Journal of METAMORPHIC GEOLOGY

CONSONORM_HG: a new method of norm calculation for mid- to high-grade metamorphic rocks

Journal:	*Journal of Metamorphic Geology*
Manuscript ID:	JMG-15-0009.R4
Manuscript Type:	Original Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Mathieu, Lucie; CONSOREM, Consortium de Recherche en Exploration Minérale - mineral exploration research consortium Trépanier, Sylvain; Mines Virginia, 300 Saint-Paul, suite 200 Daigneault, Réal; Centre d'études sur les Ressources minérales (CERM), Université du Québec à Chicoutimi, 555 Boul. de l'Université
Keywords:	CONSONORM_HG norm, normative minerals, classification, exploration, alteration indices

CONSONORM_HG: a new method of norm calculation for mid- to high-grade metamorphic rocks

L. MATHIEU, [1*] S. TRÉPANIER[2] AND R. DAIGNEAULT[3]

[1] *CONSOREM (Consortium de Recherche en Exploration Minérale - mineral exploration research consortium), 555 Boul. de l'Université, Chicoutimi, Canada, G7H 2B1.*

[2] *Mines Virginia – 300 Saint-Paul, suite 200, Québec, Canada, G1K 7R1*

[3] *Centre d'études sur les Ressources minérales (CERM) - Université du Québec à Chicoutimi, 555 Boul. de l'Université, Chicoutimi, Canada, G7H 2B1.*

*Corresponding author: mathiel@tcd.ie, 1-418-545-5011 ext. 2538

Short title: CONSONORM_HG norm for metamorphic rocks

ABSTRACT

The CONSONORM_HG norm provides a standardised solution to approximate metamorphic assemblages for mid- to high-grade metamorphic rocks. Other applications for this norm include: 1) the standardised classification of metamorphic rocks; 2) the interpretation of hydrothermal alteration using indices calculated from normative minerals; 3) the estimation of the carbonate content of rocks, even when CO_2 has not been analysed. CONSONORM_HG is designed for rocks dominated by silicates, Fe–Ti oxides or carbonates, and it approximates the main mineral assemblages of amphibolite, granulite, greenschist and blueschist facies metamorphic rocks. The norm calculates silicate assemblages using one of three Al–Ca–NaK–FeMg tetrahedra, a convenient way of representing a large number of silicate assemblages, for each of the 17 *P–T* conditions modelled and based on the Fe–Mg–Mn composition of the sample. In addition to silicate minerals, CONSONORM_HG calculates Fe–Ti oxides and other accessory minerals from minor elements, sulphides from analysed sulphur and/or from analysed metals and also carbonates from analysed CO_2 or from normative CO_2 estimated from the LOI. CONSONORM_HG also integrates many reactions to address silicon deficits, as well as quartz–carbonate reactions to approximate natural assemblages better. The normative calculation is validated using published whole rock analyses and petrographic descriptions and its various applications are discussed.

KEY WORDS

CONSONORM_HG norm, normative minerals, classification, exploration, alteration indices

INTRODUCTION

Normative minerals have been commonly used to study magmatic rocks since the publication of the CIPW norm over a century ago (Cross *et al.*, 1902, 1912). In addition to the CIPW, different normative calculations are available for low to mid-grade metamorphic rocks (Barth, 1959; Piché & Jébrak, 2004) and for sedimentary rocks (e.g. Cohen & Ward, 1991 and references included in this publication).

The CIPW norm is a quantitative system initially developed to classify igneous rocks (Cross *et al.*, 1912). This standardised classification is used to name rocks, to group samples into the units displayed on maps and to facilitate the comparison of lithological units from different regions. Other norms, such as NORMAT (Piché & Jébrak, 2004), were used to develop alteration indices for VMS (Volcanogenic Massive Sulphides) type deposits. Normative calculations are thus important tools that enable their users to relate chemical analyses to petrographic observations, to classify rocks and to interpret samples in terms of magmatic suites or hydrothermal alteration, for example.

For geologists working with igneous and/or sedimentary rocks, normative solutions are numerous and are constantly being improved. For metamorphic rocks however, the solutions are more limited and absent for high-grade rocks. A possible explanation for this deficiency is that the great variety of metamorphic minerals and rock types observed in nature is difficult to model. Also, metamorphic assemblages are best described by pseudosections established from and for a limited amount of samples. A standardized tool that could be applied to any type of metamorphic rocks from a large variety of *P–T* conditions remains to be developed.

Pioneering researchers, such as Barth (1959), demonstrated the possibility of developing norms for metamorphic rocks. Building on these earlier attempts, a new calculation for mid- to high-grade metamorphic assemblages is proposed here. This method is named the CONSONORM_HG, with HG standing for “high grade”, and is available as a Visual Basic code in the supplemental data to this publication (see File 1).

CONSONORM_HG is designed to rapidly approximate metamorphic mineral assemblages for users unfamiliar with more refined thermodynamic tools, who do not have the time to perform such calculations on large datasets or who need to approximate the mineral proportions of samples containing complex assemblages of silicates, carbonates, oxides and sulphides. CONSONORM_HG also standardises metamorphic assemblages and can thus be used, for example, to calculate alteration indices or to develop classification charts for metamorphic rocks. It should be noted that thermodynamic software also provide constraints on mineral assemblages and that CONSONORM_HG is not an alternative to such approach but, instead, it serves specific purposes detailed hereafter.

CONSONORM_HG was initially developed for mineral exploration purposes; i.e. to calculate mineral proportions ratio (alteration indices). For this reason, efforts were made to include many carbonates, oxides and sulphides in the calculation sequence, and to approximate the CO_2 and H_2O content of rocks following the NORMAT method (Piché & Jébrak, 2004). CONSONORM_HG can thus be viewed as an extension of NORMAT to high grade conditions, even if these norms employ different calculation strategies.

Various applications for the norm, including the calculation of alteration indices, are discussed in the last section of this contribution, after presenting the calculation sequence of CONSONORM_HG.

ALTERATION INDICES FROM NORMATIVE MINERALS

As demonstrated by CIPW and NORMAT (Cross *et al.*, 1912; Piché & Jébrak, 2004), norms may serve various purposes; the calculation of alteration indices being one of them. In this section, alteration indices are presented and a method is recommended for their calculation.

Alteration indices

Alteration indices are dimensionless numbers used to qualify and quantify the following types of hydrothermal alterations: C, H, K, Na, Ca, Fe, Mg and Si (see carbonatation, hydratation, potassic alteration, sericitisation, albitisation, propylitic alteration, epidotisation, hematisation, chloritisation, silicification etc.).

Most alteration indices are elemental ratios. For example, the Sericite-Albite saturation index is defined as follows: $(3 * K_2O + Na_2O)/Al_2O_3$ (Kishida & Kerrich, 1987). Usually, alteration indices are designed to be constant, preferentially close to 0, for fresh rocks, and to be >0 for altered rocks. However, an elemental ratio such as the Sericite-Albite saturation index is not constant in every fresh rock and it is difficult to discriminate between sericitisation, potassic alteration and albitisation with such an index.

As demonstrated by NORMAT, these difficulties can be minimised by alteration indices calculated using mineral proportions (Piché & Jébrak, 2004). For example, sericitisation can be quantified using the following ratio: (muscovite + paragonite) / (sum of silicates). The calculation of such indices requires mineral proportions data, which can be obtained from hand samples and thin sections measurements, or that can be estimated from thermodynamic calculations (THERMOCALC, Perple_X, Theriak, etc.; Holland & Powell, 1998; Connolly & Petrini, 2002; Connolly, 2005; De Capitani & Petrakakis, 2010) or from normative calculations (Piché & Jébrak, 2004; Trépanier *et al.*, in press).

Methods for mineral proportion estimation

Ideally, mineral proportions should be measured from thin sections, but this method is expensive, time consuming and unrealistic for large datasets, especially in a mining exploration context. Mineral proportions should thus either be obtained from Gibbs free energy minimisation software or from normative calculations.

Thermodynamic software may easily calculate mineral assemblages that will be accurate for most samples, but the software will tend to perform these calculations for specified chemical systems that may exclude elements and minerals present in rocks. For the calculation of alteration indices, this characteristic of thermodynamic software represents a limitation that is illustrated using a “sericite schist” sample from the ore zone of the Hemlo gold deposit (Cameron & Hattori, 1985). Normative minerals calculated using the CONSONORM_HG norm (see below for a description of the calculation sequence; Table 1, 2, 3) correspond to the minerals described in the field; i.e. feldspar, quartz, white mica, barite, pyrite and minor carbonates (see Cameron & Hattori, 1985; Muir 2002). The assemblage calculated using the Theriak-Domino software (De Capitani & Petrakakis, 2010) however, lacks carbonates and sulphides, and calculates undescribed garnet and a large amount of biotite (Table 1). Theriak-Domino is less performant in this case mostly because the thermodynamic database used (JUN92) does not include sulphides. For such rocks, one could use a database that documents sulphides (e.g. the supcrt92 database distributed with Perple_X; Johnson *et al.*, 1992; Connolly & Petrini, 2002). However, results obtained from Perple_X, Theriak-Domino and other software will greatly vary depending on the parameters and database selected by various users, while norms offer a limited amount of options to their users. For this reason, the thermodynamic approach is the most precise and should be favoured in most contexts

(e.g. to provide constraints on *P–T* conditions, etc.), but standardised tools such as alteration indices should preferentially be calculated using norms resembling this integrated to the NORMAT method.

NORMAT is designed for greenschist facies rocks. With a growing interest of exploration for Provinces metamorphosed at high grade, alteration indices for amphibolite to granulite facies rocks need to be developed. One of the advantages of calculating high grade mineral assemblages is that normative minerals can be compared to minerals observed in the field, helping geologists to relate their field observations to calculations preformed on chemical analyses. Also, calculating high grade normative assemblages limits approximations and errors that could arise from the calculation of low grade parageneses from high grade rocks; i.e. the calculation of chlorite from cordierite-bearing rock might hide metamorphism-induced modifications and induce misleading quantification of the chloritisation event. The main purpose of the CONSONORM_HG norm it to provide an easy way to calculate normative minerals and related alteration indices from large datasets of samples that may contain large amounts of non-silicate minerals and for which CO_2 and H_2O are not always analysed; i.e. databases typically used by mining exploration.

NORMATIVE CALCULATIONS

Norms calculate theoretical mineralogical compositions of rocks (Foucault & Raoult, 1992) and could be viewed as modeled and standardized assemblages of equilibrated minerals. For igneous rocks, the calculation targets assemblages of minerals that have generally crystallised within a limited period of time from magmas with similar compositions and temperatures.

For metamorphic rocks, norms calculate minerals formed in similar conditions of pressure (P), temperature (T) and bulk composition (equilibrium assemblage). In most metamorphic rocks, the assemblage of minerals that makes up most of the volume of a rock forms at near peak metamorphic T condition, shortly after the peak metamorphic P (for clockwise *P–T* paths). For some rocks, however, most of their volume is made up by a retrograde assemblage. A normative calculation designed for metamorphic rocks approximates the equilibrated minerals that make up most of the volume of a rock, and ignores minor phases, i.e. relic and retrograde phases in most cases.

Norms for igneous rocks

For igneous rocks, the CIPW is the oldest, best known and most commonly used tool available (Cross *et al.*, 1902, 1912). Numerous programmes and modifications have been published over the years (Kelsey 1965; Le Maitre 1976; Glazner 1984; Fears 1985; Verma *et al.*, 2003; Pruseth 2009). Some of the CIPW calculation principles that inspired the authors as they developed the CONSONORM_HG norm are described below.

Sequential calculation – The CIPW norm is a sequential calculation, in which minerals are calculated one after the other. This sequence does not correspond to the crystallisation sequence of natural rocks, thus avoiding the introduction of hypotheses that could be hazardous. The calculation starts with the formation of accessory phases, i.e. phases such as apatite that consume all the analysed phosphorus. The main silicates are then calculated by assuming that the rock contains an excess of silicon, and the calculation ends by solving silicon deficits if necessary.

Simultaneous calculation – A notable variant to the sequential calculation of the CIPW norm is MATNORM (Pruseth, 2009), which estimates the proportion of several minerals simultaneously by solving a set of equations represented as matrices. This innovation estimates mineral proportions more accurately. The maximum amount of mineral proportions that can be approximated in such a way is equal to the amount of chemical variables considered; thus, with four variables such as Al, Ca, K+Na and Fe+Mg+Mn for example, we can simultaneously estimate the proportions of four minerals (Pruseth, 2009).

Simplified formulas – Minerals usually have complex formulas, are solid solutions of various end-members and may incorporate various trace elements into their structures. The aim of a norm is to standardise the composition, not to embrace the full complexity of nature. For this reason, the calculation uses simplified formulas of end-members. Feldspar is an example of an easily modelled mineral; except that some authors choose to introduce Ba in its structure (see Pruseth, 2009 for example) and others do not. Other minerals, such as the members of the amphibole group, have complex compositions and are consequently harder to model (Barth, 1959) (see also the "Natural examples" section for more details on the difficulty to properly model amphiboles).

Norms for metamorphic rocks

The modelling of metamorphic assemblages is complex, as the mineral assemblages vary according to *P–T* conditions, the composition of the metamorphic fluid and the chemical composition of the rock. The problem posed by the compositional dependency of the assemblages was originally addressed by MESONORM, which introduced the possibility of a normative calculation for amphibolite grade rocks (Barth, 1959).

Tetrahedra – The MESONORM norm uses a K–Al–Ca–Mg tetrahedron to display several of the main assemblages most often observed in metamorphic rocks of the mesozone (amphibolite grade). Depending on its composition in K–Al–Ca–Mg, which can be regarded as its location in the tetrahedron, a rock will develop one of the mineral assemblages displayed on Barth's diagram (Barth, 1959). The main limitation of this norm is that tetrahedra were difficult to manipulate in 1959, which is no longer an obstacle with modern programming techniques.

Ternary diagrams – Instead of a tetrahedron, the NORMAT solution uses an Al–Ca–FeMg ternary diagram to display four greenschist assemblages (Piché & Jébrak, 2004). This choice was justified by the limited number of assemblages considered.

Virtual CO_2 estimation – The NORMAT norm addresses the problem of carbonate calculation for rocks with no analysed CO_2 and H_2O values by estimating normative CO_2 and H_2O compositions from the LOI (Lost On Ignition) (Piché & Jébrak, 2004). The estimate is made by calculating a carbonate-free mineral assemblage and by comparing the amount of structural H_2O (H_2O^+) consumed by the normative minerals against an analytical LOI from which analysed S, interstitial H_2O (H_2O^-) and calculated GOI (Gain On Ignition) have been subtracted. If the normative minerals are unable to consume all the volatiles of the analysed LOI, then the LOI is assumed to contain a small amount of CO_2, which is used to re-calculate a carbonate-bearing assemblage. The amount of normative CO_2 is progressively increased, through successive iterations, until the LOI can be separated into proper amounts of normative H_2O^+, S, GOI, H_2O^- and CO_2.

CONSONORM_HG TETRAHEDRA

Many tools used by geoscientists are based on the general notion of "component mapping" (see Spear, 1994; Torres-Roldan *et al*., 2000). CONSONORM_HG is no exception, as its mineral assemblages are represented and read from four components systems; i.e. tetrahedra. Compared to ternary diagrams, tetrahedra have a fourth apex that allows for a more accurate representation of a larger number of mineral assemblages. Also, tetrahedra are used because, as they represent only four components, they are easier to design and manipulate than >4 components systems. The CONSONORM_HG norm uses ACKNFM tetrahedra to fit a main silicate assemblage to each sample. These tetrahedra have the following apexes: Al (A), Ca (C), K+Na (KN) and Fe+Mg+Mn (FM). Also, note that the ACKNFM tetrahedra and accompanying AFMM tetrahedra (Al–Fe–Mg–Mn; see next section) can be view as a combination of the ACF and AFM ternary diagrams.

Building tetrahedra – Each ACKNFM tetrahedron is an assemblage of four ternary diagrams (ACKN, ACFM, AFMKN and CFMKN ternary diagrams), which represent mineral assemblages for H_2O and SiO_2 saturated rocks, and for a single *P–T* condition. Once assembled, the four ternary diagrams form the *main tetrahedron* that contains several *small tetrahedra*, each defined by an assemblage of four minerals (Fig. 1).

Thermodynamic data – To ensure the coherence of the norm calculation, every ternary diagram is produced following a similar routine. Collecting these diagrams from the published literature is thus not an option. Instead, the authors used the Perple_X software (Connolly & Petrini, 2002; Connolly, 2005), which uses thermodynamic databases and solid-solution models (see references from the Perple_X website), solves thermodynamic equations and may display the resulting equilibrium assemblages as ternary diagrams. The CONSONORM_HG norm was built using thermodynamic data from the hp02ver database, established by Holland & Powell (1998) and modified by the authors of Perple_X (Connolly & Petrini, 2002; Connolly, 2005) and using several solid-solution models (see File 2 provided as supporting information for details).

Using the tetrahedra – To use the tetrahedra, the code translates the chemical composition of the rock in K, Na, Ca, Al, Fe, Mg and Mn in barycentric coordinates to display the analysed sample as a point in the tetrahedral space (i.e. in the *main tetrahedron*). Then, the point is re-projected using each *small tetrahedron* as a reference. If the point falls outside a given *small tetrahedron*, at least one of its barycentric coordinates is negative; otherwise, they are all positive, which enables the code to select the *small tetrahedron* within which the point is located (Fig. 1). The main silicate assemblage is then calculated from the four minerals that constitute the apexes of the selected *small tetrahedron.*

CALCULATION SEQUENCE OF CONSONORM_HG

As the mineral assemblages for metamorphic rocks vary according to P and T, each normative calculation is performed for a specific *P–T* condition. The CONSONORM_HG norm may perform calculations for 17 different *P–T* conditions, or models (Fig. 2). These 17 models are located on either side of the main reactions involving silicates common in most rocks, as well as being regularly distributed in the *P–T* space. Once the user selects a model and the appropriate options (see next section), the normative calculation starts, following a sequence summarised below. Note that the

tetrahedra and other diagrams used by the code and referred to in this section are provided as additional material.

The first operations performed by CONSONORM_HG aim to extract and prepare the chemical data for the norm calculation, which is carried out as follows (step 1 of Fig. 3):

1) The chemical data are extracted from the input file. The calculation uses chemical data on the main oxides (SiO_2, Al_2O_3, CaO, MgO, FeO, Fe_2O_3, MnO, Na_2O, K_2O, TiO_2, P_2O_5), volatiles (H_2O^+, H_2O^-, S, CO_2, Cl, F) and some trace elements (Zr, Ba, B, Cr, Pb, Zn, Ni, Mo, Cu, As), which are extracted, re-calculated to 100%, and converted to moles.

2) Operations are then performed on the volatiles. If the user decides to use the analysed value of CO_2 (see next section), then the normative calculation starts with the formation of accessory minerals. Otherwise, H_2O and CO_2 values are estimated normatively from the LOI, using the method set out by Piché & Jébrak (2004). To perform this normative estimate, the code decomposes the LOI into its different constituents (equation 1). In this equation, the value of CO_2_normative% is initially zero and that of H_2O^+_normative is maximal; these values will be adjusted through successive iterations after silicate calculation. Also, one of the LOI constituents is the GOI (Gain On Ignition), i.e. the oxidation of the iron contained in the sulphides and carbonates during the heating of the sample, which is calculated from the amount of iron contained in the normative carbonates and sulphides. The value of the GOI is adjusted as these minerals are calculated (see equation 2).

$$LOI = S\_analysed + H_2O^-\_analysed + H_2O^+\_normative + CO_2\_normative - GOI \quad (1)$$

$$GOI\ (wt\%) = (sulphide\% * Fe\_molar\_in\_sulphide * 1.5 + carbonate\% * Fe\_molar\_in\_carbonate * 0.5) * 15.998 \quad (2)$$

The normative calculation starts by forming accessory minerals, sulphides, carbonates and Fe–Ti oxides as follows (see step 2 of Fig. 3):

1) Elements that can only be accommodated by a specific type of mineral are distributed between various accessory minerals. The following elements are consumed at this stage: P, Cl, F (apatite), Cr (chromite), B (one among four types of OH tourmalines), Zr (zircon), Ba (barite), Pb, Zn, Ni, Mo, Cu and As (galena, sphalerite, millerite, molybdenite, chalcopyrite, arsenopyrite). Note that sulphides will be formed, even if sulphur has not been analysed.

