
                      IF (FLAG7=FALSE) AND (FLAG5=FALSE) THEN STABLE8;
                      IF (FLAG7=FALSE) AND (FLAG5=FALSE) THEN STABLE9;
                      IF FLAG7=FALSE THEN STABLE10;
                      IF FLAG7=FALSE THEN STABLE11;
                    END;
                END;
            END;
          IF FLAGGER=FALSE THEN
            BEGIN
              LNFD(4);
              WRITE(FF,' NO AREA OF PREDOMINANCE');
              LNFD(2);
            END;
        END;
    END;
    LNFD(4);
  END;

BEGIN
END.
```

```
(*$S+*)

UNIT INDATA; INTRINSIC CODE 18 DATA 19;

  INTERFACE

    USES CONVERT;

    TYPE
    AON=ARRAY[1..15,1..15] OF CHAR;
    ASP=ARRAY[1..35,1..15] OF CHAR;
    AO10=ARRAY[1..10] OF REAL;
    AINT10=ARRAY[1..10] OF INTEGER;
    A35=ARRAY[1..35] OF REAL;
    AI35=ARRAY[1..35] OF INTEGER;
    M10X35=ARRAY[1..10] OF AI35;

    VAR
    NAME   :AON;
    SP     :ASP;
    T      :REAL;
    FEH2O  :REAL;
    NION   :INTEGER;
    FION   :AO10;
    AION   :AO10;
    ZION   :AINT10;
    BION   :AINT10;
    OXY    :AINT10;
    HYD    :AINT10;
    NP     :INTEGER;
    AM,F   :A35;
    Z      :AI35;
    METAL  :AI35;
    OXYGEN:AI35;
    HYDRO  :AI35;
    ION    :M10X35;
    FFF    :TEXT;

    PROCEDURE GETNONMET;
    PROCEDURE GETMET;
    PROCEDURE DATAINPUT;
    PROCEDURE METALINPUT;
    PROCEDURE DATACORRECT;
    PROCEDURE IONSOUT1;
    PROCEDURE IONSOUT2;
    PROCEDURE NONMETALDISPLAY;
    PROCEDURE DIS1MET;
    PROCEDURE METALDISPLAY;
    PROCEDURE LF(QQ:INTEGER);
    PROCEDURE INCHAR;
    PROCEDURE INREAL;

IMPLEMENTATION
  VAR I:INTEGER;
    Q,M:INTEGER;
      L:INTEGER;
   KK,J:INTEGER;
      K:INTEGER;
     HH:INTEGER;
    XYX:INTEGER;

PROCEDURE LF;
  BEGIN
    FOR L:=1 TO QQ DO WRITELN;
  END;

PROCEDURE DRAW(QQ:INTEGER);
  BEGIN
    FOR L:=1 TO QQ DO WRITE(FFF,'-');
    WRITELN(FFF);
  END;
```

```
PROCEDURE INCHAR;
  BEGIN
    FOR HH:=1 TO 3 DO CH3[HH]:=' ';
    HH:=1;
    WHILE (NOT EOLN) AND (HH<4) DO
      BEGIN
        READ(CH3[HH]);
        IF CH3[HH]=CHR(8) THEN
          BEGIN
            WRITE(CHR(8));
            HH:=HH-2;
          END;
        HH:=HH+1
      END;
    READLN;
    CHARTOINT;
  END;

PROCEDURE INREAL;
  BEGIN
    FOR HH:=1 TO 11 DO CH11[HH]:=' ';
    HH:=1;
    WHILE (NOT EOLN) AND (HH<12) DO
      BEGIN
        READ(CH11[HH]);
        IF CH11[HH]=CHR(8) THEN
          BEGIN
            WRITE(CHR(8));
            HH:=HH-2;
          END;
        HH:=HH+1;
      END;
    READLN;
    CHARTOREAL;
  END;

PROCEDURE GETNONMET;
  BEGIN
    HH:=1;
    WHILE NOT EOLN DO
      BEGIN
        READ(NAME[K,HH]);
        IF NAME[K,HH]=CHR(8) THEN
          BEGIN
            WRITE(CHR(8));
            HH:=HH-2;
          END;
        HH:=HH+1;
      END;
    READLN;
  END;

PROCEDURE GETMET;
  BEGIN
    HH:=1;
    WHILE NOT EOLN DO
      BEGIN
        READ(SP[M,HH]);
        IF SP[M,HH]=CHR(8) THEN
          BEGIN
            WRITE(CHR(8));
            HH:=HH-2;
          END;
        HH:=HH+1;
      END;
    READLN;
  END;
```

```
PROCEDURE DATAINPUT;
  BEGIN
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);WRITE('  ENTER TEMPERATURE (IN DEGREES KELVIN): ');
        INREAL;
      END;
    T:=RNUMBER;
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);WRITE('  ENTER GIBBS FREE ENERGY OF FORMATION');
        LF(1);WRITE('  OF WATER AT ',T:4:2,' DEGREES (CALS/MOLE): ');
        INREAL;
      END;
    FEH2O:=RNUMBER;
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
    BEGIN
      LF(1);WRITE('  ENTER THE NUMBER OF NON-METALLIC SPECIES: ');
      INCHAR;
    END;
    NION:=NUMBER;
    IF NION <> 0 THEN
      BEGIN
        FOR I:=1 TO NION DO
          BEGIN
            K:=I+2;
            LF(1);WRITE('   NAME OF NON-METALLIC SPECIE # ',I:2,': ');
            GETNONMET;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   FREE ENERGY OF FORMATION: ');
                INREAL;
              END;
            FION[I]:=RNUMBER;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   IONIC ACTIVITY: ');
                INREAL;
              END;
            AION[I]:=RNUMBER;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   ELECTRICAL CHARGE: ');
                INCHAR;
              END;
            ZION[I]:=NUMBER;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   NUMBER OF CHARACTERISTIC ATOMS: ');
                INCHAR;
              END;
            BION[I]:=NUMBER;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   NUMBER OF OXYGEN ATOMS: ');
                INCHAR;
              END;
            OXY[I]:=NUMBER;
            ERROR:=TRUE;
            WHILE ERROR=TRUE DO
              BEGIN
                LF(1);WRITE('   NUMBER OF HYDROGEN ATOMS: ');
                INCHAR;
              END;
            HYD[I]:=NUMBER;
            LF(1);
            WRITELN(CHR(12));
          END;
      END;
  END;
```

```
PROCEDURE METALINPUT;
  BEGIN
    WRITELN(CHR(12));
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);WRITE('  ENTER NUMBER OF METALLIC SPECIES: ');
        INCHAR;
      END;
    NP:=NUMBER;
    FOR M:=1 TO NP DO
      BEGIN
        LF(1);WRITE('   NAME OF METALLIC SPECIE # ',M:2,': ');
        GETMET;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   FREE ENERGY OF FORMATION: ');
            INREAL;
          END;
        F[M]:=RNUMBER;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   IONIC ACTIVITY: ');
            INREAL;
          END;
        AM[M]:=RNUMBER;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   ELECTRICAL CHARGE: ');
            INCHAR;
          END;
        Z[M]:=NUMBER;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   NUMBER OF METAL ATOMS: ');
            INCHAR;
          END;
        METAL[M]:=NUMBER;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   NUMBER OF OXYGEN ATOMS: ');
            INCHAR;
          END;
        OXYGEN[M]:=NUMBER;
        ERROR:=TRUE;
        WHILE ERROR=TRUE DO
          BEGIN
            LF(1);WRITE('   NUMBER OF HYDROGEN ATOMS: ');
            INCHAR;
          END;
        HYDRO[M]:=NUMBER;
        IF NION > 0 THEN
          BEGIN
            FOR I:=1 TO NION DO
              BEGIN
                ERROR:=TRUE;
                WHILE ERROR=TRUE DO
                  BEGIN
                    LF(1);
                    WRITELN('  ENTER NUMBER OF CHARACTERISTIC ATOMS');
                    WRITE('  FOR IONIC SPECIES # ',I:2,': ');
                    INCHAR;
                  END;
                ION[I,M]:=NUMBER;
              END;
          END
        ELSE ION[1,M]:=0;
        LF(1);
        WRITELN(CHR(12));
      END;
  END;
```

```
PROCEDURE IONSOUT1;
  BEGIN
    WRITE(FFF,'  ',I:2,'    ! ');
    FOR HH:=1 TO 8 DO WRITE(FFF,NAME[I+2,HH]);
    WRITE(FFF,'!  ',FION[I]:10:1,'  !');
    WRITELN(FFF,AION[I]:2:6,' ! ',ZION[I]:4,'    !');
  END;

PROCEDURE IONSOUT2;
  BEGIN
    WRITE(FFF,'  ',I:2,'    !      ',BION[I]:3,'          !');
    WRITELN(FFF,'    ',OXY[I]:2,'    !      ',HYD[I]:2,'     !');
  END;

PROCEDURE NONMETALDISPLAY;
  BEGIN
    WRITELN(CHR(12));
    WRITELN(FFF);
    WRITELN(FFF,'*****************************':39);
    WRITELN(FFF,'*  TEMPERATURE = ':29,T:5:2,'  *');
    WRITELN(FFF,'*****************************':39);
    WRITELN(FFF);
    IF NION=0 THEN
      BEGIN
        WRITELN(FFF);WRITELN(FFF);
        WRITE(FFF,'***  NO NON-METALLIC IONIC SPECIES ');
        WRITE(FFF,'ARE CONSIDERED  ***');
        NION:=1;
      END
    ELSE
      BEGIN
        WRITELN(FFF);
        DRAW(53);
        WRITE(FFF,' ION    !  IONIC  ! FREE ENERGY  ');
        WRITELN(FFF,'!  IONIC    !          !');
        WRITE(FFF,'NUMBER ! SPECIES ! OF FORMATION ');
        WRITELN(FFF,'! ACTIVITY ! CHARGE !');
        DRAW(53);
        FOR I:=1 TO NION DO
          IONSOUT1;
        FOR I:=1 TO 3 DO WRITELN(FFF);
        DRAW(45);
        WRITELN(FFF,' ION    ! CHARACTERISTIC ! OXYGEN ! HYDROGEN !');
        WRITELN(FFF,'NUMBER !      ATOMS       ! ATOMS  !  ATOMS   !');
        DRAW(45);
        FOR I:=1 TO NION DO
          IONSOUT2;
      END;
  END;

PROCEDURE DIS1MET;
  BEGIN
    WRITE(FFF,'  ');
    FOR HH:=1 TO 15 DO WRITE(FFF,SP[I,HH]);
    WRITE(FFF,'!  ',METAL[I]:2,'    !    ',OXYGEN[I]:2);
    WRITE(FFF,'    !     ',HYDRO[I]:2,'     !');
    WRITELN(FFF);
  END;