2) Additionally, pyrite and pyrrhotite might be formed under certain circumstances, only if sulphur has been analysed. If sulphur is not fully consumed by the minerals listed above, then pyrite or pyrrhotite, depending on the model used, are formed until sulphur is exhausted (Fig. 2).

3) At this stage, the normative or analysed CO_2, depending on the option selected by the user (see next section), is consumed to form carbonates. In nature, CO_2-poor rocks tend to contain only calcite, while rocks richer in CO_2 may contain Fe- and Mg-carbonates. To translate this observation numerically, the CONSONORM_HG method is designed as follows: 1) if CO_2 < CaO (molar), only calcite is formed; 2) if CO_2 > CaO (molar), then an assemblage of three carbonates is calculated using mineral assemblages represented on a $CaCO_3$–$FeCO_3$–$MgCO_3$ ternary diagram.

4) If the analysis contains Fe_2O_3 and/or TiO_2, an assemblage of three Ti–Fe oxides is calculated using a TiO_2–FeO–Fe_2O_3 ternary diagram. The amount of FeO allocated to the Fe–Ti oxides depends on the options selected by the user (see next section).

The Mg# (MgO/(MgO+MnO+FeO) molar) and K# (K_2O/(K_2O+Na_2O) molar) are then calculated using the amount of MgO, FeO, MnO, K_2O and Na_2O remaining in the sample after the calculation of the accessory phases.

The calculation of silicates is then started by selecting the four minerals of the main assemblage (see step 3 of Fig. 3). These minerals are selected from an ACKNFM tetrahedron, using the following procedure:

1) Because the mineralogy of Mg-rich rocks greatly contrasts with this of Fe-rich rocks, combining Fe, Mg and Mn on the FM apex of the ACKNFM tetrahedra causes problems. To address them, a total of three ACKNFM tetrahedra are available for each of the 17 models: tetrahedron I represents Mg-rich compositions, tetrahedron II represents Fe-rich and/or Mn-rich compositions and tetrahedron III represents intermediate Fe-Mg compositions. In general, tetrahedron II must be applied to garnet-bearing rocks that may contain staurolite, olivine, chloritoid or spinel, and tetrahedron I best applies to rocks within which cordierite, chlorite, carpholite, talc and/or anthophyllite are stable. The first step of silicate calculation consists of choosing the ACKNFM tetrahedron that best approximates the mineralogy of the sample considered (Fig. 4).

2) The ACKNFM tetrahedron I, II or III is selected by the intermediary of an AFMM tetrahedron, whose apexes correspond to Al, Fe, Mg and Mn (Fig. 4). The procedure used is the following: 1) the sample is represented in area I, II or III of the AFMM tetrahedron and, depending on the sample location in this tetrahedral space, the ACKNFM tetrahedron I, II or III is subsequently used; 2) using the appropriate ACKNFM tetrahedron, the sample is plotted in the main tetrahedral space to select the small tetrahedron within which the sample lies (Fig. 4). The small tetrahedron selected represents an assemblage of four minerals that corresponds to the main silicate assemblage of the sample.

3) It should be noted that, in detail, the stability fields of biotite, white mica (paragonite and muscovite), spinel, staurolite and actinolite are adjusted, within the ACKNFM tetrahedra I, II and III, depending on the Mg# and K#, enabling the tetrahedra to represent the stability field of these minerals more accurately. Also, the composition of the FM apex (olivine, or talc, etc.) is deduced using Fe–Mg–Mn ternary diagrams. It is the bulk of these data, i.e. the AFMM and ACKNFM tetrahedra, the Fe–Mg–Mn ternary diagram and the Mg# and K#, which are actually used to select the four minerals of the main silicate assemblage.

The silicates are then calculated as follows (see step 3 of Fig. 3):

1) The proportions of the four minerals selected from the ACKNFM tetrahedron are calculated simultaneously using the MATNORM method (Pruseth, 2009). A part of these minerals are solid solutions that are, at this stage, distributed between their end-members using the Mg# and K# (for example, olivine divided into fayalite and forsterite).

2) The amount of mica is then corrected to ensure that the rock contains enough K to form the amount of biotite previously calculated, and to distribute K and Na more realistically between white mica and alkali feldspars, or between biotite and glaucophane, if these phases co-exist.

3) At this stage, the assemblage may contain three types of amphiboles: anthophyllite, glaucophane and/or actinolite. However, the complexity of the amphibole group could not be fully represented on the ACKNFM tetrahedra, and the silicate assemblage is thus corrected to take these complexities into account. These corrections apply to the amphibolite facies only. They consist of reactions between feldspars, clinopyroxenes and Al–Fe–Mg-bearing minerals that aim to form maximum amounts of pargasite and tschermakite.

4) The CONSONORM_HG norm then estimates the amount of silicon consumed by the calculated phases. If the normative minerals have not fully consumed Si, then quartz is made; otherwise, the minerals are progressively consumed using several reactions until the silicon deficit is solved.

Quartz and carbonates are then reacted under certain circumstances (see step 4 of Fig. 3). Indeed, for 13 of the models, carbonates are not stable in the presence of a large amount of quartz. At this stage, a SiO_2–$CaCO_3$–$FeCO_3$–$MgCO_3$ tetrahedron is used to determine if quartz and carbonates may or may not co-exist (Fig. 5). If necessary, the carbonates are destroyed, CO_2 is used to form graphite and the amount of CaO, MgO, and FeO initially contained in the carbonates is transferred to the silicates, which are re-calculated.

At this stage, the amount of H_2O consumed by the normative hydrous minerals (H_2O^+_mineral) is estimated. If the user has chosen to work with the amount of H_2O and CO_2 analysed, then the following messages are generated:
(a) Select a higher temperature model if (H_2O_measured ± 0.1 wt%) < H_2O^+_mineral;
(b) Select a lower temperature model if (H_2O_measured ± 0.1 wt%) > H_2O^+_mineral;
(c) Or the code informs the user that he has chosen the appropriate model if (H_2O_measured ± 0.1 wt%) = H_2O^+_minerals.

Otherwise, if the user has chosen to estimate the amount of H_2O and CO_2 from the LOI (see step 5 of Fig. 3), CONSONORM_HG verifies the following conditions:

1) If LOI > (H_2O^+_mineral + CO_2_normative + H_2O^-_analysed + S_analysed – GOI) then:
(a) CO_2_normative is increased by 0.1 wt% and H_2O^+_normative is decreased accordingly.
(b) The bulk of minerals previously formed are destroyed.
(c) The code resumes the extraction of chemical elements and the normative calculation. From this step, the normative calculation iterates as many times as necessary for the correct amount of normative CO_2 and H_2O to be deduced from the LOI.

2) Otherwise the calculation ends, the amount of H_2O^+_normative is calculated (equation 3) and the following messages are provided:
(a) Select a higher temperature model if (H_2O^+_normative ± 0.1 wt%) < H_2O^+_mineral;
(b) Or the code informs the user that he has chosen the appropriate model if (H_2O^+_normative ± 0.1 wt%) = H_2O^+_mineral.

$$H_2O^+\text{\_normative} = \text{LOI} - \text{S\_analysed} - H_2O^-\text{\_analysed} - CO_2\text{\_normative} + \text{GOI} \quad (3)$$

The final operations performed by the CONSONORM_HG calculation are the following (see step 6 of Fig. 3):

1) The amount of Fe_2O_3 is adjusted between the silicates and the Fe–Ti oxides. Indeed, the CONSONORM_HG norm includes four silicates that incorporate Fe_2O_3 into

their structures (i.e. epidote, tschermakite, aegirine and riebeckite). During the silicate calculation only FeO was assigned to these silicates. At this stage, the amount of Fe_2O_3 contained in these silicates is estimated and, if necessary, Fe_2O_3 is taken from the Fe–Ti oxides to which additional FeO is assigned. The Fe–Ti oxides are then destroyed and their calculation is resumed using updated values of FeO and Fe_2O_3.

2) The final operations consist of estimating the weight percent of each normative mineral from their mole values and of estimating the density of the rocks using mineral' densities obtained from an internet compilation (i.e. "webmineral.com" by D. Barthelmy, 2014).

CALCULATION OPTIONS

The CONSONORM_HG calculation requires the user to set several options and parameters: 1) selecting one of the 17 *P–T* conditions modelled; 2) opting to use the amount of CO_2 (and H_2O) analysed or to estimate these values normatively from the LOI; and 3) setting parameters for Fe–Ti oxide calculation.

The option for Fe–Ti oxides is a user-defined number, which correspond to the value of the $Fe_2O_3/(Fe_2O_3+FeO)$ molar ratio of oxides. The value of this ratio is used to locate the sample in the TiO_2–FeO–Fe_2O_3 ternary diagram (see previous sections). It is particularly important to adjust this ratio for Fe–Ti oxide-rich samples, by using petrological observations. For example, if a rock is rich in hematite, the $Fe_2O_3/(Fe_2O_3+FeO)$ molar ratio should be set to a high value, whereas this ratio should be lower if the rock contains magnetite instead.

Also, when FeO and Fe_2O_3 have not been analysed (see Fritz & Popp, 1985 for a discussion of analytical options), they must be estimated prior performing the CONSONORM_HG calculation, as the norm does not propose options for estimating FeO and Fe_2O_3 from Fe_2O_3T (total iron). Users may easily add this option using the following methods. The FeO and Fe_2O_3 may be estimated using the LeMaître (1976) method, or using a user-defined Fe_2O_3/ Fe_2O_3T ratio deduced from statistics performed on magmatic rocks with fully analysed iron (see Middlemost, 1989). In the authors' experience, the Fe_2O_3/ Fe_2O_3T ratio is best adjusted using rocks of the studied area, or by using statistical values from a recent database (GEOROC database for example).

NATURAL EXAMPLES

Normative calculations in general and the CONSONORM_HG norm in particular have multiple geoscientific applications; of which a brief overview is proposed in this section. This overview is based on samples of metamorphic rocks from published petrological and geochemical studies.

Amphibolite and granulite facies meta-sedimentary and felsic to mafic meta-magmatic rocks were selected from around the World (see Table 2 and references included). The CONSONORM_HG norm was used to perform calculations on these samples using parameters summarised in Table 3.

Validation and use of CONSONORM_HG

Validating the norm – Various samples were used to validate the CONSONORM_HG method by comparing the proportions of observed and calculated minerals (Fig. 6). The results indicate a good correlation between observed and normative minerals with some

differences (especially for amphiboles) that are, for most, likely due to the structure of CONSONORM_HG; i.e. it is a model that uses simplified mineral formula and that is based on thermodynamic models unable to reproduce reality perfectly. Despite these errors, the correlation between real and modelled phases is good (Fig. 6), sufficient to name rocks on the basis of their normative minerals and to perform the various operations described below.

Naming and classifying metamorphic rocks – The CIPW norm (Cross *et al.*, 1902, 1912) was and is still used primarily in assisting the classification of magmatic rocks (see Streckeisen, 1976). A similar use can be envisaged for the CONSONORM_HG norm.

For example, we represent normative minerals calculated from amphibolite samples (Barros Gomes *et al.*, 1964) on an amphibole–plagioclase–other minerals ternary diagram (Fig. 7a). According to this diagram, twenty samples fall in the amphibolite field of the SCMR definition (see IUGS Subcommission on the Systematics of Metamorphic Rocks; Coutinho *et al.*, 2007) and are amphibolite rocks by definition, while another four samples are not.

We also used samples of metamorphic rocks (Table 2) to perform various normative calculations (see table 3) and to represent the results in the carbonate–calcsilicate–silicate ternary diagram (see Rosen *et al.*, 2007). Note that the marble samples from Davis & Ferry (1993) are actually pure marbles, impure marbles and carbonate-silicate rocks according to this classification (Rosen *et al.*, 2007). The CONSONORM_HG norm thus facilitates the use of such diagrams, and could become the corner stone of new systematics for the classification of metamorphic rocks.

Calculating carbonates – CO_2 is not systematically analysed, making the identification and quantification of carbonate minerals difficult. Hence, the normative approximation of CO_2 from the LOI is an essential tool; one that enables the CONSONORM_HG method to approximate accurately the amount of carbonate and wollastonite contained in the samples of Davis & Ferry (1993) (Fig. 8a). This estimate is essential to the classification of metamorphic rocks (Fig. 7b) and is crucial to the identification and quantification of the carbonatation alteration process, which is another parameter important to exploration geology.

Calculating volatile elements – As mentioned above, the CO_2 and H_2O^+ estimates proposed by CONSONORM_HG may have useful applications. To properly estimate these parameters, CONSONORM_HG subtracts the H_2O^+ estimated from the hydrated normative phases from the LOI and attributes the remaining amount of volatiles to CO_2. There is no place in this equation for sulphur, which must thus be analysed in order to be removed from the LOI.

To illustrate this aspect of CONSONORM_HG, we used micaceous carbonate samples (Evans & Bickle, 2005) that contain up to 3.1 wt% pyrrhotite, but that were not analysed for sulphur, CO_2 and $H2O^+$. The calculated minerals (Table 3) correctly approximate the observed mineral assemblages for rocks containing <0.5 wt% sulphides and poorly approximate mineral proportions for rocks richer in sulphides for the following reasons: as sulphur was not analysed, the excess FeO from the pyrrhotite was allocated to biotite, causing an overestimation of the amount of H_2O^+ consumed by the hydrous silicates and an underestimation in the amount of normative CO_2 and in the proportion of carbonates (Fig. 8b). As this example illustrates, it is essential to analyse or to estimate sulphur for

samples containing >0.5–1 wt% sulphides in order to use the full potential of the CONSONORM_HG method.

Alteration indices

Most orebodies are surrounded by hydrothermally altered rocks. Recognising and quantifying alteration is thus essential to mining exploration. Most alteration processes form large amounts of a particular type of mineral, designated as the "marker mineral" in this contribution. For example, chloritisation forms a large amount of chlorite in altered rocks, which might be turned into cordierite and anthophyllite or sillimanite, for example, once rocks have been brought to higher grade conditions. Thus, the marker minerals of chloritisation are chlorite, cordierite and, sometimes, garnet (see Table 4 for a non-exhaustive list of marker minerals).

Alteration indices calculated from major elements or from normative minerals generally attempt to quantify marker minerals' excesses or deficits. The indices proposed here are calculated as follows:

Alteration index = Marker mineral (wt%) / (Sum of silicates (wt%) + Non-silicate marker mineral (wt%)) (4)

It is recommended to normalise the proportion of marker minerals by the sum of silicates (equation 4) to avoid dilutions caused, for example, by the late emplacement of stringers of sulphides or other minerals unrelated to the primary alteration event to be characterized. Below, some examples of alteration indices calculated from CONSONORM_HG normative minerals are presented.

Montauban VMS, Quebec – Montauban is a massive sulphide deposit surrounded by the following rocks: 1) quartzite; 2) cordierite–anthophyllite- and nodular sillimanite-bearing gneiss; 3) Bt-gneiss and Bt–Musc–Qtz–Feld-gneiss (see Prabhu & Webber, 1984; Bernier & MacLean, 1993). According to mass balance calculations, quartzite are silicified rocks that may have gained some iron, cordierite–anthophyllite- and nodular sillimanite-bearing gneiss have also been altered (see chloritisation, and possibly sericitisation) and Bt-gneiss and Bt–Musc–Qtz–Feld-gneiss are interpreted as the fresh precursors to the altered rocks (Bernier & MacLean, 1993). CONSONORM_HG calculations carried out on these samples (see Table 3) show large amounts of normative cordierite in the cordierite–anthophyllite- and nodular sillimanite-bearing gneiss. Calculation of several alteration indices (Fig. 9a) indicates that index 1 (Cordierite / sum of silicate) discriminates well the rocks that gained Fe–Mg according to the mass balance calculations performed by Bernier & MacLean (1993). Also, rocks altered by the chloritisation event are characterised by index 1 >0.2–0.3 (Fig. 9a).

Hongtoushan VMS, China – Hongtoushan is a VMS deposit located in basaltic to rhyolitic rocks, and its samples have been classified as altered and unaltered (see Zheng *et al.*, 2011). Compared to unaltered rocks, the altered rocks of each lithology have mostly gained Fe and Mg according to mass balance calculations (Zheng *et al.*, 2011). Calculation of alteration index 1 (marker mineral = cordierite) indicates that each rock type has been altered to various degrees by a chloritisation event (Fig. 9b), which induced the Fe–Mg gains calculated by Zheng and collaborators (2011). Also, altered rocks are characterised by index 1 > 0.2, while this index is <0.2 for unaltered rocks (Fig. 9b).

Arunta block, Australia – The Arunta block contains small VMS deposits surrounded by the following rocks: 1) Quartz–feldspar gneiss; 2) Quartz–cordierite gneiss; 3) mafic gneiss (Warren & Shaw, 1985). The quartz–feldspar and mafic gneiss are metamorphosed felsic igneous rocks and basalts, respectively (see Warren & Shaw, 1985). These two rock types have not been altered; while the quartz–cordierite gneiss corresponds to felsic igneous rocks modified by a chloritisation event (see Warren & Shaw, 1985). Calculation of the alteration indices 1 and 4 (marker minerals = cordierite and garnets) confirms that quartz–cordierite gneiss might be the altered (chloritisation) equivalent of quartz–feldspar gneiss (Fig. 9c). Also, alteration indices 2 and 5 (marker mineral = anthophyllite and biotite) do not enable the recognition of the chloritisation event for these rocks.

Challenger gold deposit, Australia – The Challenger gold deposit is surrounded by the following rocks: 1) distal gneiss; 2) proximal gneiss (altered?); 3) Garnet–cordierite-bearing gneiss; 4) Quartz-rich veins; 5) Quartz–feldspar leucosomes (McFarlane *et al.*, 2007; Tomkins & Mavrogenes, 2002). According to mass balance calculations, distal gneiss correspond to the fresh precursors the other rock types, which have gained variable amounts of Al, K, Fe, Mg and Si (see sericitisation, chloritisation and silicification; McFarlane *et al.,* 2007). Calculation of the alteration indices 1 and 4 (marker mineral = cordierite and garnets) indicates that the garnet–cordierite-bearing gneiss and a part of the proximal gneiss rocks contain an excess of garnet and cordierite (Fig. 9d). These minerals may be indicative of the chloritisation event identified by mass balance calculations (see McFarlane *et al.,* 2007).

DISCUSSION AND CONCLUSIONS

CONSONORM_HG is a new method for approximating the main mineral assemblage of mid- to high-grade metamorphic rocks. This norm has been designed for users who need to approximate natural metamorphic assemblages rapidly, who do not have a good control on the thermodynamic parameters that controlled the crystallisation of their samples and/or who process large dataset of samples containing complex assemblages of silicates, carbonates, sulphides and other minerals. This norm is thus a user-friendly alternative to the tedious process of point counting and is to be applied to cases for which the amount of data available and/or the time allocated to the study does not authorize the use of more refined tools.

CONSONORM_HG is designed for silicate-rich, carbonate-rich, Fe–Ti oxide-bearing rocks that may contain sulphides. However, metamorphic reactions involving sulphides have not been studied in detail and CONSONORM_HG is thus unable to approximate properly the mineralogy of sulphide-only rocks. In addition, CO_2 and H_2O are accurately estimated from the LOI only if the rock is either poor in sulphur, or if sulphur has been analysed.

Also, effort has been made to properly approximate the stability fields of complex mineral groups such as amphibole, dark and white micas and garnet, for example. Among other things, the calculation takes into account the effect that the minor element Mn has on the stability field of garnet, because the thermodynamic database and software used to build the ACKNFM tetrahedra take this element into account (see Perple_X; Connolly & Petrini, 2002; Connolly, 2005). In addition to Mn, many minor and trace elements are known to modify the stability field of silicates. However, the role of these elements has

not been fully modelled thermodynamically (for example, Zn may increase the stability field of staurolite; Ashworth, 1975); CONSONORM_HG is consequently blind to the effect that these elements have on the stability fields of silicates. For this reason, the authors expect to observe discrepancies between natural and calculated assemblages, especially for rocks abnormally enriched in trace elements.