PROCEDURE DIS2MET;
  BEGIN
    WRITE(FFF,'  ');
    FOR HH:=1 TO 15 DO WRITE(FFF,SP[I,HH]);
    WRITE(FFF,'!');
    IF Q=1 THEN
      FOR J:=1 TO 5 DO
        IF J<=NION THEN
          WRITE(FFF,'  ',ION[J,I]:2,'  !');
    IF Q=2 THEN
      FOR J:=6 TO 10 DO
        IF J<=NION THEN
          WRITE(FFF,'  ',ION[J,I]:2,'  !');
    IF Q=3 THEN
      FOR J:=11 TO 15 DO
        IF J<=NION THEN
          WRITE(FFF,'  ',ION[J,I]:2,'  !');
    IF Q=4 THEN
      FOR J:=16 TO 20 DO
        IF J<=NION THEN
          WRITE(FFF,'  ',ION[J,I]:2,'  !');
    WRITELN(FFF);
  END;
```

```
PROCEDURE DIS3MET;
  BEGIN
    WRITE(FFF,'                    ');
    IF Q=1 THEN
      WRITE(FFF,'! ION1 ! ION2 ! ION3 ! ION4 ! ION5 !');
    IF Q=2 THEN
      WRITE(FFF,'! ION6 ! ION7 ! ION8 ! ION9 ! ION10!');
    IF Q=3 THEN
      WRITE(FFF,'! ION11! ION12! ION13! ION14! ION15!');
    IF Q=4 THEN
      WRITE(FFF,'! ION16! ION17! ION18! ION19! ION20!');
    WRITELN(FFF);
    DRAW(53);
  END;

PROCEDURE METALDISPLAY;

  BEGIN
    WRITELN(CHR(12));
    WRITELN(FFF);WRITELN(FFF);
    DRAW(53);
   WRITELN(FFF,'                          ! FREE ENERGY  ! SPECIES  !         !');
   WRITELN(FFF,'METALLIC SPECIES ! OF FORMATION ! ACTIVITY ! CHARGE !');
    DRAW(53);
    FOR I:=1 TO NP DO
      BEGIN
        WRITE(FFF,'  ');
        FOR HH:=1 TO 15 DO WRITE(FFF,SP[I,HH]);
        WRITE(FFF,'! ',F[I]:11:1,'  !',AM[I]:2:6);
        WRITELN(FFF,' !   ',Z[I]:2,'   !');
      END;
    FOR I:=1 TO 3 DO WRITELN(FFF);
    DRAW(46);
    WRITELN(FFF,'                    ! METAL ! OXYGEN ! HYDROGEN !');
    WRITELN(FFF,'METALLIC SPECIES ! ATOMS ! ATOMS  !  ATOMS   !');
    DRAW(46);
    FOR I:=1 TO NP DO DIS1MET;
    FOR I:=1 TO 3 DO WRITELN(FFF);
    L:=TRUNC((NION-1)/5.0)+1;
    FOR Q:=1 TO L DO
      BEGIN
        WRITELN(FFF);
        DRAW(53);
        WRITE(FFF,'                    ');
        WRITELN(FFF,'  ! NUMBER OF CHARACTERISTIC ATOMS   !');
        WRITE(FFF,'METALLIC SPECIES !     OF EACH IONIC SPECIES');
        WRITELN(FFF,'         !');
        DIS3MET;
        FOR I:=1 TO NP DO DIS2MET;
        FOR I:=1 TO 3 DO WRITELN(FFF);
      END;
  END;

  PROCEDURE NMETC7;
    BEGIN
      WRITE('     MODIFICATION OF NUMBER OF HYDROGEN ATOMS FOR ');
      FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
      LF(3);
      WRITELN(' CURRENT VALUE IS ',HYD[I]:3);
      ERROR:=TRUE;
      WHILE ERROR=TRUE DO
        BEGIN
          LF(1);
          WRITE(' DESIRED VALUE: ');
          INCHAR;
        END;
      HYD[I]:=NUMBER;
    END;
```

```
PROCEDURE NMETC6;
  BEGIN
    WRITE('     MODIFICATION OF NUMBER OF OXYGEN ATOMS FOR ');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',OXY[I]:3);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    OXY[I]:=NUMBER;
  END;

PROCEDURE NMETC5;
  BEGIN
    WRITE('     MODIFICATION OF NUMBER OF CHARACTERISTIC ATOMS');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',BION[I]:3);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    BION[I]:=NUMBER;
  END;

PROCEDURE NMETC4;
  BEGIN
    WRITE('     MODIFICATION OF CHARGE FOR ');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',ZION[I]:4);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    ZION[I]:=NUMBER;
  END;

PROCEDURE NMETC3;
  BEGIN
    WRITE('     MODIFICATION OF IONIC ACTIVITY FOR ');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',AION[I]:2:6);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INREAL;
      END;
    AION[I]:=RNUMBER;
  END;

PROCEDURE NMETC2;
  BEGIN
    WRITE('     MODIFICATION OF FREE ENERGY FOR ');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',FION[I]:10:2);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INREAL;
      END;
    FION[I]:=RNUMBER;
  END;
```

```
PROCEDURE NMETC1;
  BEGIN
    WRITELN('      MODIFICATION OF SPECIES NAME');
    LF(2);
    WRITE(' CURRENT NAME IS ');
    FOR K:=1 TO 15 DO WRITE(NAME[KK,K]);
    LF(2);
    WRITE(' DESIRED NAME: ');
    FOR K:=1 TO 15 DO NAME[I,K]:=' ';
    K:=KK;
    GETNONMET;
  END;

PROCEDURE NONMETCOR;
VAR FLAG2:BOOLEAN;
  BEGIN
    WRITELN(CHR(12));
    WRITELN('      MODIFICATION OF NON-METALLIC SPECIES DATA');
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(3);
        WRITE(' ENTER IONIC SPECIE NUMBER: ');
        INCHAR;
        IF (NUMBER<1) OR (NUMBER>NION) THEN ERROR:=TRUE;
      END;
    I:=NUMBER;
    KK:=I+2;
    REPEAT
      FLAG2:=FALSE;
      LF(2);
      WRITELN(' PARAMETER TO BE MODIFIED - ');
      LF(1);
      WRITELN('     1 - ION NAME');
      WRITELN('     2 - FREE ENERGY OF FORMATION');
      WRITELN('     3 - ACTIVITY');
      WRITELN('     4 - CHARGE');
      WRITELN('     5 - NUMBER OF CHARACTERISTIC ATOMS');
      WRITELN('     6 - NUMBER OF OXYGEN ATOMS');
      WRITELN('     7 - NUMBER OF HYDROGEN ATOMS');
      ERROR:=TRUE;
      WHILE ERROR=TRUE DO
        BEGIN
          LF(1);
          WRITE(' ENTER OPTION: ');
          INCHAR;
        END;
      J:=NUMBER;
      IF (J<1) OR (J>7) THEN FLAG2:=TRUE;
    UNTIL FLAG2=FALSE;
    WRITELN(CHR(12));
    CASE J OF
         1: NMETC1;
         2: NMETC2;
         3: NMETC3;
         4: NMETC4;
         5: NMETC5;
         6: NMETC6;
         7: NMETC7;
    END;
  END;
```

```
PROCEDURE METC8;
  BEGIN
    WRITE('      MODIFICATION OF ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(3);
        WRITE(' CHANGE VALUE FOR WHAT ION NUMBER?: ');
        INCHAR;
        IF (NUMBER<1) OR (NUMBER>NION) THEN ERROR:=TRUE;
        END;
    K:=NUMBER;
    LF(2);
    WRITELN(' CURRENT VALUE IS ',ION[K,I]:2);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    ION[K,I]:=NUMBER;
  END;

PROCEDURE METC7;
  BEGIN
    WRITE('      MODIFICATION OF NUMBER OF HYDROGEN ATOMS FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT NAME IS ',HYDRO[I]:3);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    HYDRO[I]:=NUMBER;
  END;

PROCEDURE METC6;
  BEGIN
    WRITE('      MODIFICATION OF NUMBER OF OXYGEN ATOMS FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',OXYGEN[I]:3);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    OXYGEN[I]:=NUMBER;
  END;

PROCEDURE METC5;
  BEGIN
    WRITE('      MODIFICATION OF NUMBER OF METAL ATOMS FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',METAL[I]:3);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    METAL[I]:=NUMBER;
  END;
```

```
PROCEDURE METC4;
  BEGIN
    WRITE('     MODIFICATION OF CHARGE FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',Z[I]:2);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INCHAR;
      END;
    Z[I]:=NUMBER;
  END;

PROCEDURE METC3;
  BEGIN
    WRITE('     MODIFICATION OF ACTIVITY FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',AM[I]:2:6);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INREAL;
      END;
    AM[I]:=RNUMBER;
  END;

PROCEDURE METC2;
  BEGIN
    WRITE('     MODIFICATION OF FREE ENERGY FOR ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(3);
    WRITELN(' CURRENT VALUE IS ',F[I]:10:2);
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INREAL;
      END;
    F[I]:=RNUMBER;
  END;

PROCEDURE METC1;
  BEGIN
    WRITELN('     MODIFICATION OF SPECIE NAME');
    LF(3);
    WRITE(' CURRENT NAME IS ');
    FOR K:=1 TO 15 DO WRITE(SP[I,K]);
    LF(2);
    WRITE(' DESIRED VALUE: ');
    FOR K:=1 TO 15 DO SP[I,K]:=' ';
    M:=I;
    GETMET;
  END;
```

```
PROCEDURE METCOR;
  VAR FLAG1:BOOLEAN;
  BEGIN
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' ENTER METALLIC SPECIE NUMBER: ');
        INCHAR;
        IF (NUMBER<1) OR (NUMBER>N?) THEN ERROR:=TRUE;
      END;
    I:=NUMBER;
    REPEAT
      FLAG1:=FALSE;
      LF(2);
      WRITELN(' PARAMETER TO BE MODIFIED -');
      LF(1);
      WRITELN('     1 - SPECIES NAME');
      WRITELN('     2 - FREE ENERGY OF FORMATION');
      WRITELN('     3 - ACTIVITY');
      WRITELN('     4 - CHARGE');
      WRITELN('     5 - NUMBER OF METAL ATOMS');
      WRITELN('     6 - NUMBER OF OXYGEN ATOMS');
      WRITELN('     7 - NUMBER OF HYDROGEN ATOMS');
      WRITELN('     8 - NUMBER OF CHARACTERISTIC ATOMS');
      ERROR:=TRUE;
      WHILE ERROR=TRUE DO
        BEGIN
          LF(1);
          WRITE(' ENTER OPTION: ');
          INCHAR;
        END;
      J:=NUMBER;
      IF (J<1) OR (J>8) THEN FLAG1:=TRUE;
    UNTIL FLAG1=FALSE;
    WRITELN(CHR(12));
    CASE J OF
          1: METC1;
          2: METC2;
          3: METC3;
          4: METC4;
          5: METC5;
          6: METC6;
          7: METC7;
          8: METC8;
    END;
  END;