Despite its obvious limitations, CONSONORM_HG correctly reproduces the main assemblages of published rocks. It thus provides a proper approximation of the minerals that make up most of the volume of a metamorphic rock, and ignores minerals that were not targeted by the model, such as un-equilibrated prograde or retrograde phases and unusual phases not yet thermodynamically described. As demonstrated in the last section of this contribution, it is now possible to use CONSONORM_HG as a basis for new standardised classifications for metamorphic rocks and for identifying hydrothermally altered rocks using normative mineral-based indices, among other possible applications.

ACKNOWLEDGEMENTS

The authors wishes to address special thanks to the editor and reviewers of the manuscript, Richard White, Jean-François Moyen and Mark Pearce, whose comments contributed to greatly improve the quality of this article. This study was performed on behalf of the CONSOREM research group (Consortium de Recherche en Exploration Minérale). This project was supported by Canada Economic Development, the Ministère de l'Énergie et des Ressources naturelles du Québec, the Conférence régionale des élus Saguenay-Lac-Saint-Jean and companies members of the CONSOREM. The authors address warm thanks to their colleagues, Silvain Rafini and Stéphane Faure for constructive comments. The authors are indebted to the mining company' members of CONSOREM for defining this project of for stimulating discussions on this topic. Special thanks are addressed to Michel Jébrak, UQAM, for stimulating discussions and to the authors of Perple_X and Theriak-Domino codes for providing such useful tools to the scientific community.

ANNEXE: CONSONORM_HG CODE

The CONSONORM_HG norm is provided as three Visual Basic classes. One of the classes contains the calculation sequence for a single rock sample (CONSONORM_HG.vb). Another class defines the object used to store chemical and normative mineral data. The last class serves as an interface with the user. It is used to input options and data to the code, to calculate several samples successively and to output the data as a .txt file (FormMain.vb). If compilation is needed, note that the authors successfully used InstallShield, by Flexera Software LLC, for their own compilations. The code contained in these three classes is copied to a .pdf file (supplementary material). This file contains instructions for importing the code to the "Microsoft Visual Studio" software, as well as instructions regarding the format of input data.

The density data contained in the code are from Piché & Jébrak (2004) and from the "webmineral.com" website (developed by David Barthelmy). We also used the "MathNet.Numerics.LineraAlgebra" module to define matrices (see http://numerics.mathdotnet.com).

Once performed, the calculation provides the following outputs:

1) Weight percent of normative minerals, re-calculated to 100%.

2) Values for the GOI, H_2O^+ and CO_2 normative, H_2O^+ contained in the minerals and the estimated density of the sample.

3) Sum of the analysed major and trace elements provided to the code prior to being re-calculated to 100% ("TOTALoxyde" field). This serves as a quality control; indeed, if this sum is very different from 100%, it is recommended that the input data be checked.

4) Sum of the normative minerals prior to being re-calculated to 100% ("TOTALmineral" field). This serves as a quality control; if this sum is very different from 100%, it means that the normative calculation failed and the results should be discarded.

REFERENCES

Ashworth, J.R., 1975. Staurolite at anomalously high grade. *Contributions to Mineralogy and Petrology*, **53**(4), 281-291.

Barros Gomes, C., Santini, P. & Dutra, C.V., 1964. Petrochemistry of a Precambrian Amphibolite from the Jaraguá Area, São Paulo, Brazil. *The Journal of Geology*, **72**(5), 664-680.

Barth, T.F.W., 1959. Principles of classification and norm calculations of metamorphic rocks. *Journal of geology*, **67**, 135-152.

Barthelmy, D., 2014. website, http://webmineral.com

Bernier, L., 1992. Caractéristiques géologiques, lithogéochimiques et pétrologiques des gîtes polymétalliques de Montauban et de Dussault./Geological, lithogeochemical and petrological caracteristics of the Montauban and Dussault polymetallique deposits. Ministère de l'Énergie et des Ressources du Québec (gouvernement Québec), report DV-92-03, 31-34.

Bernier, L. & MacLean, W.H. 1993. Lithogeochemistry of a metamorphosed VMS alteration zone at Montauban Grenville Province, Quebec. *Exploration and Mining Geology*, **2**, 367-386.

Brunsmann, A., Franz, G., Erzinger, J. & Landwehr, D., 2000. Zoisite-and clinozoisite-segregations in metabasites (Tauern Window, Austria) as evidence for high-pressure fluid-rock interaction. *Journal of Metamorphic Geology*, **18**(1), 1-22.

Buddington, A.F., 1952. Chemical petrology of some metamorphosed Adirondack gabbroic, syenitic and quartz syenitic rocks. *American Journal of Science,* **Bowen volume**, 37-84.

Cameron, E.M. & Hattori, K., 1985. The Hemlo gold deposit, Ontario: A geochemical and isotopic study. *Geochimica et Cosmochimica Acta,* **49**(10), 2041-2050.

Cohen, D. & Ward, C.R., 1991. SEDNORM—a program to calculate a normative mineralogy for sedimentary rocks based on chemical analyses. *Computers & Geosciences*, **17**(9), 1235-1253.

Connolly, J.A.D., 2005. Computation of phase equilibria by linear programming: a tool for geodynamic modeling and its application to subduction zone decarbonation. *Earth and Planetary Science Letters*, **236**(1), 524-541.

Connolly, J.A.D. & Petrini, K., 2002. An automated strategy for calculation of phase diagram sections and retrieval of rock properties as a function of physical conditions. *Journal of Metamorphic Geology*, **20**(7), 697-708.

Coutinho, J., Kräutner, H., Sassi, F., Schmid, R. & Sisir, S., 2007. Amphibolite and Granulite. Recommendations by the IUGS Subcommission on the Systematics of Metamorphic Rocks, Web version 01.02.2007, https://www.bgs.ac.uk/scmr/products.html

Cross, W., Iddings, J.P., Pirrson, L.V. & Washington, H.S., 1902. A quantitative chemicomineralogical classification and nomenclature of igneous rocks. *Journal of Geology*, **10**, 555-590.

Cross, W., Iddings, J.P., Pirsson, L.V. & Washington, H.S., 1912. Modifications of the "Quantitative System of Classification of Igneous Rocks. *The Journal of Geology*, **20**, 550-561.

Davis, S.R. & Ferry, J.M., 1993. Fluid infiltration during contact metamorphism of interbedded marble and calc-silicate hornfels, Twin Lakes area, central Sierra Nevada, California. *Journal of Metamorphic Geology*, **11**(1), 71-88.

De Capitani, C. & Petrakakis, K., 2010. The computation of equilibrium assemblage diagrams with Theriak/Domino software. *American Mineralogist*, **95**, 1006-1016.

De La Roche, H., Leterrier, J., Grandclaude, P. & Marchal, M., 1980. A classification of volcanic and plutonic rocks using R1-R2-diagram and major-element analyses—Its relationships with current nomenclature. *Chemical geology*, **29**(1), 183-210.

Evans, K.A. & Bickle, M.J., 2005. An investigation of the relationship between bulk composition, inferred reaction progress and fluid-flow parameters for layered micaceous carbonates from Maine, USA. *Journal of Metamorphic Geology*, **23**(3), 181-197.

Fears, D., 1985. A corrected CIPW program for interactive use. *Computers & Geosciences*, **11**(6), 787-797.

Foucault, R., 1992. Dictionnaire de géologie / Dictionnary of geology. E.d. Masson, Paris, France, p. 352.

Fritz, S.F. & Popp, R.K., 1985. A single-dissolution technique for determining FeO and Fe2O3 in rock and mineral samples. *American Mineralogist,* **70**(9-10), 961-968.

GEOROC 2014. website, http://georoc.mpch-mainz.gwdg.de/georoc/

Glazner, A.F., 1984. A short CIPW norm program. *Computers & Geosciences*, **10**(4), 449-450

Gu, L., Zheng, Y., Tang, X., Zaw, K., Della-Pasque, F., Wu, C. & Wang, X., 2007. Copper, gold and silver enrichment in ore mylonites within massive sulphide orebodies at Hongtoushan VHMS deposit, NE China. *Ore Geology Reviews*, **30**(1), 1-29.

Holland, T.J.B. & Powell, R., 1998. An internally consistent thermodynamic data set for phases of petrological interest. *Journal of metamorphic geology*, **16**, 309-343.

Johnson, J.W., Oelkers, E.H. & Helgeson, H.C., 1992. SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bar and 0 to 1000 C. *Computers & Geosciences*, **18**(7), 899-947.

Kelsey, C.H., 1965. Calculation of the CIPW norm. *Mineralogical Magazine*, **34**, 276-282.

Le Maître, R.W., 1976. The chemical variability of some common igneous rock. *Journal of Petrology*, **17**, 589-639.

McFarlane, C.R., Mavrogenes, J.A. & Tomkins, A.G., 2007. Recognizing hydrothermal alteration through a granulite facies metamorphic overprint at the Challenger Au deposit, South Australia. *Chemical geology*, **243**(1), 64-89.

Middlemost, E.A., 1989. Iron oxidation ratios, norms and the classification of volcanic rocks. *Chemical Geology*, **77**(1), 19-26.

Muir, T.L., 2002. The Hemlo gold deposit, Ontario, Canada: principal deposit characteristic and constraints on mineralization. *Ore Geology Reviews*, **21**(1), 1-66.

Novák, J. K. & Vrbová, H., 1996. Petrogenesis and geochemistry of mafic rocks from the Kutná Hora Crystalline Complex and the neighbouring part of the Rataje Micaschist Zone. *Geological Institute, Academy of Sciences of the Czech Republic, Geolines*, **4**, 1-41.

Perple_X 2014. website, http://www.perplex.ethz.ch/

Piché, M. & Jébrak, M., 2004. Normative minerals and alteration indices developed for mineral exploration. *Journal of Geochemical Exploration*, **82**, 59-77.

Prabhu, M.K. & Webber, G.R., 1984. Origin of quartzofeldspathic gneisses at Montauban-les-Mines, Québec. *Canadian Journal of Earth Sciences*, **21**(3), 336-345.

Pruseth, K., 2009. Calculation of the CIPW norm: new formulas. *Journal of Earth Science Systems,* **118**(1), 101-113.

Rosen, B., Desmons, J. & Fettes, D., 2007. Metacarbonate and related rocks. Recommendations by the IUGS Subcommission on the Systematics of Metamorphic Rocks, Web version 01.02.2007, https://www.bgs.ac.uk/scmr/products.html

Spear, F.S., 1994. Metamorphic phase equilibria and pressure–temperature-time paths. *Mineralogical Society of America Monographs*, p. 799.

Streckeisen, A., 1976. To each plutonic rock its proper name. *Earth-Science Reviews,* **12**(1), 1-33.

Tomkins, A.G. & Mavrogenes, J.A., 2001. Redistribution of gold within arsenopyrite and löllingite during pro-and retrograde metamorphism: application to timing of mineralization. *Economic Geology*, **96**(3), 525-534.

Tomkins, A.G. & Mavrogenes, J.A., 2002. Mobilization of gold as a polymetallic melt during pelite anatexis at the Challenger deposit, South Australia: a metamorphosed Archean gold deposit. *Economic Geology*, **97**(6), 1249-1271.

Tomkins, A.G., Pattison, D.R. & Zaleski, E., 2004. The Hemlo gold deposit, Ontario: An example of melting and mobilization of a precious metal-sulfosalt assemblage during amphibolite facies metamorphism and deformation. *Economic Geology,* **99**(6), 1063-1084.

Torres-Roldan, R.L., Garcia-Casco, A. & Garcia-Sanchez, P.A., 2000. CSpace: an integrated workplace for the graphical and algebraic analysis of phase assemblages on 32-bit Wintel platforms. *Computers & Geosciences*, **26**(7), 779-793.

Trépanier, S., Mathieu, L. & Daigneault, R., In press. CONSONORM_LG: new normative minerals and alteration indexes for low-grade metamorphic rocks. *Economic Geology*

Verma, S.P., Torres-Alvarado, I.S. & Velasco-Tapia, F., 2003. A revised CIPW norm. *Swiss Bulletin of Mineralogy and Petrology*, **83**(2), 197-216.

Vrana, S., Stedra, V. & Fisera, M., 2005. Petrology and geochemistry of the Bestvina granulite body metamorphosed at eclogite facies conditions, Bohemian Massif. *Journal of Geosciences*, **50**(3-4), 95-106.

Warren, R.G. & Shaw, R.D., 1985. Volcanogenic Cu–Pb–Zn bodies in granulites of the central Arunta Block, central Australia. *Journal of Metamorphic Geology*, **3**(4), 481-499.

Zheng, Y.C., Gu, L., Tang, X., Wu, C., Li, C. & Liu, S., 2011. Geology and geochemistry of highly metamorphosed footwall alteration zones in the Hongtoushan volcanogenic massive sulfide deposit, Liaoning Province, China. *Resource geology*, **61**(2), 113-139.

SUPPORTING INFORMATION

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

File 1.pdf. Copy of the CONSONORM_HG code, and instructions regarding its importation to "Microsoft Visual Studio" software.

File 2.pdf. Graphical representations of the ternary diagrams and tetrahedra of CONSONORM_HG.

FIGURE CAPTIONS

Fig. 1. Sketch of a hypothetical ACKNFM tetrahedron (*main tetrahedron*) and its constitutive *small tetrahedra*.

Fig. 2. The 17 models of CONSONORM_HG displayed on a *P–T* diagram. The letters and numbers designate the model as follows: 1) the first number is the pressure (for example, 9 stands for 9 kbars); 2) the letters are an abbreviation of the facies (AMP for amphibolite, GRA for granulite, SV for green schist and SB for blue schist); 3) the last number designates the temperature (for example, 750 stands for 750 degrees Celsius).

Fig. 3. Calculation sequence of CONSONORM_HG (see text for details).

Fig. 4. Sketch summarising the strategy used by CONSONORM_HG to select the main silicate assemblage to be calculated, using a succession of tetrahedra.

Fig. 5. Sketch of a hypothetical SiO_2–$CaCO_3$–$FeCO_3$–$MgCO_3$ tetrahedron used to determine if the normative assemblage may contain quartz and carbonates or if these minerals must react to form graphite and additional silicates.

Fig. 6. Binary diagram that compares observed to calculated minerals. The 120 samples presented in this graph are from Buddington (1952), Barros Gomes *et al.*, (1964), Davis & Ferry (1993), Novak & Vrbova (1996), Brunsmann *et al.*, (2000) and Vrana *et al.*, (2005) (see Tables 2 and 3 for more details). The observed minerals' proportions (volume %) have been converted (wt%) using density data from Barthelmy (2014).

Fig. 7. Ternary diagrams used to: (a) classify rocks of the amphibolite facies (diagram from Coutinho *et al.*, 2007) and; (b) to classify carbonate-bearing rocks (diagram from Rosen *et al.*, 2007). The samples presented in these diagrams are: (a) amphibolite (data from Barros Gomes *et al.*, 1964); (b) marbles (Davis & Ferry 1993) and other types of metamorphic rocks (Buddington, 1952; Barros Gomes *et al.*, 1964; Novak & Vrbova, 1996; Brunsmann *et al.*, 2000; Vrana *et al.*, 2005; n= 48).

Fig. 8. (a) Binary diagram comparing the samples from Davis & Ferry (1993) to normative minerals; (b) Binary diagram comparing the samples from Evans & Bickle (2005) to normative minerals. The numbers correspond to the proportion of pyrrhotite (PO) observed in the samples. The observed minerals' proportions (volume %) have been converted (wt%) using density data from Barthelmy (2014).

Fig. 9. Box plots displaying the values of various alteration indexes calculated for the following deposits: (a) Montauban (data from Prabhu & Webber, 1984; Bernier & MacLean, 1993); (b) Hongtoushan (Zheng *et al.*, 2011); (c) Arunta block (Warren & Shaw, 1985); d) Challenger (McFarlane *et al.*, 2007; Tomkins & Mavrogenes, 2002).

TABLE CAPTIONS

Table 1. Minerals (wt%) calculated for a sample of the Hemlo deposit.

Table 2. Samples used to validate the CONSONORM_HG calculation.

Table 3. Parameters used to perform the CONSONORM_HG calculation.

Table 4. Marker minerals for alteration indexes.

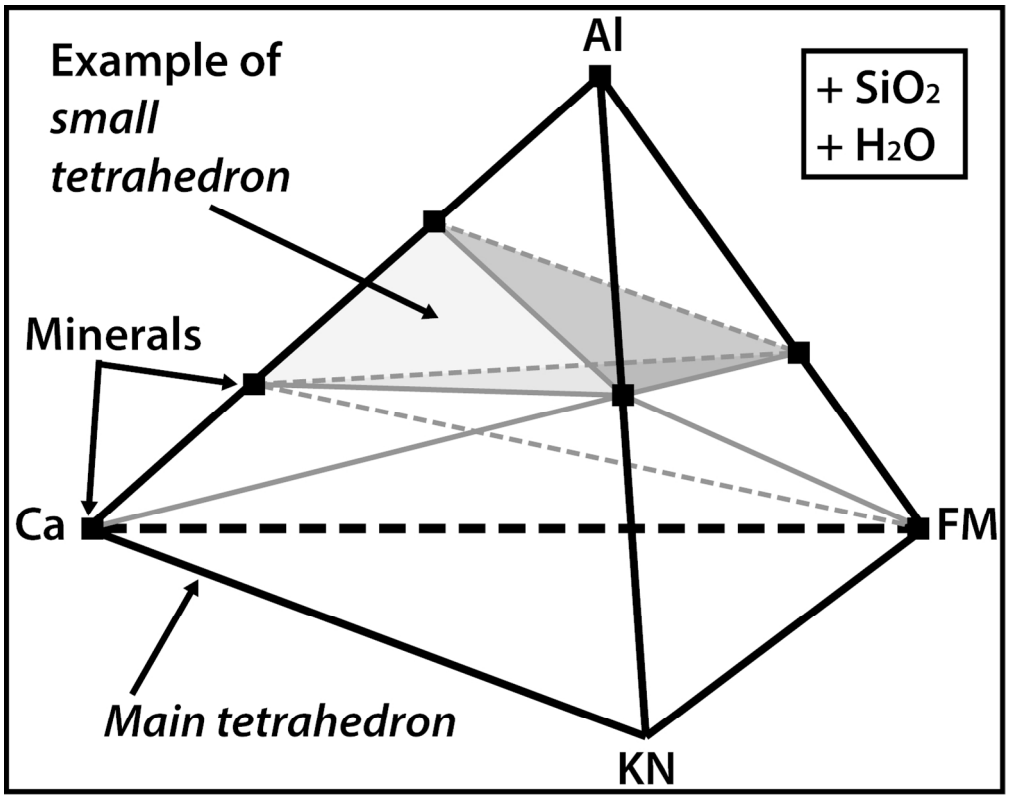


Fig. 1

Sketch of a hypothetical ACKNFM tetrahedron (main tetrahedron) and its constitutive small tetrahedra.
132x120mm (300 x 300 DPI)

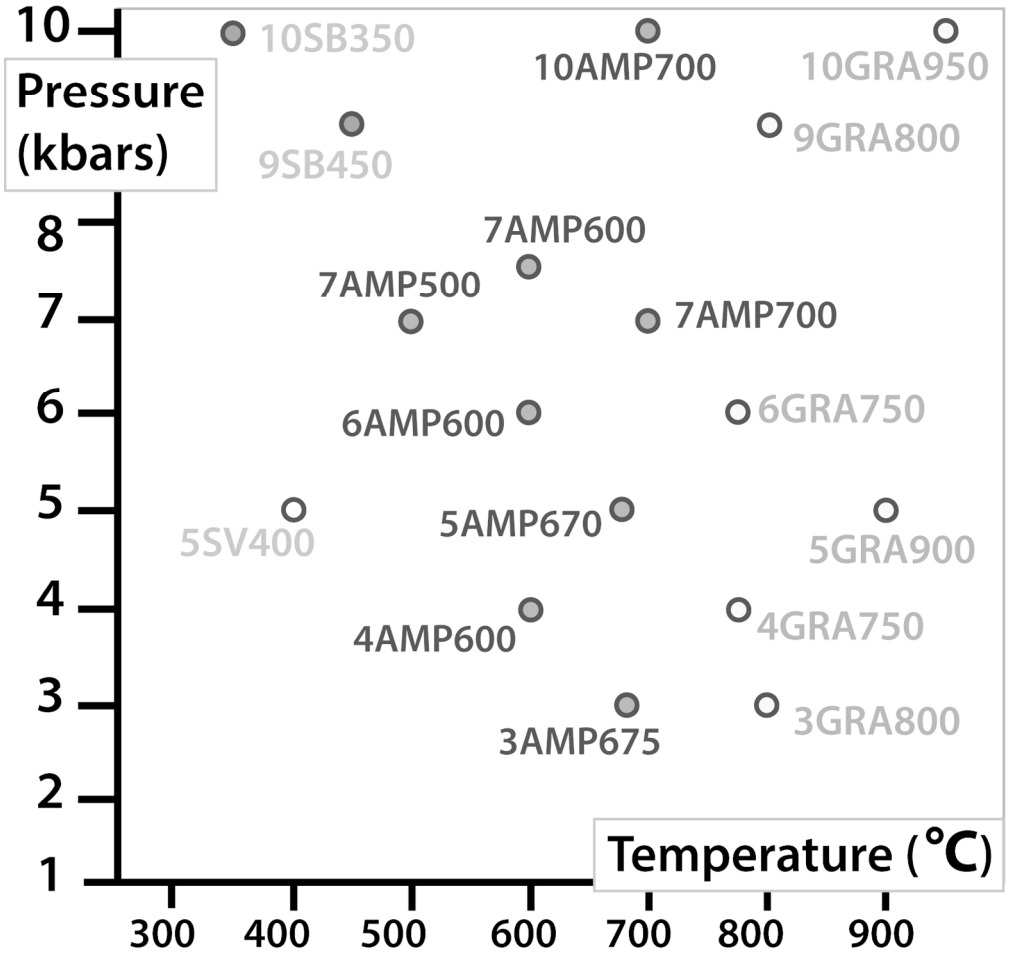


Fig. 2

The 17 models of CONSONORM_HG displayed on a P–T diagram. The letters and numbers designate the model as follows: 1) the first number is the pressure (for example, 9 stands for 9 kbars); 2) the letters are an abbreviation of the facies (AMP for amphibolite, GRA for granulite, SV for green schist and SB for blue schist); 3) the last number designates the temperature (for example, 750 stands for 750 degrees Celsius).

143x156mm (300 x 300 DPI)

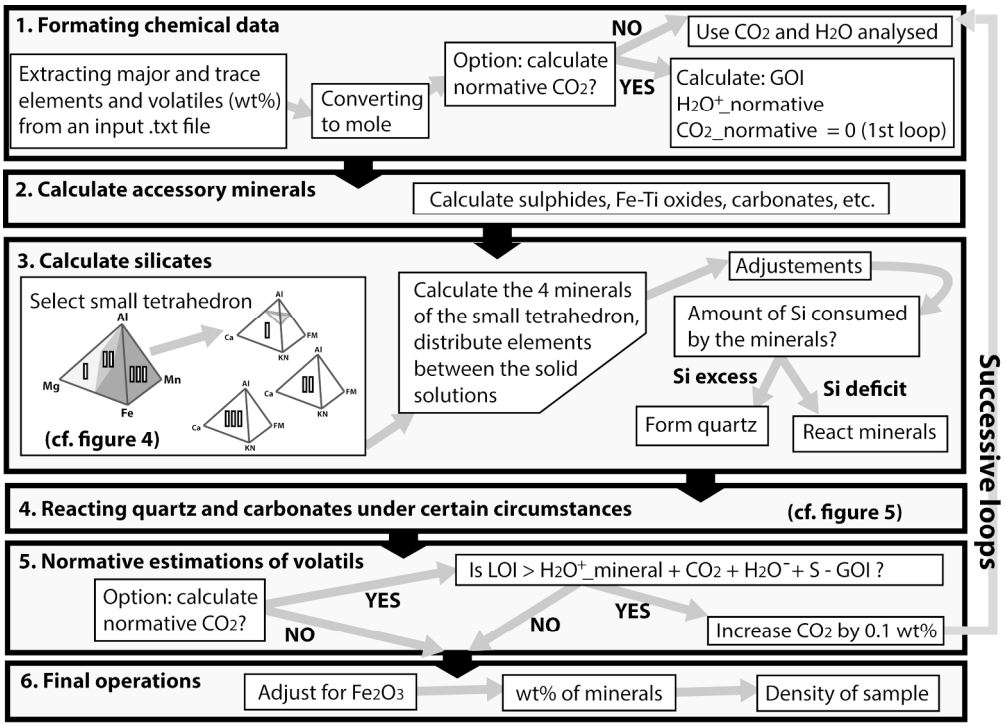


Fig. 3

Calculation sequence of CONSONORM_HG (see text for details).
192x160mm (300 x 300 DPI)

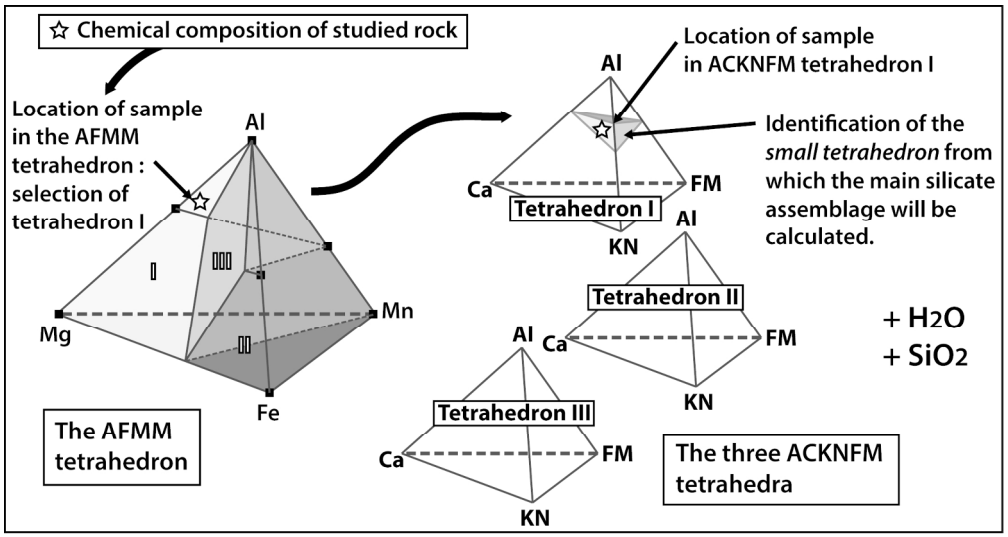


Fig. 4

Sketch summarising the strategy used by CONSONORM_HG to select the main silicate assemblage to be calculated, using a succession of tetrahedra.
180x116mm (300 x 300 DPI)

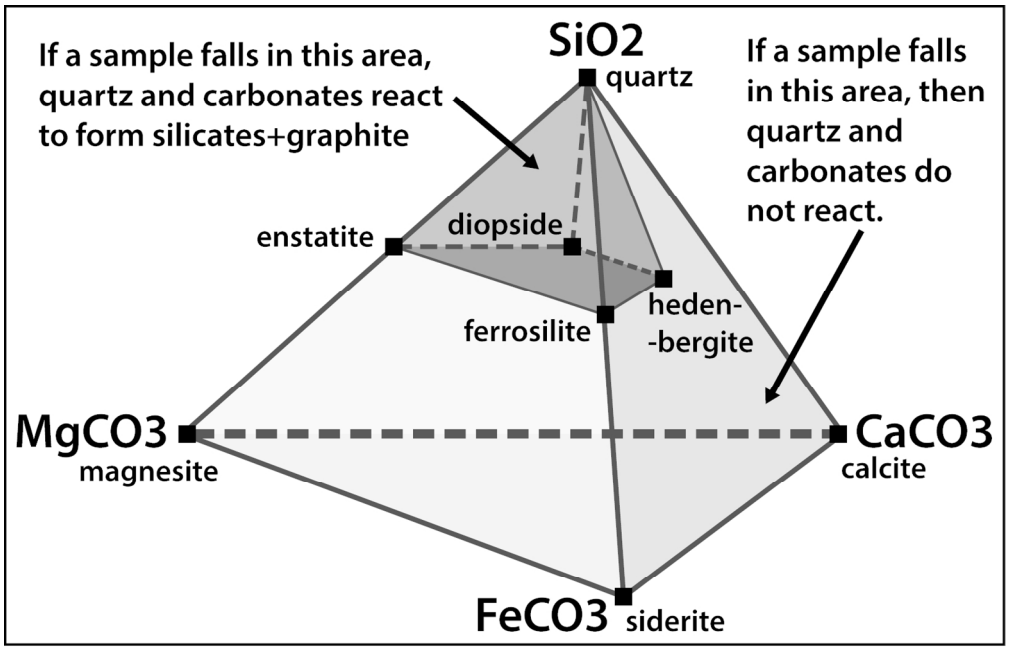


Fig. 5

Sketch of a hypothetical SiO2–CaCO3–FeCO3–MgCO3 tetrahedron used to determine if the normative assemblage may contain quartz and carbonates or if these minerals must react to form graphite and additional silicates.
137x104mm (300 x 300 DPI)

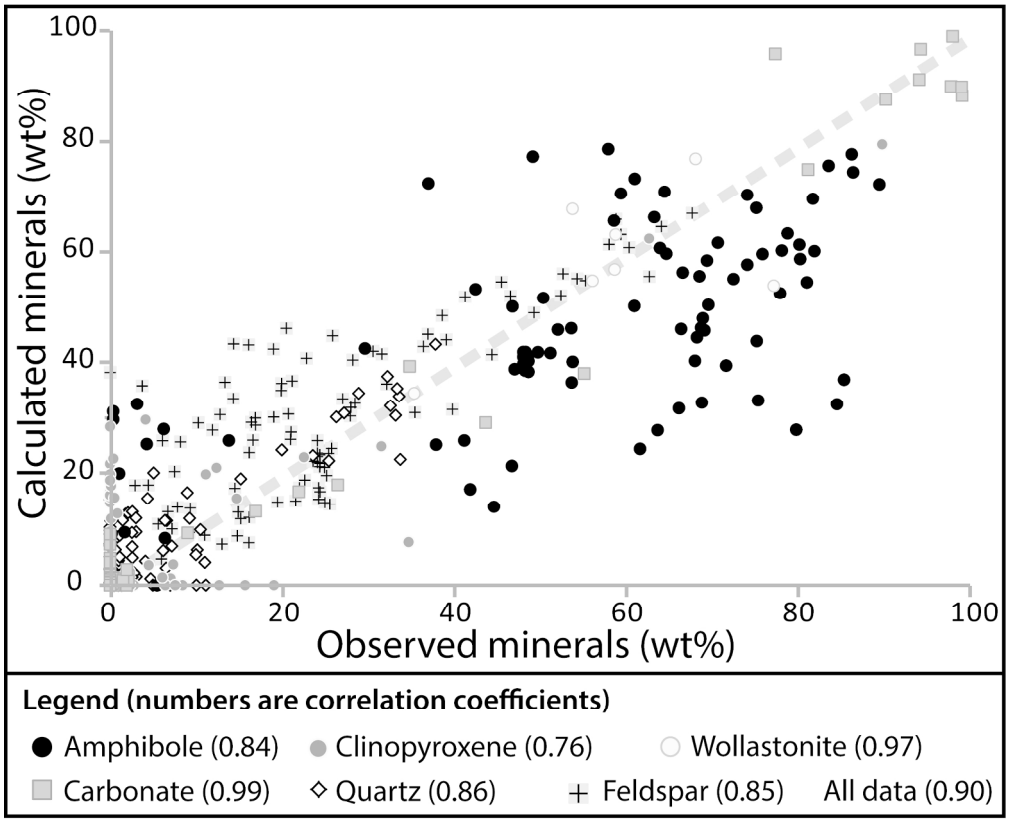


Fig. 6

Binary diagram that compares observed to calculated minerals. The 120 samples presented in this graph are from Buddington (1952), Barros Gomes et al., (1964), Davis & Ferry (1993), Novak & Vrbova (1996), Brunsmann et al., (2000) and Vrana et al., (2005) (see Tables 2 and 3 for more details). The observed minerals' proportions (volume %) have been converted (wt%) using density data from Barthelmy (2014).
193x178mm (300 x 300 DPI)

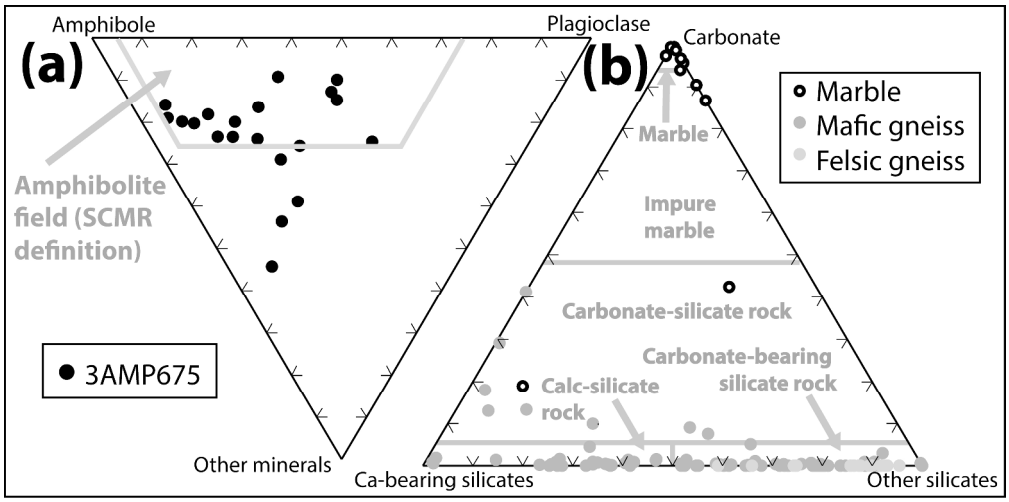


Fig. 7

Ternary diagrams used to: (a) classify rocks of the amphibolite facies (diagram from Coutinho et al., 2007) and; (b) to classify carbonate-bearing rocks (diagram from Rosen et al., 2007). The samples presented in these diagrams are: (a) amphibolite (data from Barros Gomes et al., 1964); (b) marbles (Davis & Ferry 1993) and other types of metamorphic rocks (Buddington, 1952; Barros Gomes et al., 1964; Novak & Vrbova, 1996; Brunsmann et al., 2000; Vrana et al., 2005; n= 48).
364x210mm (300 x 300 DPI)

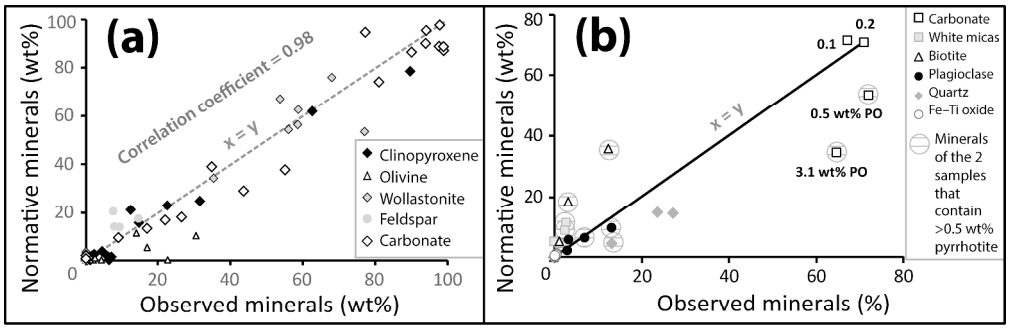


Fig. 8

(a) Binary diagram comparing the samples from Davis & Ferry (1993) to normative minerals; (b) Binary diagram comparing the samples from Evans & Bickle (2005) to normative minerals. The numbers correspond to the proportion of pyrrhotite (PO) observed in the samples. The observed minerals' proportions (volume %) have been converted (wt%) using density data from Barthelmy (2014).
305x122mm (300 x 300 DPI)

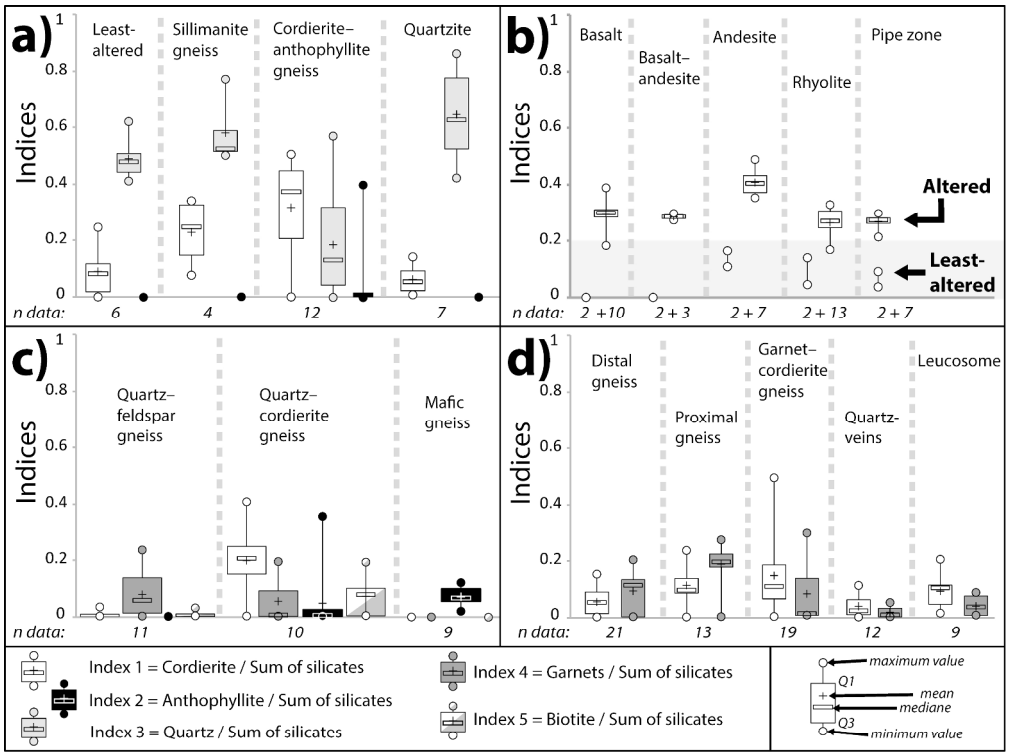


Fig. 9

Box plots displaying the values of various alteration indexes calculated for the following deposits: (a) Montauban (data from Prabhu & Webber, 1984; Bernier & MacLean, 1993); (b) Hongtoushan (Zheng et al., 2011); (c) Arunta block (Warren & Shaw, 1985); d) Challenger (McFarlane et al., 2007; Tomkins & Mavrogenes, 2002).
289x240mm (300 x 300 DPI)

TABLES

Table 1. Minerals (wt%) calculated for a sample of the Hemlo deposit.

Minerals (wt%)	CONSONORM_HG	Theriak-Domino
Albite	0.82	5.87
Orthose	23.96	15.07
Anorthite	0.64	1.38
White mica	13.68	13.61
Biotite	1.44	13.02
Quartz	42.41	43.44
Garnet		3.42
Oxide Fe-Ti	2.55	4.16
Barite	1.03	
Pyrite	12.59	
Calcite	0.35	
Other	0.52	0.2

Table 2. Samples used to validate the CONSONORM_HG calculation.

Reference	Rock type	Metamorphic grade*
Cameron & Hattori (1985)	Sericite schist	~ 6-7 kbars, 600-650°C (Tomkins *et al.,* 2004)
Buddington (1952)	Meta-syenite, charnockites	upper-amphibolite to granulite
Barros Gomes *et al.* (1964)	Metabasite	amphibolite
Davis & Ferry (1993)	Marbles, calc-silicate hornfels	~ 3 kbars, 630°C (amphibolite)
Prabhu & Webber (1984), Bernier & MacLean (1993)	Quartzite, quartzofeldspathic gneiss, etc.	~ 6.5 kbars, 620°C (amphibolite; Bernier, 1992)
Novak & Vrbova (1996)	Meta-mafic rocks	amphibolite (possibly retrograde eclogites)
Brunsmann *et al.* (2000)	Metabasite	~ over 6 kbars, 500-550°C (amphibolite)
Vrana *et al.* (2005)	Felsic granulites, migmatites	~ 18-22 kbars, 800-920°C (eclogite)
Evans & Bickle (2005)	Micaceous meta-carbonates	~ 3.5 kbars, 450°C (amphibolite)
Zheng *et al.* (2011)	Mafic to felsic meta-volcanites	~ 600-650°C (amphibolite; Gu *et al*., 2007)
Warren & Shaw (1985)	Mafic to felsic gneiss	~ 8 kbars, 850-920°C (granulite)
McFarlane *et al.* (2007), Tomkins & Mavrogenes (2002)	Gneiss, veins, migmatites	~ 7 kbars, 800°C (granulite; Tomkins & Mavrogenes, 2001)

*Metamorphic grades were determined by the authors referenced in this table, unless indicated otherwise.