PROCEDURE TEMPCOR;
BEGIN
  WRITELN(' CURRENT TEMPERATURE VALUE IS ',T:6:2,' DEGREES KELVIN');
  ERROR:=TRUE;
  WHILE ERROR=TRUE DO
    BEGIN
      LF(1);
      WRITE(' DESIRED VALUE: ');
      INREAL;
    END;
  T:=RNUMBER;
END;
```

```
PROCEDURE FECOR;
  BEGIN
    WRITELN(' CURRENT FREE ENERGY IS ',FEH2O:8:2,' CALS/MOLE');
    ERROR:=TRUE;
    WHILE ERROR=TRUE DO
      BEGIN
        LF(1);
        WRITE(' DESIRED VALUE: ');
        INREAL;
      END;
    FEH2O:=RNUMBER;
  END;

PROCEDURE DATACORRECT;
VAR OPT:CHAR;
  BEGIN
    WRITELN(CHR(12));
      REPEAT
        WRITELN(CHR(12));
        LF(1);
        WRITELN(' MODIFY WHICH VALUE?');
        LF(1);
        WRITELN('     1 - TEMPERATURE');
        WRITELN('     2 - FREE ENERGY OF H2O');
        WRITELN('     3 - NON-METALLIC ION DATA');
        WRITELN('     4 - METALLIC SPECIES DATA');
        WRITELN('     5 - ALL DATA IS NOW CORRECT');
        LF(1);
        WRITE(' ENTER OPTION: ');
        READLN(OPT);
        WRITELN(CHR(12));
        CASE OPT OF
              '1': TEMPCOR;
              '2': FECOR;
              '3': NONMETCOR;
              '4': METCOR;
              '5': OPT:='N';
            END;
        LF(2);
      UNTIL OPT='N';
  END;

BEGIN
END.
```

USER'S GUIDE FOR THE

INTERACTIVE PH - POTENTIAL PROGRAM

This introductory guide should be read prior to operation of the program.

"When all else fails, push the reset button...

When the reset button fails, pull the plug..."

Eginhard Muth

INTRODUCTION

The microcomputer version of the PH - POTENTIAL PROGRAM is easy to use, even for those with little or no computer experience. This guide explains, in simple terms, how to get started and what the operator must do to use the program effectively.

GETTING STARTED

Getting started is easy. Simply place the diskettes in the proper disk drives and turn on the power. The program will begin execution automatically. The following explains the process in detail.

1 - Open disk drive #1. Insert the diskette labeled POURBAIX 1 into disk drive #1. (For the beginner, hold the diskette in either hand, palm up, and with your thumb on the label, gently slide in the diskette and close the disk drive door.)

2 - Repeat step 1 for disk drive #2 and the diskette labeled POURBAIX 2.

NOTE: For single drive systems, the program can be executed with only POURBAIX 1.

3 - Turn on the power for the computer. If the power was already on turn the computer off and then on again. The disk drives will make whirring noises as the computer prepares for operation. In just a few seconds, the program will begin execution automatically and prompt for inputs.

ENTERING THE DATA

The program sequentially prompts for the required data. Simply type in the name or number as asked. If the computer responds with "IMPROPER NUMBER" see Appendix 1 for possible causes.

Data to be Entered (R = Real, I = Integer, C = Character.)

1 - TEMPERATURE (degrees Kelvin). (R)

2 - GIBB'S FREE ENERGY OF FORMATION OF H2O AT THE TEMPERATURE BEING CONSIDERED (in calories per mole). (R)

3 - NUMBER OF NON-METALLIC IONS CONSIDERED. (I)

For each non -metallic ionic species considered:

4A - NAME OF NON-METALLIC ION. (C)

4B - FREE ENERGY OF FORMATION (in calories per mole). (R)

4C - THERMODYNAMIC ACTIVITY. (R)

4D - ELECTRICAL CHARGE. (I)

4E - NUMBER OF CHARACTERISTIC ATOMS (atoms other than oxygen or hydrogen). (I)

4F - NUMBER OF OXYGEN ATOMS IN THE IONIC SPECIES. (I)

4G - NUMBER OF HYDROGEN ATOMS IN THE IONIC SPECIES. (I)

5 - NUMBER OF METALLIC SPECIES CONSIDERED. (I)

For each metallic species considered:

6A - NAME OF METALLIC SPECIES. (C)

6B - GIBB'S FREE ENERGY OF FORMATION (calories per mole). (R)

6C - THERMODYNAMIC ACTIVITY. (R)

6D - ELECTRICAL CHARGE. (I)

6E - NUMBER OF METAL ATOMS IN THE SPECIES. (I)

6F - NUMBER OF OXYGEN ATOMS. (I)

6G - NUMBER OF HYDROGEN ATOMS. (I)

6H - NUMBER OF CHARACTERISTIC ATOMS OF IONIC SPECIES #1, 2, 3, etc. (I)

PROGRAM EXECUTION

Execution of the program begins with the last piece of data entered. The following message will appear on the screen:

MODIFY THE DATA ? [Y/N]:

Type N if the data was entered correctly or if you want to see the data before modification.

Typing Y will start the data modification routine. The computer will then display the following menu:

MODIFY WHICH VALUE ?

1 - TEMPERATURE

2 - FREE ENERGY OF H2O

3 - NON-METALLIC ION DATA

4 - METALLIC SPECIES DATA

5 - ALL DATA IS NOW CORRECT

ENTER OPTION:

For options 1 and 2, the old values are displayed along with a prompt for the desired value. For options 3 and 4, the computer prompts for the parameter to be modified before displaying the old value and awaiting input of the new value. Once a value has been modified, the "MODIFY WHICH VALUE ?" menu is again displayed. Option 5 will end the data modification routine.

The computer will next prompt with:

DISPLAY TABULATED DATA ? [Y/N]:

Type N if you do not want to see the data displayed in tabular form.

Typing Y will start the data display routine. The computer prompts with:

DATA DISPLAY OPTIONS

P - PRINTER

T - TERMINAL

ENTER OPTION:

The data is then displayed as specified.

APPLE NOTE: To temporarily halt execution of the program type (CTRL) S ; that is type S while depressing the CTRL key. This freezes the screen display until (CTRL) S is typed again.

The computer next prompts with:

EQUATION LISTING OPTIONS

P - PRINTER

T - TERMINAL

ENTER OPTION:

The equations will be listed as specified.

The computer displays the message "EXECUTION COMPLETE...." to signal the end of the run and prompts with:

EXECUTE AGAIN ? [Y/N]:

Type N to quit.

Type Y and the computer asks:

REPEAT EXECUTION WITH SAME DATA ? [Y/N]:

Type N and the computer asks:

MODIFY CURRENT DATA SET ? [Y/N]:

Type Y to initiate the data modification routine.

Type N and the system reinitializes and prompts for new inputs.

APPENDIX 1

REASONS FOR "IMPROPER DATA" MESSAGE

1 - Entering an alphabetic character for a numeric variable.

Retype the number correctly.

2 - A decimal point (.) for an integer variable.

Retype the number omitting the decimal point.

3 - A plus sign (+) for electric charge.

Retype the number omitting the plus sign.

Note: Negative values do require the minus sign.

PROGRAMMING NOTES

The interactive version of the PH - POTENTIAL PROGRAM is largely a translation of the WATFIV version that makes use of UCSD - APPLE PASCAL features. Despite the limitations imposed on the size of procedures and the limitations imposed by the stack, the program is designed in modules to facilitate modification and/or extensions. The program is intended to be implementable on any microcomputer that supports the UCSD operating system.

The current version makes extensive use of Intrinsic Units, separately compiled program segments that have been installed into the SYSTEM.LIBRARY. These Units are swapped into active memory only when a call is made to a procedure found in the Unit. A brief description of each of the Units follows.

GETVAL

This Unit converts an integer, C[j], into a corresponding character array, SAVE. For example, a C[j] value of 123 is converted to the array of values SAVE[1]='1', SAVE[2]='2', SAVE[3]='3'.

CONVERT

This Unit converts character arrays to integer or real values through the procedures CHARTOINT and CHARTOREAL.

INDATA

This Unit contains the data entry and data modification routines. All inputs are read as characters to prevent crashing by variable type mismatches. Integer and real values are extracted by the CONVERT Unit.

STABLE

This Unit calculates the areas of predominance for the species being considered.

****
* TEMPERATURE = 298.15 *
****

ION NUMBER	IONIC SPECIES	FREE ENERGY OF FORMATION	IONIC ACTIVITY	CHARGE
1	CL-	-31372.0	0.100000	-1
2	HS-	2880.0	0.100000	-1

ION NUMBER	CHARACTERISTIC ATOMS	OXYGEN ATOMS	HYDROGEN ATOMS
1	1	0	0
2	1	0	1

METALLIC SPECIES	FREE ENERGY OF FORMATION	SPECIES ACTIVITY	CHARGE
FE	0.0	1.00000	0
FE3O4	-242700.	1.00000	0
FE2O3	-177400.	1.00000	0
HFEO2-	-90300.0	0.000001	-1
FEOH++	-54830.0	0.000001	2
FE(OH)2+	-104700.	0.000001	1
FECL2(AQ)	-81590.0	0.000001	0
FECL++	-34400.0	0.000001	2
FES(SOL)	-24000.0	1.00000	0
FES2(SOL)	-39900.0	1.00000	0
	-196820.	0.000001	0

METALLIC SPECIES	METAL ATOMS	OXYGEN ATOMS	HYDROGEN ATOMS
FE	1	0	0
FE3O4	3	4	0
FE2O3	2	3	0
HFEO2-	1	2	1
FEOH++	1	1	1
FE(OH)2+	1	2	2
FECL2(AQ)	1	0	0
FECL++	1	0	0
FES(SOL)	1	0	0
FES2(SOL)	1	0	0
	1	4	0

METALLIC SPECIES	NUMBER OF CHARACTERISTIC ATOMS OF EACH IONIC SPECIES				
	ION1	ION2	ION3	ION4	ION5
FE	0	0			
FE3O4	0	0			
FE2O3	0	0			
HFEO2-	0	0			
FEOH++	0	0			
FE(OH)2+	0	0			
FECL2(AQ)	2	0			
FECL++	1	0			
FES(SOL)	0	1			
FES2(SOL)	0	2			
	0	1			

NUMBER	EQUILIBRIUM	BALANCED CHEMICAL EQUATION
1	E= -0.0864 - 0.0592PH	3*FE + 4*H2O = FE3O4 + 8*H+ + 8*E-
2	E= -0.0530 - 0.0592PH	2*FE + 3*H2O = FE2O3 + 6*H+ + 6*E-
3	E= 0.3229 - 0.0887PH	FE + 2*H2O = HFEO2- + 3*H+ + 2*E-
4	E= -0.0914 - 0.0197PH	FE + H2O = FEOH++ + H+ + 3*E-
5	E= 0.0071 - 0.0394PH	FE + 2*H2O = FE(OH)2+ + 2*H+ + 3*E-
6	E= -0.5270	FE + 2*CL- = FECL2(AQ) + 2*E-
7	E= -0.1424	FE + CL- = FECL++ + 3*E-
8	E= -0.5532 - 0.0296PH	FE + HS- = FES(SOL) + H+ + 2*E-
9	E= -0.4654 - 0.0296PH	FE + 2*HS- = FES2(SOL) + 2*H+ + 4*E-
10	E= 0.0878 - 0.0532PH	FE + 4*H2O + HS- = + 9*H+ + 10*E-
11	E= 0.2144 - 0.0592PH	2*FE3O4 + H2O = 3*FE2O3 + 2*H+ + 2*E-
12	E= -1.3144 + 0.0296PH	3*HFEO2- - 2*H2O = FE3O4 - H+ + 2*E-
13	E= -0.1318 + 0.2958PH	FE3O4 - H2O = 3*FEOH++ - 5*H+ + E-
14	E= 0.7555 + 0.1183PH	FE3O4 + 2*H2O = 3*FE(OH)2+ - 2*H+ + E-
15	E= 1.2353 - 0.2367PH	3*FECL2(AQ) + 4*H2O - 6*CL- = FE3O4 + 8*H+ + 2*E-
16	E= -0.5902 + 0.4733PH	FE3O4 - 4*H2O + 3*CL- = 3*FECL++ - 8*H+ + E-
17	E= 1.3141 - 0.1479PH	3*FES(SOL) + 4*H2O - 3*HS- = FE3O4 + 5*H+ + 2*E-
18	E= -1.2235 + 0.0296PH	FE3O4 - 4*H2O + 6*HS- = 3*FES2(SOL) - 2*H+ + 4*E-
19	E= 0.1511 - 0.0511PH	FE3O4 + 8*H2O + 3*HS- = 3* + 19*H+ + 22*E-
20	E= -0.8048	2*HFEO2- - H2O = FE2O3 + 2*E-
21	PH= 0.98	2*FEOH++ + H2O = FE2O3 + 4*H+
22	PH= -3.05	2*FE(OH)2+ - H2O = FE2O3 + 2*H+

23	E= 0.8950 - 0.1775PH	2*FECL2(AQ) + 3*H2O - 4*CL- = FE2O3 + 6*H+ + 2*E-
24	PH= 1.51	2*FECL++ + 3*H2O - 2*CL- = FE2O3 + 6*H+
25	E= 0.9476 - 0.1183PH	2*FES(SOL) + 3*H2O - 2*HS- = FE2O3 + 4*H+ + 2*E-
26	E= -1.7028 + 0.0592PH	FE2O3 - 3*H2O + 4*HS- = 2*FES2(SOL) - 2*H+ + 2*E-