Table 3. Parameters used to perform the CONSONORM_HG calculation.

Reference	Facies	Fe2O3/Fe2O3T	CO2	n data
Cameron & Hattori (1985)	6AMP600	analysed*	analysed	1
Buddington (1952)	5AMP670	analysed	analysed	20
Barros Gomes *et al.* (1964)	3AMP675	analysed	analysed	20
Davis & Ferry (1993)	3AMP675	0.2	normative estimate	19
Prabhu & Webber (1984)	7AMP700	0.2	normative estimate	5
Bernier & MacLean (1993)	7AMP700	analysed	normative estimate	25
Novak & Vrbova (1996)	3AMP675	analysed	analysed	34
Brunsmann *et al.* (2000)	7AMP500	analysed	normative estimate	16
Vrana *et al.* (2005)	10AMP700	analysed	analysed	6
Evans & Bickle (2005)	6AMP600	analysed	normative estimate	4
Zheng *et al.* (2011)	5AMP670	0.2	normative estimate	51
Warren & Shaw (1985)	9GRA800	analysed or 0.2	analysed	30
McFarlane *et al.* (2007), Tomkins & Mavrogenes (2002)	5GRA900	0.2	(LOI unavailable)	74

*Normative calculation performed with analysed values of FeO and Fe_2O_3.

**All the normative calculations are performed with a $Fe_2O_3/(Fe_2O_3+FeO)$ molar ratio for Fe-Ti-oxides of 0.5.

Table 4. Marker minerals for alteration indexes.

Alteration	Marker minerals (at various metamorphic grade)
Chloritisation (Fe-Mg)	Chlorite, pyrope-almandine, cordierite, carpholite, sudoite
Chloritisation + acidic (Fe-Mg-Al)	Staurolite, chloritoid, spinel
Argilisation (Al)	Aluminosilicate
Hematisation (Fe)	Hematite, magnetite
Biotitisation (Fe-Mg-K)	Biotite
Sericitisation (Na-K)	Muscovite, paragonite
Potassic alteration (K)	Orthose, leucite
Albitisation (Na)	Albite, nepheline
Epidotisation (Ca)	Epidote, wollastonite, grossular, anorthite, lawsonite
Propylitic alteration (Ca-Fe-Mg)	Actinolite, margarite, clinopyroxene, hornblende, clinozoisite
Silicification (Si)	Quartz
Carbonatation (C)	Carbonates

Minerals (wt%)	CONSONORM_HG	Theriak-Domino
Albite	0.82	5.87
Orthose	23.96	15.07
Anorthite	0.64	1.38
White mica	13.68	13.61
Biotite	1.44	13.02
Quartz	42.41	43.44
Garnet		3.42
Oxide Fe-Ti	2.55	4.16
Barite	1.03	
Pyrite	12.59	
Calcite	0.35	
Other	0.52	0.2

Reference	**Rock type**
Cameron & Hattori (1985)	Sericite schist
Buddington (1952)	Meta-syenite, charnockites
Barros Gomes *et al.* (1964)	Metabasite
Davis & Ferry (1993)	Marbles, calc-silicate hornfels
Prabhu & Webber (1984), Bernier & MacLean (1993)	Quartzite, quartzofeldspathic gneiss, etc.
Novak & Vrbova (1996)	Meta-mafic rocks
Brunsmann *et al.* (2000)	Metabasite
Vrana *et al.* (2005)	Felsic granulites, migmatites
Evans & Bickle (2005)	Micaceous meta-carbonates
Zheng *et al.* (2011)	Mafic to felsic meta-volcanites
Warren & Shaw (1985)	Mafic to felsic gneiss
McFarlane *et al.* (2007), Tomkins & Mavrogenes (2002)	Gneiss, veins, migmatites

Metamorphic grade*
~ 6-7 kbars, 600-650°C (Tomkins *et al.,* 2004)
upper-amphibolite to granulite
amphibolite
~ 3 kbars, 630°C (amphibolite)
~ 6.5 kbars, 620°C (amphibolite; Bernier, 1992)
amphibolite (possibly retrograde eclogites)
~ over 6 kbars, 500-550°C (amphibolite)
~ 18-22 kbars, 800-920°C (eclogite)
~ 3.5 kbars, 450°C (amphibolite)
~ 600-650°C (amphibolite; Gu *et al* ., 2007)
~ 8 kbars, 850-920°C (granulite)
~ 7 kbars, 800°C (granulite; Tomkins & Mavrogenes, 2001)

Reference	Facies	Fe2O3/Fe2O3T
Cameron & Hattori (1985)	6AMP600	analysed*
Buddington (1952)	5AMP670	analysed
Barros Gomes *et al.* (1964)	3AMP675	analysed
Davis & Ferry (1993)	3AMP675	0.2
Prabhu & Webber (1984)	7AMP700	0.2
Bernier & MacLean (1993)	7AMP700	analysed
Novak & Vrbova (1996)	3AMP675	analysed
Brunsmann *et al.* (2000)	7AMP500	analysed
Vrana *et al.* (2005)	10AMP700	analysed
Evans & Bickle (2005)	6AMP600	analysed
Zheng *et al.* (2011)	5AMP670	0.2
Warren & Shaw (1985)	9GRA800	analysed or 0.2
McFarlane *et al.* (2007), Tomkins & Mavrogenes	5GRA900	0.2

CO2	n data
analysed	1
analysed	20
analysed	20
normative estimate	19
normative estimate	5
normative estimate	25
analysed	34
normative estimate	16
analysed	6
normative estimate	4
normative estimate	51
analysed	30
(LOI unavailable)	74

Alteration
Chloritisation (Fe-Mg)
Chloritisation + acidic (Fe-Mg-Al)
Argilisation (Al)
Hematisation (Fe)
Biotitisation (Fe-Mg-K)
Sericitisation (Na-K)
Potassic alteration (K)
Albitisation (Na)
Epidotisation (Ca)
Propylitic alteration (Ca-Fe-Mg)
Silicification (Si)
Carbonatation (C)

Marker minerals (at various metamorphic grade)
Chlorite, pyrope-almandine, cordierite, carpholite, sudoite
Staurolite, chloritoid, spinel
Aluminosilicate
Hematite, magnetite
Biotite
Muscovite, paragonite
Orthose, leucite
Albite, nepheline
Epidote, wollastonite, grossular, anorthite, lawsonite
Actinolite, margarite, clinopyroxene, hornblende, clinozoisite
Quartz
Carbonates

ADDITIONAL MATERIAL

Article: CONSONORM_HG: a new method of norm calculation for mid- to high-grade metamorphic rocks, by Mathieu, L., Trépanier, S. & Daigneault, R. (Journal of Metamorphic Geology).

This document contains a copy of the CONSONORM_HG code, and instructions regarding its importation to “Microsoft Visual Studio” software.

Contents

1. Instructions

This file contains a copy of three Visual Basic .NET classes compatible with .NET version 4 and later. The classes are:

- Class 1: FormMain.vb
- Class 2: CONSONORM.vb
- Class 3: ValueMx.vb

This document contains instructions for implementing the .NET classes to the “Microsoft Visual Studio 2013” software (older version of Visual Studio are also possible). The simple procedure described below is accessible to anyone who has already programmed “Hello World”.

a) Open Visual Studio and create a new project as follows:

> File (menu) > New > Project

Select “Application Windows Forms” and save your new project to an emplacement of your choice, by naming it “CONSONORM_HG”.

b) A new project has been created. Rename the default “Form1.vb” as “FormMain.vb” (select “YES” when the pop-up window opens).

c) Copy and paste the code of class 1 (“FormMain.vb”), reproduced below, into the “FormMain.vb” class.

d) Create two new classes by right clicking on the “CONSONORM_HG” project and by selecting:

> Add > New element

Select “Class”, and repeat this operation twice. Name the new classes “CONSONORM” and “ValueMx”.

e) Copy and paste the code of class 2 (“CONSONORM.vb”) and class 3 (“ValueMx.vb”), reproduced below, into the “CONSONORM.vb” and “ValueMx” classes.

Your project now contains the code of the CONSONORM_HG norm, but it cannot be used yet.

f) Log on to the Math.NET website (http://numerics.mathdotnet.com).

Download the Math.NET Iridium package, or install it using the "package manager console" (cf. instructions provided by the Math.NET website). The authors used the 2008.8.16.470 version of this package, but updated versions will work as well.

Once you are done with this step, the Math.NET Iridium package should be referenced. You will notice that the first code line of the "CONSONORM_HG.vb" class stops returning an error message. If this is not the case, please resume the procedure until Math.NET is properly referenced. This .NET compatible module is required to properly compile the VB code.

g) Designing the GUI

Eventually, you will need to design the user interface. Figure S1 proposes a design for a simple GUI. The elements can be arranged at your convenience, but the name of the main controls must be respected.

Figure S1 displays the required controls and their names. The text and other ornamentations are not essential. Instructions for implementing controls in GUIs are available from the Visual Studio website and will not be detailed here.

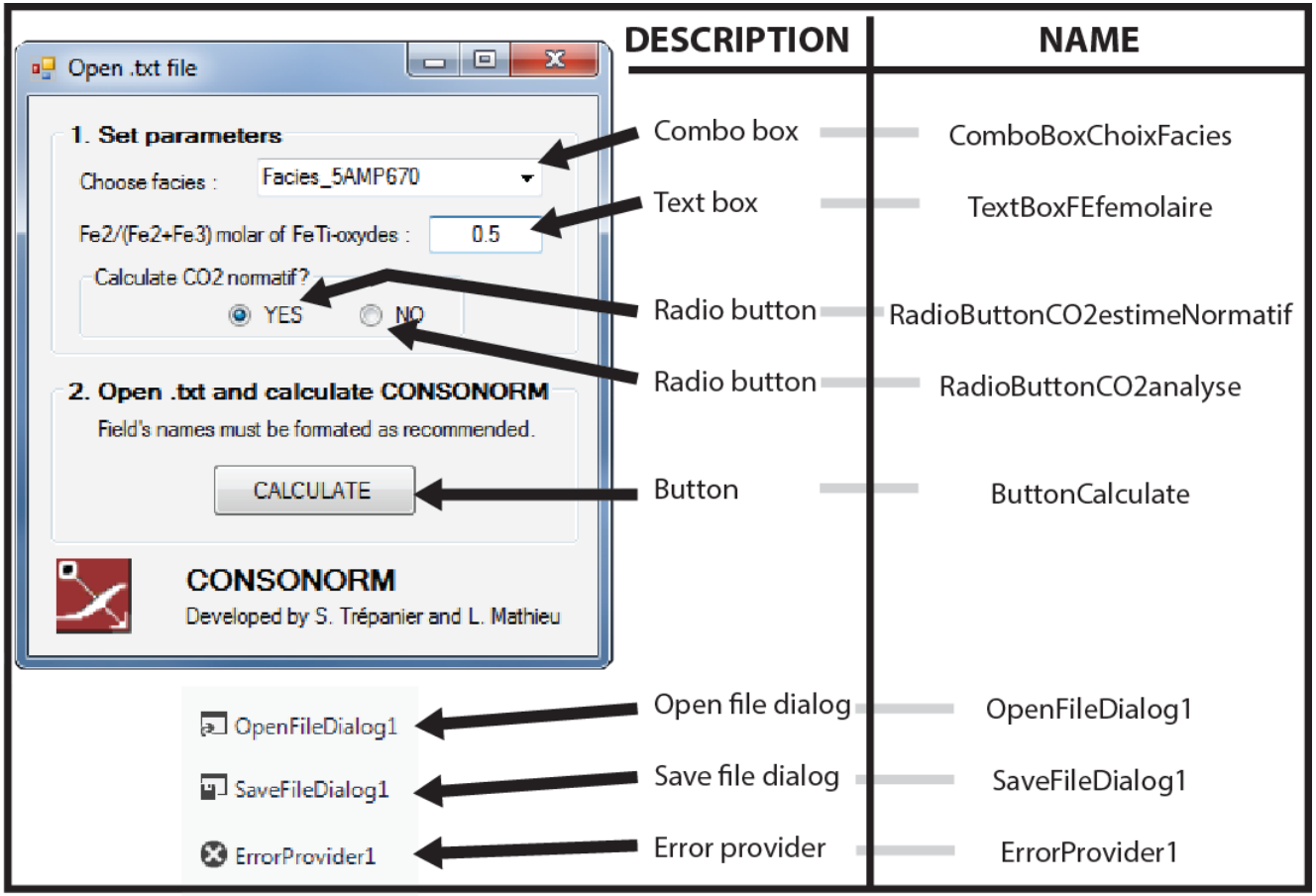


Figure S1: Possible design for the GUI. The controls' names must be respected.

The combo box “ComboBoxChoixFacies” must contain the following elements, that you can copy and paste in it:

Model_3AMP675
Model_3GRA800
Model_5GRA900
Model_5AMP670
Model_7AMP700
Model_7AMP600
Model_10GRA950
Model_9GRA800
Model_10AMP700
Model_7AMP500
Model_4AMP600
Model_5SV400
Model_9SB450
Model_10SB350

The code is now compiled. Please use the Debug button () to start calculating with CONSONORM_HG.

The code will prompt you to open a .txt file. This file must be designed as follows (figure S2):

- The file must contain the names of chemical data on its first line.
- The next lines must contain numerical data (cf. double numbers) that correspond to the chemical analysis of various samples.
- The data must be delimited by a large space (cf. use the tab key of your keyboard).

Chemical data names must be formatted as follows:

- Major elements must be provided in wt% and named as follows: SiO2, Al2O3, TiO2, CaO, Fe2O3, FeO, K2O, Na2O, P2O5, MgO, MnO, H2O_PLUS, H2O_MINUS, CO2, S, LOI
- Trace elements must be provided in ppm and named as follows: B, F, Cl, Cr, Ni, Cu, Zn, As, Zr, Mo, Ba, Pb

The code will provide outputs that you may save as a .txt file (figure S3). For a description of these outputs, please refer to the article.

EvansBickle - Bloc-notes

Fichier Edition Format Affichage ?

SiO2	TiO2	Al2O3	Fe2O3	FeO	MnO	MgO	CaO	Na2O	K2O	P2O5	LOI
23.4	0.2	3.7	0.1	1.5	0.1	3.6	34.7	1.1	0.5	0	30.9
26.8	0.6	13.5	0.4	4.4	0.1	7.5	22.1	1.5	3.5	0.1	17.2
23.4	0.4	8.4	0.3	2.6	0.1	4.4	31.2	1.5	1.8	0	24.7
21.4	0.2	3.1	0.2	1.3	0.1	2.3	37.9	0.7	0.5	0	31.4

Figure S2: Recommended design for the input .txt file. The data displayed are from Evans & Bickle (2005). [**reference:** Evans, K.A. & Bickle, M.J., 2005. An investigation of the relationship between bulk composition, inferred reaction progress and fluid-flow parameters for layered micaceous carbonates from Maine, USA. *Journal of Metamorphic Geology*, **23**(3), 181-197.].

EvansBickle_OUT.txt - Bloc-notes

Fichier Edition Format Affichage ?

Calculation performed for facies: 6AMP600 (6 kbars, 600 Celsius degrees)
CO2 and H2O estimated normatively, by successive itterations.
Ratio Fe3/(Fe2+Fe3) molar used for calculating Fe-Ti oxides = 0.5

albite	amesiteFE	annite	anorthite	fayalite	forsterite	muscovite	nepheline	orthose	paragonite
5.83	0.25	0		0	2.85 4.61	14.53			0.1
2.41	28.55	7.35		0.59 1.21	10.86	7.21	1.46	4.23	0.77
6.45	14.65		0.03 0.06	0.07	8.83	3.84 4.59	0.47 1.7	4.75	0.38
2.3	4.13		0.03 0.07	0.01	5.27	1.06 14.93	0.68	0.2	0.38

Figure S3: Output (partial) provided by the CONSONORM_HG code using data from Evans and Bickle (2005).

2. FormMain.vb

```
Imports System.Threading

Public Class FormMain

    Private CultureInfo As Global.System.Globalization.CultureInfo

    Private GlobalMole As New Dictionary(Of Integer, Dictionary(Of String, ValueMx))
    Private GlobalSample As New Dictionary(Of Integer, Dictionary(Of String, Double))
    Private MxToExport As New List(Of String)
    Private MxCompleteList As New List(Of String)
    Private NaKproblems As New Dictionary(Of String, String)
    Private MgFeproblems As New Dictionary(Of String, String)

    Private CO2normatif As Boolean    'If False, the code will use H2O and CO2 analysed. If true, the code will estimate CO2 from the
LOI.
    Private FEoxydeRatio As Double    'Used to estimate the Fe2O3/(Fe2O3 + FeO) molar ratio of Fe-Ti-oxydes. This ratio can be
estimated from mineralogical observations (cf. amoutn of hematite versus magnetite observed, etc.).
    Private FaciesEnum As enumFacies  'You can perform the calculation for one of the 17 facies available. "3AMP675" is the default
facies, and this can be modified using the following function

    Private CalculationSuccessful As Boolean
    Private Facies As String

    Private Sub ButtonCalculate_Click(sender As Object, e As EventArgs) Handles ButtonCalculate.Click

        Try
            If Me.ValidateChildren Then

                If Me.RadioButtonCO2analyse.Checked Then : CO2normatif = False
                ElseIf Me.RadioButtonCO2estimeNormatif.Checked Then : CO2normatif = True
                End If

                If Application.CurrentCulture.NumberFormat.NumberDecimalSeparator = "." Then
```

```
    Me.TextBoxFEfemolaire.Text = Me.TextBoxFEfemolaire.Text.Replace(",", ".")
ElseIf Application.CurrentCulture.NumberFormat.NumberDecimalSeparator = "," Then
    Me.TextBoxFEfemolaire.Text = Me.TextBoxFEfemolaire.Text.Replace(".", ",")
End If
FEoxydeRatio = Me.TextBoxFEfemolaire.Text

If Me.ComboBoxChoixFacies.SelectedItem = "Model_10AMP700" Then
    FaciesEnum = enumFacies.facies10AMP700
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_10GRA950" Then
    FaciesEnum = enumFacies.facies10GRA950
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_10SB350" Then
    FaciesEnum = enumFacies.facies10SB350
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_3AMP675" Then
    FaciesEnum = enumFacies.facies3AMP675
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_3GRA800" Then
    FaciesEnum = enumFacies.facies3GRA800
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_4AMP600" Then
    FaciesEnum = enumFacies.facies4AMP600
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_4GRA750" Then
    FaciesEnum = enumFacies.facies4GRA750
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_5AMP670" Then
    FaciesEnum = enumFacies.facies5AMP670
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_5GRA900" Then
    FaciesEnum = enumFacies.facies5GRA900
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_5SV400" Then
    FaciesEnum = enumFacies.facies5SV400
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_6AMP600" Then
    FaciesEnum = enumFacies.facies6AMP600
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_6GRA750" Then
    FaciesEnum = enumFacies.facies6GRA750
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_7AMP500" Then
    FaciesEnum = enumFacies.facies7AMP500
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_7AMP600" Then
    FaciesEnum = enumFacies.facies7AMP600
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_7AMP700" Then
    FaciesEnum = enumFacies.facies7AMP700
ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_9GRA800" Then
```

```
                FaciesEnum = enumFacies.facies9GRA800
            ElseIf Me.ComboBoxChoixFacies.SelectedItem = "Model_9SB450" Then
                FaciesEnum = enumFacies.facies9SB450
            End If

            CalculationSuccessful = False
            SetFacies()
            ouvrirTable()    'Open .txt
            Me.Cursor = Cursors.WaitCursor
            If CalculationSuccessful = True Then : MsgBox("Calculation successful.", MsgBoxStyle.Information)
            End If
            Me.Cursor = Cursors.Default
        End If

    Catch ex As Exception
        Me.Cursor = Cursors.Default
        If ex.InnerException Is Nothing Then
            MsgBox(ex.Message, MsgBoxStyle.Critical)
        Else
            MsgBox(ex.Message & ex.InnerException.Message, MsgBoxStyle.Critical)
        End If
    End Try
End Sub


Private Sub ouvrirTable()

    'Open dialog which requests the user to select a .txt file
    OpenFileDialog1.FileName = ""

    'Set CULTURE info : user must use a DOT for decimals, and NO COMA for large numbers
    CultureInfo = Globalization.CultureInfo.GetCultureInfo("en-US")
    Thread.CurrentThread.CurrentCulture = CultureInfo
    Thread.CurrentThread.CurrentUICulture = CultureInfo

        'OpenFileDialog1.Filter = "Fichier Text (*.txt)|*.txt"
        If Me.OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
GlobalSample.Clear()

Me.Cursor = Cursors.WaitCursor

'Read text file and store its first line (cf. fields' names)
Dim fic As String
fic = OpenFileDialog1.FileName
fic = My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
Dim ObjReader As New System.IO.StreamReader(OpenFileDialog1.FileName)
Dim FirstLine0 As String = ObjReader.ReadLine()
Dim FirstLine() As String = Split(FirstLine0, "      ")

Dim ID As Integer = 1
Dim DataInInput As Boolean = False
Dim MxInInput As Boolean = False
Do While ObjReader.Peek() <> -1

    'Read each line of .txt file
    Dim Line0 As String = ObjReader.ReadLine()
    Dim Line() As String = Split(Line0, "     ")

    'Process first sample
    Dim i As Integer = 0
    Dim DICOtemp As New Dictionary(Of String, Double)
    For Each ElementName As String In FirstLine

        'Solve , and . issues
        Dim ValueString As String
        If Line(i).Contains(",") And Line(i).Contains(".") Then
            ValueString = Line(i).Replace(",", "")
        ElseIf Line(i).Contains(",") Then
            ValueString = Line(i).Replace(",", ".")
        Else
            ValueString = Line(i)
        End If
```

```
                    'Import data / multiply "inferior to detection limit" by -0.5
                    Dim ValueToAdd As Double
                    If ValueString = "" Then
                        ValueToAdd = 0
                    ElseIf CDbl(ValueString) < 0 Then
                        ValueToAdd = CDbl(ValueString) / (-2)
                    Else
                        ValueToAdd = CDbl(ValueString)
                    End If

                    DICOtemp(ElementName) = ValueToAdd
                    i = i + 1
                Next

                'Store chemical data read from .txt in the SAMPLE Dictionary
                Dim Sample As New Dictionary(Of String, Double)
                Sample = SetSample()

                For Each ElementName As String In DICOtemp.Keys
                    If Sample.ContainsKey(ElementName) Then
                        Sample(ElementName) = DICOtemp(ElementName)
                        DataInInput = True
                    End If
                Next
                GlobalSample.Add(ID, Sample)
                ID = ID + 1
            Loop

            'Preform calculation and output data
            If DataInInput = True Then
                PerformCalculation()
                OutputData()
            Else
                MsgBox("No data was extracted from the input file, please check your file", MsgBoxStyle.Critical)
            End If
        End If
End Sub
```

```
Private Sub PerformCalculation()

    GlobalMole.Clear()
    MgFeproblems.Clear()
    NaKproblems.Clear()
    Dim SetExceptDico As Boolean = True

    Dim ID As Integer = 1
    For Each Nb As Integer In GlobalSample.Keys

        Dim mole As New Dictionary(Of String, ValueMx)
        mole = SetMxDIco()

        Dim moleOUT As New Dictionary(Of String, ValueMx)
        Dim Calc As New CONSONORM(mole, GlobalSample(Nb), CO2normatif, FEoxydeRatio, FaciesEnum)

        If SetExceptDico = True Then : SetExceptDico = False
            For Each min As String In mole.Keys
                If mole(min).typeValue = ValueMx.enumTypeValue.silicate Then

                    Dim cation As New Dictionary(Of String, Double)
                    cation = Calc.MineralsDefinition(min)

                    If cation.ContainsKey("Fe2") And Not cation.ContainsKey("Mg") Then : MgFeproblems.Add(min, "Fe")
                    ElseIf cation.ContainsKey("Mg") And Not cation.ContainsKey("Fe2") Then : MgFeproblems.Add(min, "Mg")
                    ElseIf cation.ContainsKey("K") And Not cation.ContainsKey("Na") Then : NaKproblems.Add(min, "K")
                    ElseIf cation.ContainsKey("Na") And Not cation.ContainsKey("K") Then : NaKproblems.Add(min, "Na")
                    End If
                End If
            Next
        End If

        moleOUT = Calc.calculerEchantillon(MgFeproblems, NaKproblems)

        Dim MxToExportTEMP As New List(Of String)
        MxToExportTEMP = Calc.ReturnExport
```

```
        For Each temp In MxToExportTEMP
            If Not MxToExport.Contains(temp) Then : MxToExport.Add(temp)
            End If
        Next

        GlobalMole.Add(Nb, moleOUT)
    Next
End Sub

Private Sub OutputData()
    MxCompleteList.Clear()
    CompleteMxListAlphabeticOrder()

    If MxToExport.Count > 0 Then : CalculationSuccessful = True
    End If

    Dim ColumnEnTete As New Dictionary(Of String, Integer)
    Dim ID2 As Integer = 0
    For Each Mx As String In MxCompleteList
        If MxToExport.Contains(Mx) Then
            ColumnEnTete.Add(Mx, ID2)
            ID2 += 1
        End If
    Next

    Dim EchFInal(ColumnEnTete.Count - 1, 1) As String
    For Each ColName As String In ColumnEnTete.Keys
        Dim Location As Integer = ColumnEnTete(ColName)
        EchFInal(Location, 0) = ColName
    Next

    Dim ArraySize As Integer = 1
    For Each ID As Integer In GlobalMole.Keys
        Dim MxResults As New Dictionary(Of String, ValueMx)
        MxResults = GlobalMole(ID)

        ArraySize = ArraySize + 1
```

```
    ReDim Preserve EchFInal(ColumnEnTete.Count - 1, ArraySize)

    For Each min As String In MxToExport
        If ColumnEnTete.ContainsKey(min) Then
            Dim Location As Integer = ColumnEnTete(min)

            If MxResults(min).weightPct100 > 0 Then
                EchFInal(Location, ArraySize - 1) = CStr(System.Math.Round(MxResults(min).weightPct100, 2))
            ElseIf min = "GOF" Then
                EchFInal(Location, ArraySize - 1) = 0
            ElseIf MxResults(min).typeValue = ValueMx.enumTypeValue.text Then
                EchFInal(Location, ArraySize - 1) = MxResults(min).remark
            Else : EchFInal(Location, ArraySize - 1) = ""
            End If
        End If
    Next
Next


'Write to txt file
SaveFileDialog1.FileName = ""

If Me.SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
    Me.Cursor = Cursors.WaitCursor

    Dim fic As String
    fic = SaveFileDialog1.FileName
    Dim theWriter As New System.IO.StreamWriter(SaveFileDialog1.FileName)

    theWriter.WriteLine("   Calculation performed for model: " & Facies)
    If CO2normatif = True Then : theWriter.WriteLine("    CO2 and H2O estimated normatively, by successive itterations.")
    Else : theWriter.WriteLine("    CO2 and H2O analysed used for this calculation.")
    End If
    theWriter.WriteLine("   Ratio Fe3/(Fe2+Fe3) molar used for calculating Fe-Ti oxides = " & CStr(FEoxydeRatio))
    theWriter.WriteLine(" ")

    Dim LENGTH As Integer = EchFInal.GetLength(1)
```

```
            Dim SIZE As Integer = EchFInal.GetLength(0)

            For i As Integer = 0 To LENGTH - 1
                Dim LINE As String = ""

                For j As Integer = 0 To SIZE - 1
                    LINE = LINE & EchFInal(j, i) & "    "
                Next

                theWriter.WriteLine(LINE)
            Next
            theWriter.Close()
        End If

    End Sub

#Region "SetParameters"

    Private Sub SetFacies()

        If FaciesEnum = enumFacies.facies3AMP675 Then
            Facies = "3AMP675 (3 kbars, 675 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies3GRA800 Then
            Facies = "3GRA800 (3 kbars, 800 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies5GRA900 Then
            Facies = "5GRA900 (5 kbars, 900 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies5AMP670 Then
            Facies = "5AMP670 (5 kbars, 670 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies7AMP700 Then
            Facies = "7AMP700 (7 kbars, 700 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies7AMP600 Then
            Facies = "7AMP600 (7.5 kbars, 600 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies10GRA950 Then
            Facies = "10GRA950 (10 kbars, 950 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies9GRA800 Then
            Facies = "9GRA800 (9 kbars, 800 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies10AMP700 Then
```

```
            Facies = "10AMP700 (10 kbars, 700 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies7AMP500 Then
            Facies = "7AMP500 (7 kbars, 500 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies4AMP600 Then
            Facies = "4AMP600 (4 kbars, 600 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies5SV400 Then
            Facies = "5SV400 (5 kbars, 400 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies9SB450 Then
            Facies = "9SB450 (9 kbars, 450 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies10SB350 Then
            Facies = "10SB350 (10 kbars, 350 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies4GRA750 Then
            Facies = "4GRA750 (4 kbars, 750 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies6GRA750 Then
            Facies = "6GRA750 (6 kbars, 750 Celsius degrees)"
        ElseIf FaciesEnum = enumFacies.facies6AMP600 Then
            Facies = "6AMP600 (6 kbars, 600 Celsius degrees)"
        Else
            Facies = "3AMP675 (3 kbars, 675 Celsius degrees)"
        End If
    End Sub

    Public Enum enumFacies
        facies3AMP675 = 1
        facies3GRA800 = 2
        facies5GRA900 = 3
        facies5AMP670 = 4
        facies7AMP700 = 5
        facies7AMP600 = 6
        facies10GRA950 = 7
        facies9GRA800 = 8
        facies10AMP700 = 9
        facies7AMP500 = 10
        facies4AMP600 = 11
        facies5SV400 = 12
        facies9SB450 = 13
        facies10SB350 = 14
```

```
        facies4GRA750 = 15
        facies6GRA750 = 16
        facies6AMP600 = 17
    End Enum

    Private Function SetSample() As Dictionary(Of String, Double)

        'Format of "sample" dictionary: this dictionary must contain the complete list of elements provided below, with major
elements in wt% and minor elements in ppm.
        'In wt%: SiO2 / Al2O3 / TiO2 / CaO / Fe2O3 / FeO  / K2O / Na2O / P2O5 / MgO / MnO / H2O_PLUS / H2O_MINUS / CO2 / S / LOI
        'In ppm: B / F / Cl / Cr / Ni / Cu / Zn / As / Zr / Mo / Ba / Pb

        Dim Sample As New Dictionary(Of String, Double)

        Sample.Add("SiO2", 0)
        Sample.Add("Al2O3", 0)
        Sample.Add("TiO2", 0)
        Sample.Add("CaO", 0)
        Sample.Add("Fe2O3", 0)
        Sample.Add("FeO", 0)
        Sample.Add("K2O", 0)
        Sample.Add("Na2O", 0)
        Sample.Add("P2O5", 0)
        Sample.Add("MgO", 0)
        Sample.Add("MnO", 0)
        Sample.Add("H2O_PLUS", 0)
        Sample.Add("H2O_MINUS", 0)
        Sample.Add("CO2", 0)
        Sample.Add("S", 0)
        Sample.Add("LOI", 0)
        Sample.Add("B", 0)
        Sample.Add("F", 0)
        Sample.Add("Cl", 0)
        Sample.Add("Cr", 0)
        Sample.Add("Ni", 0)
        Sample.Add("Cu", 0)
        Sample.Add("Zn", 0)
```

```
        Sample.Add("As", 0)
        Sample.Add("Zr", 0)
        Sample.Add("Mo", 0)
        Sample.Add("Ba", 0)
        Sample.Add("Pb", 0)
        Sample.Add("TOTAL", 0)

        Return Sample
    End Function

    Private Function SetMxDIco() As Dictionary(Of String, ValueMx)

        Dim mole As New Dictionary(Of String, ValueMx)

        mole("Si") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Al") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Ti") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Ca") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Fe2") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Fe3") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("K") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Na") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("P") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Mg") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Mn") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("H") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("C") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("S") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("B") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("F") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Cl") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Cr") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Ni") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Cu") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Zn") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("As") = New ValueMx(0, ValueMx.enumTypeValue.element)
        mole("Zr") = New ValueMx(0, ValueMx.enumTypeValue.element)
```

```
mole("Mo") = New ValueMx(0, ValueMx.enumTypeValue.element)
mole("Ba") = New ValueMx(0, ValueMx.enumTypeValue.element)
mole("Pb") = New ValueMx(0, ValueMx.enumTypeValue.element)

mole("apatiteCl") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("apatiteF") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("apatiteH") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("apatite") = New ValueMx(0, ValueMx.enumTypeValue.otherMX, 3.16)
mole("arsenopyrite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 6.19)
mole("barite") = New ValueMx(0, ValueMx.enumTypeValue.otherMX, 4.48)
mole("chalcopyrite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 4.2)
mole("chromite") = New ValueMx(0, ValueMx.enumTypeValue.otherMX, 4.79)
mole("tourmaline") = New ValueMx(0, ValueMx.enumTypeValue.otherMX, 3)
mole("dravite") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("foitite") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("foititeMg") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("galena") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 7.4)
mole("millerite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 5.5)
mole("molybdenite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 5)
mole("pyrite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 5.01)
mole("pyrrhotite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 4.6)
mole("schorl") = New ValueMx(0, ValueMx.enumTypeValue.otherMX)
mole("sphalerite") = New ValueMx(0, ValueMx.enumTypeValue.sulfure, 4.08)
mole("zircon") = New ValueMx(0, ValueMx.enumTypeValue.otherMX, 4.65)

mole("GOF") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("CO2_NORM") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("H2O_NORM") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("H2O_Mineral") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("DENSITY") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("TOTALoxyde") = New ValueMx(0, ValueMx.enumTypeValue.derivated)
mole("TOTALmineral") = New ValueMx(0, ValueMx.enumTypeValue.derivated)

mole("EXCESDEFICIT") = New ValueMx(0, ValueMx.enumTypeValue.text)
mole("COMMENT") = New ValueMx(0, ValueMx.enumTypeValue.text)

mole("ilmenite") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 4.79)
```

```
mole("rutile") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 4.25)
mole("ulvospinel") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 4.8)
mole("magnetite") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 5.15)
mole("hematite") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 5.28)
mole("pseudobrookite") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 4.4)
mole("wustite") = New ValueMx(0, ValueMx.enumTypeValue.oxyde, 6)
mole("HEMilm01") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("HEMilm06") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("HEMilm07") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("PSBfe01") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("PSBfe02") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("PSBfe03") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("PSBfe06") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)
mole("PSBfe07") = New ValueMx(0, ValueMx.enumTypeValue.oxyde)

mole("ankerite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 3.05)
mole("ankerite03") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("ankerite14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("calcite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 2.71)
mole("CCmst05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCmst06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCmst09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCmst10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCmst12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCmst14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsid14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsidB09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
```

```
mole("CCsidmst05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsidmst06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsidmst10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsidmst12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("CCsidmst14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("dolomite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 2.84)
mole("graphite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 2.16)
mole("magnesite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 3)
mole("MSTcc05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("MSTcc06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("MSTcc09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("MSTcc12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("MSTcc14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc09") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDcc14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("siderite") = New ValueMx(0, ValueMx.enumTypeValue.carbonate, 3.96)
mole("SIDmst05") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDmst06") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDmst10") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDmst12") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)
mole("SIDmst14") = New ValueMx(0, ValueMx.enumTypeValue.carbonate)

mole("albite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.63)
mole("actinolite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.25)
mole("aegirine") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.52)
mole("akermanite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.94)
mole("almandine") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 4.32)
mole("amesite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.8)
mole("amesiteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.85)
mole("amesiteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.77)
mole("annite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.34)
mole("anorthite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.75)
mole("anthophyllite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.73)
```

```
mole("anthophylliteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.8)
mole("anthophylliteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.67)
mole("antigorite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.54)
mole("antigoriteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.6)
mole("antigoriteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.5)
mole("biotite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3)
mole("brucite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.39)
mole("carpholite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.9)
mole("chloritoid") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.63)
mole("chloritoidFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.7)
mole("chloritoidMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.57)
mole("clinozoisite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.34)
mole("cordierite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.57)
mole("cordieriteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.67)
mole("cordieriteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.48)
mole("corundum") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 4.05)
mole("CPX") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.47)
mole("CPXgrt") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("daphnite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.2)
mole("daphniteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.4)
mole("daphniteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3)
mole("diaspore") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.4)
mole("diopside") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.26)
mole("kyanite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.61)
mole("enstatite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.2)
mole("epidote") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.45)
mole("fayalite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 4.66)
mole("FeO") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("ferrosilite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.95)
mole("FM") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("forsterite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.22)
mole("FSP04") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP05_A") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP05_B") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP06") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP07") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP08") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
```

```
mole("FSP09") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP11") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("FSP17") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("gehlenite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.98)
mole("glaucophane") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.05)
mole("glaucophaneFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.13)
mole("glaucophaneMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.07)
mole("GROSSss") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("grossular") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.59)
mole("GRThalf") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("GRTpyalm") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("GRTss") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("GRTss05") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("hedenbergite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.68)
mole("hercynite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.95)
mole("jadeite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.3)
mole("K2O") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("kalsilite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.6)
mole("Kfeld") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.56)
mole("kirschsteinite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.43)
mole("lawsonite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.09)
mole("leucite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.48)
mole("lime") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.35)
mole("margarite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.03)
mole("merwinite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.15)
mole("MnO") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("monticellite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.2)
mole("monticelliteSS") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.3)
mole("muscovite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.83)
mole("Na2O") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("NaK") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("nepheline") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.61)
mole("olivine") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.94)
mole("OPX") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.6)
mole("OPXtalc") = New ValueMx(0, ValueMx.enumTypeValue.silicate)
mole("orthose") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.56)
mole("paragonite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.78)
```

```
        mole("pargasite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.12)
        mole("pargasiteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.18)
        mole("pargasiteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.07)
        mole("penninite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.7)
        mole("penniniteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.85)
        mole("penniniteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.6)
        mole("periclase") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.78)
        mole("phlogopite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.83)
        mole("pyrope") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.56)
        mole("pyrophyllite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.81)
        mole("pyroxmangite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.8)
        mole("quartz") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.65)
        mole("rhodonite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.6)
        mole("riebeckite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.4)
        mole("sillimanite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.24)
        mole("spessartine") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 4.18)
        mole("spinel") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.64)
        mole("spinelss") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.8)
        mole("staurolite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.59)
        mole("staurioliteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.64)
        mole("staurioliteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.54)
        mole("sudoite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.65)
        mole("talc") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.75)
        mole("tephroite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 4.25)
        mole("tremoliteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.51)
        mole("tremoliteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.05)
        mole("tschermakite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.17)
        mole("tschermakiteFE") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 3.38)
        mole("tschermakiteMg") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.96)
        mole("WhiteMica") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.8)
        mole("wollastonite") = New ValueMx(0, ValueMx.enumTypeValue.silicate, 2.84)
        Return mole
    End Function

    Private Sub CompleteMxListAlphabeticOrder()

        MxCompleteList.Add("aegirine")
```

```
MxCompleteList.Add("akermanite")
MxCompleteList.Add("albite")
MxCompleteList.Add("almandine")
MxCompleteList.Add("amesiteFE")
MxCompleteList.Add("amesiteMg")
MxCompleteList.Add("annite")
MxCompleteList.Add("anorthite")
MxCompleteList.Add("anthophylliteFE")
MxCompleteList.Add("anthophylliteMg")
MxCompleteList.Add("antigoriteFE")
MxCompleteList.Add("antigoriteMg")
MxCompleteList.Add("brucite")
MxCompleteList.Add("carpholite")
MxCompleteList.Add("chloritoidFE")
MxCompleteList.Add("chloritoidMg")
MxCompleteList.Add("clinozoisite")
MxCompleteList.Add("cordieriteFE")
MxCompleteList.Add("cordieriteMg")
MxCompleteList.Add("corundum")
MxCompleteList.Add("daphniteFE")
MxCompleteList.Add("daphniteMg")
MxCompleteList.Add("diaspore")
MxCompleteList.Add("diopside")
MxCompleteList.Add("kyanite")
MxCompleteList.Add("enstatite")
MxCompleteList.Add("epidote")
MxCompleteList.Add("fayalite")
MxCompleteList.Add("ferrosilite")
MxCompleteList.Add("forsterite")
MxCompleteList.Add("gehlenite")
MxCompleteList.Add("glaucophaneFE")
MxCompleteList.Add("glaucophaneMg")
MxCompleteList.Add("grossular")
MxCompleteList.Add("hedenbergite")
MxCompleteList.Add("hercynite")
MxCompleteList.Add("jadeite")
MxCompleteList.Add("kalsilite")
```

```
MxCompleteList.Add("kirschsteinite")
MxCompleteList.Add("lawsonite")
MxCompleteList.Add("leucite")
MxCompleteList.Add("lime")
MxCompleteList.Add("margarite")
MxCompleteList.Add("merwinite")
MxCompleteList.Add("monticellite")
MxCompleteList.Add("muscovite")
MxCompleteList.Add("nepheline")
MxCompleteList.Add("orthose")
MxCompleteList.Add("paragonite")
MxCompleteList.Add("pargasiteFE")
MxCompleteList.Add("pargasiteMg")
MxCompleteList.Add("penniniteFE")
MxCompleteList.Add("penniniteMg")
MxCompleteList.Add("periclase")
MxCompleteList.Add("phlogopite")
MxCompleteList.Add("pyrope")
MxCompleteList.Add("pyrophyllite")
MxCompleteList.Add("pyroxmangite")
MxCompleteList.Add("quartz")
MxCompleteList.Add("rhodonite")
MxCompleteList.Add("riebeckite")
MxCompleteList.Add("sillimanite")
MxCompleteList.Add("spessartine")
MxCompleteList.Add("spinel")
MxCompleteList.Add("staurolitеFE")
MxCompleteList.Add("staurolitеMg")
MxCompleteList.Add("sudoite")
MxCompleteList.Add("talc")
MxCompleteList.Add("tephroite")
MxCompleteList.Add("tremoliteFE")
MxCompleteList.Add("tremoliteMg")
MxCompleteList.Add("tschermakiteFE")
MxCompleteList.Add("tschermakiteMg")
MxCompleteList.Add("wollastonite")
MxCompleteList.Add("hematite")
```

```
        MxCompleteList.Add("ilmenite")
        MxCompleteList.Add("magnetite")
        MxCompleteList.Add("pseudobrookite")
        MxCompleteList.Add("rutile")
        MxCompleteList.Add("ulvospinel")
        MxCompleteList.Add("wustite")
        MxCompleteList.Add("ankerite")
        MxCompleteList.Add("calcite")
        MxCompleteList.Add("dolomite")
        MxCompleteList.Add("graphite")
        MxCompleteList.Add("magnesite")
        MxCompleteList.Add("siderite")
        MxCompleteList.Add("barite")
        MxCompleteList.Add("chromite")
        MxCompleteList.Add("tourmaline")
        MxCompleteList.Add("zircon")
        MxCompleteList.Add("apatite")
        MxCompleteList.Add("pyrite")
        MxCompleteList.Add("pyrrhotite")
        MxCompleteList.Add("galena")
        MxCompleteList.Add("sphalerite")
        MxCompleteList.Add("millerite")
        MxCompleteList.Add("molybdenite")
        MxCompleteList.Add("chalcopyrite")
        MxCompleteList.Add("arsenopyrite")
        MxCompleteList.Add("GOF")
        MxCompleteList.Add("CO2_NORM")
        MxCompleteList.Add("H2O_NORM")
        MxCompleteList.Add("H2O_Mineral")
        MxCompleteList.Add("TOTALoxyde")
        MxCompleteList.Add("TOTALmineral")
        MxCompleteList.Add("DENSITY")
        MxCompleteList.Add("EXCESDEFICIT")
        MxCompleteList.Add("COMMENT")
    End Sub

#End Region
```

```
    Private Sub TextBoxFEfemolaire_Validating(sender As Object, e As System.ComponentModel.CancelEventArgs) Handles
TextBoxFEfemolaire.Validating
        If Not IsNumeric(Me.TextBoxFEfemolaire.Text) Then
            e.Cancel = True
            ErrorProvider1.SetError(TextBoxFEfemolaire, "Must be numeric between 0 and 1 included (also check the decimal format)")
        ElseIf Me.TextBoxFEfemolaire.Text < 0 Or Me.TextBoxFEfemolaire.Text > 1 Then
            e.Cancel = True
            ErrorProvider1.SetError(TextBoxFEfemolaire, "Must be numeric between 0 and 1 included (also check the decimal format)")
        Else
            ErrorProvider1.SetError(TextBoxFEfemolaire, "")
        End If
    End Sub
End Class
```