27	E= 0.1481 - 0.0507PH	FE2O3 + 5*H2O + 2*HS- = 2* + 12*H+ + 14*E-
28	E= -0.9202 + 0.1183PH	HFEO2- - H2O = FEOH++ - 2*H+ + E-
29	E= -0.6245 + 0.0592PH	HFEO2- = FE(OH)2+ - H+ + E-
30	PH= 9.58	FECL2(AQ) + 2*H2O - 2*CL- = HFEO2- + 3*H+
31	E= -1.0730 + 0.1775PH	HFEO2- - 2*H2O + CL- = FECL++ - 3*H+ + E-
32	PH= 14.81	FES(SOL) + 2*H2O - HS- = HFEO2- + 2*H+
33	E= -1.2538 + 0.0296PH	HFEO2- - 2*H2O + 2*HS- = FES2(SOL) - H+ + 2*E-
34	E= 0.0290 - 0.0444PH	HFEO2- + 2*H2O + HS- = + 6*H+ + 8*E-
35	PH= 5.00	FEOH++ + H2O = FE(OH)2+ + H+
36	E= 0.7796 - 0.0592PH	FECL2(AQ) + H2O - 2*CL- = FEOH++ + H+ + E-
37	PH= 2.58	FECL++ + H2O - CL- = FEOH++ + H+
38	E= 0.8322	FES(SOL) + H2O - HS- = FEOH++ + E-
39	E= -1.5874 - 0.0592PH	FEOH++ - H2O + 2*HS- = FES2(SOL) + H+ + E-
40	E= 0.1646 - 0.0676PH	FEOH++ + 3*H2O + HS- = + 8*H+ + 7*E-
41	E= 1.0753 - 0.1183PH	FECL2(AQ) + 2*H2O - 2*CL- = FE(OH)2+ + 2*H+ + E-
42	PH= 3.79	FECL++ + 2*H2O - CL- = FE(OH)2+ + 2*H+
43	E= 1.1279 - 0.0592PH	FES(SOL) + 2*H2O - HS- = FE(OH)2+ + H+ + E-
44	E= -1.8831	FE(OH)2+ - 2*H2O + 2*HS- = FES2(SOL) + E-
45	E= 0.1223 - 0.0592PH	FE(OH)2+ + 2*H2O + HS- = + 7*H+ + 7*E-
46	E= 0.6268	FECL2(AQ) - CL- = FECL++ + E-

47	PH= -0 89	FECL2(AQ) - 2*CL- + HS- = FES(SOL) + H+
48	E= -0 4039 - 0 0592PH	FECL2(AQ) - 2*CL- + 2*HS- = FES2(SOL) + 2*H+ + 2*E-
49	E= 0 2414 - 0 0666PH	FECL2(AQ) + 4*H2O - 2*CL- + HS- = + 9*H+ + 8*E-
50	E= 0.6794 + 0.0592PH	FES(SOL) + CL- - HS- = FECL++ - H+ + E-
51	E= -1.4346 - 0.1183PH	FECL++ - CL- + 2*HS- = FES2(SOL) + 2*H+ + E-
52	E= 0.1864 - 0.0761PH	FECL++ + 4*H2O - CL- + HS- = + 9*H+ + 7*E-
53	E= -0.3776 - 0.0296PH	FES(SOL) + HS- = FES2(SOL) + H+ + 2*E-
54	E= 0.2480 - 0.0592PH	FES(SOL) + 4*H2O = + 8*H+ + 8*E-
55	E= 0.4566 - 0.0690PH	FES2(SOL) + 4*H2O - HS- = + 7*H+ + 6*E-

****
* AREA OF PREDOMINANCE FOR: FE *
****

NO LOWER LIMIT

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.527	-2.00	-0.527	-0.89	6
-0.527	-0.89	-0.991	14.81	8
-0.991	14.81	-1.097	16.00	3

MINIMUM PH = -2.00
MAXIMUM PH = 16.00

****
* AREA OF PREDOMINANCE FOR: FE3O4 *
****

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.801	14.30	-0.750	16.00	18
-0.801	14.30	-0.876	14.81	17
-0.876	14.81	-0.841	16.00	12
-0.750	16.00	-0.841	16.00	

****
* AREA OF PREDOMINANCE FOR: FE2O3 *
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: HFEO2- *
****

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.876	14.81	-0.841	16.00	17
-0.991	14.81	-1.097	16.00	3
-0.876	14.81	-0.991	14.81	32
-0.841	16.00	-1.097	16.00	

****
* AREA OF PREDOMINANCE FOR: FEOH++ *
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: FE(OH)2+ *
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: FECL2(AQ) *
****

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.286	-2.00	-0.351	-0.89	48
-0.527	-2.00	-0.527	-0.89	6
-0.286	-2.00	-0.527	-2.00	
-0.351	-0.89	-0.527	-0.89	47

****
* AREA OF PREDOMINANCE FOR: FECL++ *
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: FES(SOL) *
****

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.351	-0.89	-0.801	14.30	53
-0.801	14.30	-0.876	14.81	17
-0.527	-0.89	-0.991	14.81	8
-0.351	-0.89	-0.527	-0.89	47
-0.876	14.81	-0.991	14.81	32

****
* AREA OF PREDOMINANCE FOR: FES2(SOL) *
****

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
0.595	-2.00	-0.648	16.00	55
-0.286	-2.00	-0.351	-0.89	48
-0.351	-0.89	-0.801	14.30	53
-0.801	14.30	-0.750	16.00	18
0.595	-2.00	-0.285	-2.00	
-0.648	16.00	-0.750	16.00	

****
* AREA OF PREDOMINANCE FOR: *
****

NO UPPER LIMIT

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
0.595	-2.00	-0.648	16.00	55

MINIMUM PH = -2.00
MAXIMUM PH = 16.00

****
* AREA OF PREDOMINANCE FOR: HFEO2- * DISSOLVED SPECIES DIAGRAM
****

NO LOWER LIMIT

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
-0.396	9.58	-0.681	16.00	34

MINIMUM PH = 9.58 LINE NO. 30
MAXIMUM PH = 16.00

****
* AREA OF PREDOMINANCE FOR: FEOH++ * DISSOLVED SPECIES DIAGRAM
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: FE(OH)2+ * DISSOLVED SPECIES DIAGRAM
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: FECL2(AQ) * DISSOLVED SPECIES DIAGRAM
****

NO LOWER LIMIT

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
0.375	-2.00	-0.396	9.58	49

MINIMUM PH = -2.00
MAXIMUM PH = 9.58 LINE NO. 30

****
* AREA OF PREDOMINANCE FOR: FECL++ * DISSOLVED SPECIES DIAGRAM
****

NO AREA OF PREDOMINANCE

****
* AREA OF PREDOMINANCE FOR: * DISSOLVED SPECIES DIAGRAM
****

NO UPPER LIMIT

POTENTIAL 1	PH 1	POTENTIAL 2	PH 2	LINE NO.
0.375	-2.00	-0.396	9.58	49
-0.396	9.58	-0.681	16.00	34

MINIMUM PH = -2.00
MAXIMUM PH = 16.00


DATE
ILME