3. CONSONORM.vb

```
Imports MathNet.Numerics.LinearAlgebra

Public Class CONSONORM

    Private mole As New Dictionary(Of String, ValueMx)
    Private Sample As New Dictionary(Of String, Double)

    Private CO2normatif As Boolean
    Private GOF As Double = 0
    Private FEoxydeRatio As Double
    Private FaciesEnum As FormMain.enumFacies
    Private TetraLocation As enumTetraedre
    Private SiToGain As Double
    Private AFMfactor As Double = 0

    Private MgMnRatio As Double
    Private FeMnRatio As Double
    Private MgFeRatio As Double
    Private FeMgMnRatio As Double
    Private NaRatio As Double
    Private Kratio As Double
    Private KRatioForWM As Double
    Private Fe2forOxydes As Double

    Private incrementCO2Cycle As Double = 0.05 / 44.02   'You can increase this number to decrease the amount of itterations used to
estimate CO2-normative (cf. to decrease the calculation time requiered)
    Private cycle As Double = 0
    Private MxToExport As New List(Of String)
    Private NaKproblems As New Dictionary(Of String, String)
    Private MgFeproblems As New Dictionary(Of String, String)
    Private MxToForm0() As String
    Private DestroyCARBorNot As Boolean = False
```

```
    'Definition of tetraedres
    Private assemblageCARBONATE01 As New Dictionary(Of Integer, Array)
    Private assemblageCARBONTE02 As New Dictionary(Of Integer, Array)
    Private assemblageOXYDE As New Dictionary(Of Integer, Array)
    Private assemblageSILICATE As New Dictionary(Of Integer, Array)
    Private ASSEMBsilicateAFM As New Dictionary(Of Integer, Array)
    Private assemblageMxAFM As New ArrayList()


    Private Enum enumTetraedre
        TetraI = 1
        TetraII = 2
        TetraIII = 3
    End Enum

    Sub New(ByVal moleIn As Dictionary(Of String, ValueMx), ByVal SampleIn As Dictionary(Of String, Double), ByVal CO2 As Boolean,
ByVal FETIoxyde As Double, ByVal EnFacies As FormMain.enumFacies)

        CO2normatif = CO2
        FaciesEnum = EnFacies
        mole = moleIn
        Sample = SampleIn

        If FETIoxyde = 1 Then : FETIoxyde = 0.999
        End If
        FEoxydeRatio = (1 - FETIoxyde) / FETIoxyde
    End Sub


#Region "MainCalculation"

    Public Function calculerEchantillon(ByVal MgFeproblem As Dictionary(Of String, String), ByVal NaKproblem As Dictionary(Of String,
String))

        MgFeproblems = MgFeproblem
        NaKproblems = NaKproblem
        SelectAssemblageOxydeCarbonate()
```

```
        Try
            Do
                emptyMoles()
                ExtractMoles()
                SetRatios()
                mineralsCalculation()         'Calculate the normative minerals

                'Increase cycle
                If CO2normatif = True Then
                    If mole("H").NBmole * 17 > 5 Then : cycle = cycle + 5
                    ElseIf mole("H").NBmole * 17 > 1 Then : cycle = cycle + 2
                    Else : cycle = cycle + 1
                    End If
                Else : cycle = 0    'Exit loop
                End If
            Loop Until cycle = 0 Or Sample("LOI") = 0 Or (mole("H2O_Mineral").weightPct + mole("CO2_NORM").weightPct) >=
(Sample("LOI") + mole("GOF").weightPct - Sample("S")) Or cycle > 1000

            mineralsCalculationFinal()
            TurnMolarInWeightPercent()    'Turn Moles in Wt% / remove empty records
            ResultMessages()

        Catch ex As Exception
            ' Debug.Print(ex.Message & "Error in calculating the CONSONORM for this sample")
        End Try

        Return mole
    End Function

    Protected Sub mineralsCalculation()

        MakeOtherMx()
        MakeCarbonate()

        Fe2forOxydes = -1
        MakeOxyde(mole("Fe3").NBmole)
```

```
        If mole("wustite").NBmole > 0 Then : mole("Fe2").NBmole = mole("Fe2").NBmole + mole("wustite").NBmole
            mole("wustite").NBmole = 0
        End If

        SetRatios()

        'Make silicates
        '     Get location in tetraedre for silicate calculation / location reseted only a limited amount of times during a
calulation
        If AFMfactor = 0 Or AFMfactor = 5 Or AFMfactor = 10 Or AFMfactor = 20 Or AFMfactor = 30 Or AFMfactor = 50 Or AFMfactor = 100
Or AFMfactor = 250 Or AFMfactor = 500 Or AFMfactor = 900 Then
            LocateTetraedre()
        End If
        AFMfactor = AFMfactor + 1

        MakeSilicate()

        'Get H2O data / adjust GOF from carbonates
        calculateGOF()
        If CO2normatif = True Then
            mole("H").NBmole = (mole("H").NBmole * 9.01 + mole("GOF").weightPct) / 9.01
        End If
        mole("H2O_NORM").NBmole = mole("H").NBmole
        mole("H2O_NORM").weightPct = mole("H").NBmole * 9.01

        'How much water needed to make these silicates?
        mole("H2O_Mineral").NBmole = ELEMENTinSilicatesAutres("H")
        mole("H2O_Mineral").weightPct = mole("H2O_Mineral").NBmole * 9.01
    End Sub

    Private Sub mineralsCalculationFinal()

        'Make more oxydes with Fe left, considering that all FeLeft = FeII (note: FeLeft is only formed when there is Si-deficit) /
or, modify amount of oxydes according to amount of Fe3+ used by silicates
        Dim additionalFE2 As Double = GetFE3inSilicate()

        If additionalFE2 > 0 Or mole("FeO").NBmole > 0 Then
```

```
            Dim FE3remainingForOxydes As Double = mole("Fe3").NBmole - additionalFE2
            If FE3remainingForOxydes < 0 Then
                additionalFE2 = additionalFE2 + FE3remainingForOxydes
                FE3remainingForOxydes = 0.0001
            End If

            Fe2forOxydes = Fe2forOxydes + additionalFE2 + mole("FeO").NBmole
            mole("FeO").NBmole = 0

            For Each Mx As String In mole.Keys
                If mole(Mx).typeValue = ValueMx.enumTypeValue.oxyde Then : mole(Mx).NBmole = 0
                End If
            Next
            MakeOxyde(FE3remainingForOxydes)
        End If
    End Sub

    Public Function ReturnExport()
        Return MxToExport
    End Function

#End Region

#Region "SetParameters"

    Private Sub ExtractMoles()

        If Sample("S") = 0 Then : Sample("S") = 32.07 * ((Sample("Cu") * (2 / 63.546) + Sample("Mo") / 95.96 + Sample("Pb") / 207.2 +
Sample("Zn") / 65.38 + Sample("Ni") / 58.6934 + Sample("As") / 74.921) / 10000)
        End If

        If CO2normatif = True Then
            If Sample("LOI") > 0 Then
                Sample("CO2") = cycle * incrementCO2Cycle * 44.01
                mole("CO2_NORM").NBmole = cycle * incrementCO2Cycle
                mole("CO2_NORM").weightPct = cycle * incrementCO2Cycle * 44.01
```

```
    Else
        Sample("CO2") = 0
        mole("CO2_NORM").NBmole = 0
        mole("CO2_NORM").weightPct = 0
    End If
    Sample("H2O_PLUS") = Sample("LOI") - Sample("S") - Sample("H2O_MINUS") - mole("CO2_NORM").weightPct
End If

Dim TOTAL As Double = 0
If Sample("TOTAL") <= 0 Then
    For Each Elem As String In Sample.Keys
        If Elem = "Mo" Then : TOTAL += Sample(Elem) / 10000
        ElseIf Elem.Contains("O") Or Elem = "S" Then : TOTAL += Sample(Elem)
        Else : TOTAL += Sample(Elem) / 10000
        End If
    Next
    Sample("TOTAL") = TOTAL
Else
    TOTAL = Sample("TOTAL")
End If
mole("TOTALoxyde").NBmole = TOTAL

'Re-calculation to 100% + molar
mole("Si").NBmole = ((Sample("SiO2") * 100 / TOTAL) / 60.09)
mole("Al").NBmole = ((Sample("Al2O3") * 100 / TOTAL) / 50.97)
mole("Ti").NBmole = ((Sample("TiO2") * 100 / TOTAL) / 79.9)
mole("Ca").NBmole = ((Sample("CaO") * 100 / TOTAL) / 56.08)
mole("Fe2").NBmole = ((Sample("FeO") * 100 / TOTAL) / 71.85)
mole("Fe3").NBmole = ((Sample("Fe2O3") * 100 / TOTAL) / 79.85)
mole("K").NBmole = ((Sample("K2O") * 100 / TOTAL) / 47.1)
mole("Na").NBmole = ((Sample("Na2O") * 100 / TOTAL) / 30.991)
mole("P").NBmole = ((Sample("P2O5") * 100 / TOTAL) / 70.975)
mole("Mg").NBmole = ((Sample("MgO") * 100 / TOTAL) / 40.32)
mole("Mn").NBmole = ((Sample("MnO") * 100 / TOTAL) / 70.94)
mole("H").NBmole = ((Sample("H2O_PLUS") * 100 / TOTAL) / 9)
mole("C").NBmole = ((Sample("CO2") * 100 / TOTAL) / 44)
mole("S").NBmole = ((Sample("S") * 100 / TOTAL) / 32.065)
```

```
    mole("B").NBmole = ((0.0001 * Sample("B") * 100 / TOTAL) / 10.811)
    mole("F").NBmole = ((0.0001 * Sample("F") * 100 / TOTAL) / 18.9984032)
    mole("Cl").NBmole = ((0.0001 * Sample("Cl") * 100 / TOTAL) / 35.453)
    mole("Cr").NBmole = ((0.0001 * Sample("Cr") * 100 / TOTAL) / 51.996)
    mole("Ni").NBmole = ((0.0001 * Sample("Ni") * 100 / TOTAL) / 58.6934)
    mole("Cu").NBmole = ((0.0001 * Sample("Cu") * 100 / TOTAL) / 63.546)
    mole("Zn").NBmole = ((0.0001 * Sample("Zn") * 100 / TOTAL) / 65.38)
    mole("As").NBmole = ((0.0001 * Sample("As") * 100 / TOTAL) / 74.921)
    mole("Zr").NBmole = ((0.0001 * Sample("Zr") * 100 / TOTAL) / 91.224)
    mole("Mo").NBmole = ((0.0001 * Sample("Mo") * 100 / TOTAL) / 95.96)
    mole("Ba").NBmole = ((0.0001 * Sample("Ba") * 100 / TOTAL) / 137.327)
    mole("Pb").NBmole = ((0.0001 * Sample("Pb") * 100 / TOTAL) / 207.2)

End Sub

Private Sub SetRatios()

    FeMgMnRatio = UpdateRatios(mole("Mg").NBmole + mole("Fe2").NBmole, mole("Mn").NBmole, 1)
    MgMnRatio = UpdateRatios(mole("Mg").NBmole, mole("Mn").NBmole, 1)
    FeMnRatio = UpdateRatios(mole("Fe2").NBmole, mole("Mn").NBmole, 1)

    MgFeRatio = UpdateRatios(mole("Mg").NBmole, mole("Fe2").NBmole, 0.5)
    NaRatio = UpdateRatios(mole("Na").NBmole, mole("K").NBmole, 0.5)
    Kratio = UpdateRatios(mole("K").NBmole, mole("Na").NBmole, 0.5)
End Sub

Private Function UpdateRatios(ByVal Fact1 As Double, ByVal Fact2 As Double, ByVal Alternative As Double) As Double
    Dim Ratio As Double

    If Fact1 = 0 And Fact2 = 0 Then : Ratio = Alternative
    Else : Ratio = Fact1 / (Fact2 + Fact1)
    End If
    Return Ratio
End Function

Private Sub calculateGOF()
    GOF = 0
```

```
    Dim DicoSulfure As New List(Of String)
    DicoSulfure.Add("pyrite")
    DicoSulfure.Add("pyrrhotite")
    DicoSulfure.Add("chalcopyrite")
    DicoSulfure.Add("arsenopyrite")

    For Each Mx As String In DicoSulfure
        GOF += mole(Mx).NBmole * 1.5 * 15.998           'Calculate GOF from amount of Fe2 used by sulfure
    Next

    GOF += mole("siderite").NBmole * 0.5 * 15.998
    GOF += mole("ankerite").NBmole * (1 - MgFeRatio) * 0.5 * 15.998

    mole("GOF").weightPct = GOF
End Sub

Private Sub SelectAssemblageOxydeCarbonate()

    If FaciesEnum = FormMain.enumFacies.facies3AMP675 Then : Assemblage01()
    ElseIf FaciesEnum = FormMain.enumFacies.facies3GRA800 Then : Assemblage02()
    ElseIf FaciesEnum = FormMain.enumFacies.facies5GRA900 Then : Assemblage03()
    ElseIf FaciesEnum = FormMain.enumFacies.facies5AMP670 Then : Assemblage04()
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP700 Then : Assemblage05()
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 Then : Assemblage06()
    ElseIf FaciesEnum = FormMain.enumFacies.facies10GRA950 Then : Assemblage07()
    ElseIf FaciesEnum = FormMain.enumFacies.facies9GRA800 Then : Assemblage08()
    ElseIf FaciesEnum = FormMain.enumFacies.facies10AMP700 Then : Assemblage09()
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Then : Assemblage10()
    ElseIf FaciesEnum = FormMain.enumFacies.facies4AMP600 Then : Assemblage11()
    ElseIf FaciesEnum = FormMain.enumFacies.facies5SV400 Then : Assemblage12()
    ElseIf FaciesEnum = FormMain.enumFacies.facies9SB450 Then : Assemblage13()
    ElseIf FaciesEnum = FormMain.enumFacies.facies10SB350 Then : Assemblage14()
    ElseIf FaciesEnum = FormMain.enumFacies.facies4GRA750 Then : Assemblage15()
    ElseIf FaciesEnum = FormMain.enumFacies.facies6GRA750 Then : Assemblage16()
    ElseIf FaciesEnum = FormMain.enumFacies.facies6AMP600 Then : Assemblage17()
    End If
```

```
    End Sub

    Private Sub SelectAssemblageSilicate()

        assemblageSILICATE.Clear()

        If FaciesEnum = FormMain.enumFacies.facies3AMP675 Then : assemblageSILICATE = Silicates3AMP675()
        ElseIf FaciesEnum = FormMain.enumFacies.facies3GRA800 Then : assemblageSILICATE = Silicates3GRA800()
        ElseIf FaciesEnum = FormMain.enumFacies.facies5GRA900 Then : assemblageSILICATE = Silicates5GRA900()
        ElseIf FaciesEnum = FormMain.enumFacies.facies5AMP670 Then : assemblageSILICATE = Silicates5AMP670()
        ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP700 Then : assemblageSILICATE = Silicates7AMP700()
        ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 Then : assemblageSILICATE = Silicates7AMP600()
        ElseIf FaciesEnum = FormMain.enumFacies.facies10GRA950 Then : assemblageSILICATE = Silicates10GRA950()
        ElseIf FaciesEnum = FormMain.enumFacies.facies9GRA800 Then : assemblageSILICATE = Silicates9GRA800()
        ElseIf FaciesEnum = FormMain.enumFacies.facies10AMP700 Then : assemblageSILICATE = Silicates10AMP700()
        ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Then : assemblageSILICATE = Silicates7AMP500()
        ElseIf FaciesEnum = FormMain.enumFacies.facies4AMP600 Then : assemblageSILICATE = Silicates4AMP600()
        ElseIf FaciesEnum = FormMain.enumFacies.facies5SV400 Then : assemblageSILICATE = Silicates5SV400()
        ElseIf FaciesEnum = FormMain.enumFacies.facies9SB450 Then : assemblageSILICATE = Silicates9SB450()
        ElseIf FaciesEnum = FormMain.enumFacies.facies10SB350 Then : assemblageSILICATE = Silicates10SB350()
        ElseIf FaciesEnum = FormMain.enumFacies.facies4GRA750 Then : assemblageSILICATE = Silicates4GRA750()
        ElseIf FaciesEnum = FormMain.enumFacies.facies6GRA750 Then : assemblageSILICATE = Silicates6GRA750()
        ElseIf FaciesEnum = FormMain.enumFacies.facies6AMP600 Then : assemblageSILICATE = Silicates6AMP600()
        End If
    End Sub

    Protected Sub emptyMoles()

        For Each n As String In mole.Keys
            mole(n).NBmole = 0
            mole(n).weightPct = 0
        Next
    End Sub
#End Region

#Region "Operations"
```

```
Protected Sub MakeOtherMx(ByVal mineral As String)

    If mineral = "apatite" Then
        Dim apatite As Double = 0
        If mole("Ca").NBmole > mole("P").NBmole * (5 / 3) Then : apatite = mole("P").NBmole * (9 / 3)
        Else : apatite = mole("Ca").NBmole * (9 / 5)
        End If
        mole("Ca").NBmole = mole("Ca").NBmole - (5 / 9) * apatite

        Dim HClF As Double = apatite / 9
        If mole("F").NBmole > HClF Then : mole("apatiteF").NBmole = HClF * 9
        Else : mole("apatiteF").NBmole = mole("F").NBmole * 9
        End If
        HClF = HClF - mole("apatiteF").NBmole
        If mole("Cl").NBmole > HClF Then : mole("apatiteCl").NBmole = HClF * 9
        Else : mole("apatiteCl").NBmole = mole("Cl").NBmole * 9
        End If
        HClF = HClF - mole("apatiteCl").NBmole
        If HClF > 0 Then : mole("apatiteH").NBmole = HClF * 9
        End If

    ElseIf mineral = "chromite" Then
        If mole("Cr").NBmole <= (0.5 * mole("Fe2").NBmole) Then : mole("chromite").NBmole = mole("Cr").NBmole * (3 / 2)
        Else : mole("chromite").NBmole = mole("Fe2").NBmole * 3
        End If
        mole("Fe2").NBmole = mole("Fe2").NBmole - (1 / 3) * mole("chromite").NBmole

    ElseIf mineral = "dravite" Or mineral = "schorl" Or mineral = "foitite" Or mineral = "foititeMg" Then
        Dim Fact1, Fact2, Fact3, Fact4, Fact5 As Double
        Dim F1, F2, F3, F4, F5, F As Double
        Dim moleMX As Double

        If mineral = "schorl" And mole("B").NBmole > 0 Then
            F1 = 13 / 1
            F2 = 13 / 3
            F3 = 999
            F4 = 13 / 6
```

```
    F5 = 13 / 3
ElseIf mineral = "dravite" And mole("B").NBmole > 0 Then
    F1 = 13 / 1
    F2 = 999
    F3 = 13 / 3
    F4 = 13 / 6
    F5 = 13 / 3
ElseIf mineral = "foitite" And mole("B").NBmole > 0 Then
    F1 = 999
    F2 = 12 / 2
    F3 = 999
    F4 = 12 / 7
    F5 = 12 / 3
ElseIf mineral = "foititeMg" And mole("B").NBmole > 0 Then
    F1 = 999
    F2 = 999
    F3 = 12 / 2
    F4 = 12 / 7
    F5 = 12 / 3
End If

Fact1 = mole("Na").NBmole * F1
Fact2 = mole("Fe2").NBmole * F2
Fact3 = mole("Mg").NBmole * F3
Fact4 = mole("Al").NBmole * F4
Fact5 = mole("B").NBmole * F5
If Fact1 < Fact2 And Fact1 < Fact3 And Fact1 < Fact4 And Fact1 < Fact5 Then : moleMX = Fact1
    F = F1
ElseIf Fact2 < Fact1 And Fact2 < Fact3 And Fact2 < Fact4 And Fact2 < Fact5 Then : moleMX = Fact2
    F = F2
ElseIf Fact3 < Fact1 And Fact3 < Fact2 And Fact3 < Fact4 And Fact3 < Fact5 Then : moleMX = Fact3
    F = F3
ElseIf Fact4 < Fact1 And Fact4 < Fact3 And Fact4 < Fact2 And Fact4 < Fact5 Then : moleMX = Fact4
    F = F4
ElseIf Fact5 < Fact1 And Fact5 < Fact3 And Fact5 < Fact2 And Fact5 < Fact4 Then : moleMX = Fact5
    F = F5
End If
```

```
            mole(mineral).NBmole = moleMX

            If F1 < 999 Then : mole("Na").NBmole = mole("Na").NBmole - moleMX / F
            End If
            If F2 < 999 Then : mole("Fe2").NBmole = mole("Fe2").NBmole - moleMX / F
            End If
            If F3 < 999 Then : mole("Mg").NBmole = mole("Mg").NBmole - moleMX / F
            End If
            mole("Al").NBmole = mole("Al").NBmole - moleMX / F
            mole("B").NBmole = mole("B").NBmole - moleMX / F

        ElseIf mineral = "chalcopyrite" Then
            If mole("Cu").NBmole < mole("Fe2").NBmole Then
                mole("chalcopyrite").NBmole = mole("Cu").NBmole * 2
            Else : mole("chalcopyrite").NBmole = mole("Fe2").NBmole * 2
            End If
            mole("Fe2").NBmole = mole("Fe2").NBmole - (1 / 2) * mole("chalcopyrite").NBmole

        ElseIf mineral = "arsenopyrite" Then
            If mole("As").NBmole < mole("Fe2").NBmole Then : mole("arsenopyrite").NBmole = mole("As").NBmole * 2
            Else : mole("arsenopyrite").NBmole = mole("Fe2").NBmole * 2
            End If
            mole("Fe2").NBmole = mole("Fe2").NBmole - (1 / 2) * mole("arsenopyrite").NBmole

        ElseIf mineral = "pyrrhotite" Or mineral = "pyrite" Then
            Dim Sneeded As Double = mole("Pb").NBmole + mole("Zn").NBmole + mole("Ni").NBmole + mole("Mo").NBmole * 0.5 +
mole("Cu").NBmole * 2 + mole("As").NBmole + mole("Ba").NBmole
            If mole("S").NBmole > Sneeded Then
                Dim S As Double = mole("S").NBmole - Sneeded

                If mineral = "pyrite" Then
                    If mole("Fe2").NBmole > (S / 2) Then : mole("pyrite").NBmole = S * 0.5
                    Else : mole("pyrite").NBmole = mole("Fe2").NBmole
                    End If
                ElseIf mineral = "pyrrhotite" Then
                    If mole("Fe2").NBmole > S Then : mole("pyrrhotite").NBmole = S
                    Else : mole("pyrrhotite").NBmole = mole("Fe2").NBmole
```

```
                End If
            End If
            mole("Fe2").NBmole = mole("Fe2").NBmole - mole(mineral).NBmole
        End If
    End If
End Sub

Private Sub MakeOtherMx()

    'Make apatite, chromite, sulfures, tourmaline

    'Make apatite (use P) / only F-apatite and Cl-apatite are possible. If Cl or F cannot be limiting factors (so apatite will be
made even if Cl and F are not analysed)
    If mole("P").NBmole > 0 And mole("Ca").NBmole > 0 Then : MakeOtherMx("apatite")
    End If

    'Make chromite (use Cr)
    If mole("Cr").NBmole > 0 Then : MakeOtherMx("chromite")
    End If

    'Make zircon (use Zr)
    If mole("Zr").NBmole > 0 Then : mole("zircon").NBmole = mole("Zr").NBmole
    End If

    'Make tourmaline (use B / with simplified formula of Schorl)
    If mole("B").NBmole > 0 Then
        If mole("Na").NBmole > (mole("Al").NBmole) / 6 Then
            If mole("Mg").NBmole > (mole("Fe2").NBmole) / 6 Then
                MakeOtherMx("dravite")
                MakeOtherMx("schorl")
                MakeOtherMx("foititeMg")
                MakeOtherMx("Foitite")
            Else
                MakeOtherMx("schorl")
                MakeOtherMx("dravite")
                MakeOtherMx("Foitite")
                MakeOtherMx("foititeMg")
```

```
                End If
            Else
                If mole("Mg").NBmole > (mole("Fe2").NBmole) / 6 Then
                    MakeOtherMx("foititeMg")
                    MakeOtherMx("Foitite")
                    MakeOtherMx("dravite")
                    MakeOtherMx("schorl")
                Else
                    MakeOtherMx("Foitite")
                    MakeOtherMx("foititeMg")
                    MakeOtherMx("schorl")
                    MakeOtherMx("dravite")
                End If
            End If
        End If

        'Make sulfures and sulfate (if S is not analysed, sulfures are made from metals)
        If mole("Pb").NBmole > 0 Then : mole("galena").NBmole = mole("Pb").NBmole
        End If
        If mole("Zn").NBmole > 0 Then : mole("sphalerite").NBmole = mole("Zn").NBmole
        End If
        If mole("Ni").NBmole > 0 Then : mole("millerite").NBmole = mole("Ni").NBmole
        End If
        If mole("Mo").NBmole > 0 Then : mole("molybdenite").NBmole = mole("Mo").NBmole
        End If
        If mole("Cu").NBmole > 0 Then : MakeOtherMx("chalcopyrite")
        End If
        If mole("As").NBmole > 0 Then : MakeOtherMx("arsenopyrite")
        End If
        If mole("Ba").NBmole > 0 Then : mole("barite").NBmole = mole("Ba").NBmole
        End If

        'PY and PO: can only be made if S has been analysed
        If mole("S").NBmole > 0 Then
            If FaciesEnum = FormMain.enumFacies.facies3GRA800 Or FaciesEnum = FormMain.enumFacies.facies5GRA900 Or FaciesEnum =
FormMain.enumFacies.facies10GRA950 Or FaciesEnum = FormMain.enumFacies.facies4GRA750 Or FaciesEnum =
FormMain.enumFacies.facies6GRA750 Then
```

```
            MakeOtherMx("pyrrhotite")
        Else : MakeOtherMx("pyrite")
        End If
    End If
End Sub

Protected Sub ResultMessages()

    Dim message As String = ""
    Dim m2 As String = "Deficit:   "
    Dim m1 As String = "Exces:   "

    Dim ExcesData As Double = 0

    If mole("K2O").NBmole > 0 Then
        ExcesData = mole("K2O").NBmole
        mole("K2O").NBmole = 0
    End If
    If mole("Na2O").NBmole > 0 Then
        ExcesData = ExcesData + mole("Na2O").NBmole
        mole("Na2O").NBmole = 0
    End If

    If ExcesData > 0 Then
        m1 = "Exces: K2O+Na2O = " & ExcesData & " (wt%)"
    End If


    Dim H2OwtSilicate As Double = mole("H2O_Mineral").weightPct100
    Dim H2Owt As Double = mole("H2O_NORM").weightPct100
    Dim incertitude As Double = 0.1  'H2O in wt%

    If CO2normatif = True Then
        If Sample("LOI") <> 0 Then
            If H2OwtSilicate <= (H2Owt + incertitude) And H2OwtSilicate >= (H2Owt - incertitude) Then
                message = "No disagreement between model chosen and H2O measured."
            ElseIf H2OwtSilicate > (H2Owt + incertitude) Then
```

```
                message = "Please use a higher temperature facies."
                m2 = " Deficit: H2O = " & (H2Owt + incertitude) - H2OwtSilicate & " (wt%)"
            ElseIf H2OwtSilicate < (H2Owt - incertitude) Then
                message = "Please use a lower temperature facies."
                m2 = ", H2O = " & (H2Owt + incertitude) - H2OwtSilicate & " (wt%)"
            End If
        Else : message = "Please measure H2O, CO2 and/or the PAF to obtain more accurate results."
        End If

    ElseIf CO2normatif = False Then
        If mole("H").NBmole <> 0 Or mole("C").NBmole <> 0 Then
            If H2OwtSilicate <= (H2Owt + incertitude) And H2OwtSilicate >= (H2Owt - incertitude) Then
                message = "No disagreement between model chosen and H2O measured."
            ElseIf H2OwtSilicate > (H2Owt + incertitude) Then
                message = "Please use a higher temperature model."
                m2 = " Deficit: H2O = " & (H2Owt + incertitude) - H2OwtSilicate & " (wt%)"
            ElseIf H2OwtSilicate < (H2Owt - incertitude) Then
                message = "Please use a lower temperature model."
                m2 = ", H2O = " & (H2Owt + incertitude) - H2OwtSilicate & " (wt%)"
            End If
        Else : message = "Please measure H2O, CO2 and/or the PAF to obtain more accurate results."
        End If
    End If

    mole("EXCESDEFICIT").remark = m1 + m2
    mole("COMMENT").remark = message
End Sub

Protected Sub TurnMolarInWeightPercent()

    'Turn Molar Weight (MM) in proportions (%) of normative minerals

    'WT% for minerals
    Dim SumMineralsWT As Double = 0
    Dim SumMineralsMOLE As Double = 0

    For Each MinName As String In mole.Keys
```

```
            Dim FactorWT As Double = SetWeightPctMinerals(MinName)
            If Not Double.IsNaN(FactorWT) Then
                If FactorWT > 0 Then
                    mole(MinName).weightPct = mole(MinName).NBmole * FactorWT

                    If mole(MinName).weightPct < 0.001 Or Double.IsNaN(mole(MinName).weightPct) Then
                        ' mole(MinName).weightPct = 0
                    Else : SumMineralsWT += mole(MinName).weightPct
                    End If
                End If
            End If
        Next

        'Re-calculate to 100%
        For Each MinName As String In mole.Keys
            mole(MinName).weightPct100 = (mole(MinName).weightPct / SumMineralsWT) * 100
        Next

        mole("apatite").weightPct100 = mole("apatiteH").weightPct100 + mole("apatiteF").weightPct100 + mole("apatiteCl").weightPct100
        mole("apatiteH").weightPct100 = 0
        mole("apatiteF").weightPct100 = 0
        mole("apatiteCl").weightPct100 = 0

        mole("tourmaline").weightPct100 = mole("schorl").weightPct100 + mole("dravite").weightPct100 + mole("foitite").weightPct100 +
mole("foititeMg").weightPct100
        mole("schorl").weightPct100 = 0
        mole("dravite").weightPct100 = 0
        mole("foitite").weightPct100 = 0
        mole("foititeMg").weightPct100 = 0



        'WT% for derived data
        Dim MultiplicatingFactor As Double = 0
        For Each Name As String In mole.Keys
            If mole(Name).typeValue = ValueMx.enumTypeValue.derivated Then
                If Name = "CO2_NORM" Then : MultiplicatingFactor = 44.01
```

```
                ElseIf Name = "H2O_NORM" Or Name = "H2O_Mineral" Then : MultiplicatingFactor = 9.01
                ElseIf Name = "K2O" Then : MultiplicatingFactor = 47.1
                ElseIf Name = "Na2O" Then : MultiplicatingFactor = 31
                Else : MultiplicatingFactor = 1
                End If
                mole(Name).weightPct100 = MultiplicatingFactor * mole(Name).NBmole
            End If
        Next
        mole("TOTALmineral").weightPct100 = SumMineralsWT

        'Calculate density
        Dim DensityData As Double = 0
        For Each MinName As String In mole.Keys
            If mole(MinName).density > 0 And mole(MinName).weightPct100 > 0 Then
                DensityData += mole(MinName).weightPct100 / mole(MinName).density
            End If
        Next
        DensityData = DensityData / 100     'Volume
        mole("DENSITY").weightPct100 = 1 / DensityData     'Density


        'Keep track of data to be exported
        For Each Min In mole.Keys
            If Not MxToExport.Contains(Min) Then
                If mole(Min).weightPct100 > 0.001 Or mole(Min).typeValue = ValueMx.enumTypeValue.derivated Or mole(Min).typeValue =
ValueMx.enumTypeValue.text Then
                    MxToExport.Add(Min)
                End If
            End If
        Next
    End Sub

    Private Function SetWeightPctMinerals(ByVal MxName As String) As Double

        Dim WTpct As Double = 0
        Dim SUMconstituant As Double = 0
        Dim cations As New Dictionary(Of String, Double)
```

```
        cations = MineralsDefinition(MxName)
        For Each Constituant As String In cations.Keys
            If Constituant = "Al" Then : WTpct += cations(Constituant) * 26.98
            ElseIf Constituant = "As" Then : WTpct += cations(Constituant) * 74.92
            ElseIf Constituant = "B" Then : WTpct += cations(Constituant) * 10.81
            ElseIf Constituant = "Ba" Then : WTpct += cations(Constituant) * 137.33
            ElseIf Constituant = "C" Then : WTpct += cations(Constituant) * 12.1
            ElseIf Constituant = "Ca" Then : WTpct += cations(Constituant) * 40.08
            ElseIf Constituant = "Cl" Then : WTpct += cations(Constituant) * 35.45
            ElseIf Constituant = "Cr" Then : WTpct += cations(Constituant) * 52
            ElseIf Constituant = "Cu" Then : WTpct += cations(Constituant) * 65.55
            ElseIf Constituant = "F" Then : WTpct += cations(Constituant) * 19
            ElseIf Constituant = "Fe2" Then : WTpct += cations(Constituant) * 55.85
            ElseIf Constituant = "Fe3" Then : WTpct += cations(Constituant) * 55.85
            ElseIf Constituant = "H" Then : WTpct += cations(Constituant) * 1.01
            ElseIf Constituant = "K" Then : WTpct += cations(Constituant) * 39.09
            ElseIf Constituant = "Mg" Then : WTpct += cations(Constituant) * 24.31
            ElseIf Constituant = "Mn" Then : WTpct += cations(Constituant) * 54.94
            ElseIf Constituant = "Mo" Then : WTpct += cations(Constituant) * 95.96
            ElseIf Constituant = "Na" Then : WTpct += cations(Constituant) * 22.99
            ElseIf Constituant = "Ni" Then : WTpct += cations(Constituant) * 58.69
            ElseIf Constituant = "O" Then : WTpct += cations(Constituant) * 16
            ElseIf Constituant = "P" Then : WTpct += cations(Constituant) * 30.97
            ElseIf Constituant = "Pb" Then : WTpct += cations(Constituant) * 207.2
            ElseIf Constituant = "S" Then : WTpct += cations(Constituant) * 32.06
            ElseIf Constituant = "Si" Then : WTpct += cations(Constituant) * 28.09
            ElseIf Constituant = "Ti" Then : WTpct += cations(Constituant) * 47.9
            ElseIf Constituant = "Zn" Then : WTpct += cations(Constituant) * 65.38
            ElseIf Constituant = "Zr" Then : WTpct += cations(Constituant) * 91.22
            End If

            If Constituant <> "KNa" And Constituant <> "MgFe" And Constituant <> "ClF" Then
                If Constituant <> "Si" And Constituant <> "H" And Constituant <> "O" And Constituant <> "C" And Constituant <> "S"
And Constituant <> "F" And Constituant <> "Cl" Then
                    SUMconstituant += cations(Constituant)
                End If
            End If
```

```
        Next

        If SUMconstituant > 0 Then
            WTpct = WTpct / SUMconstituant
        End If

        Return WTpct
    End Function

    Public Function ELEMENTinSilicatesAutres(ByVal Element As String) As Double

        Dim mineralsElement As Double = 0

        For Each MxSi As String In mole.Keys
            If mole(MxSi).typeValue = ValueMx.enumTypeValue.silicate Or mole(MxSi).typeValue = ValueMx.enumTypeValue.otherMX Then

                Dim cation As New Dictionary(Of String, Double)
                cation = SetCationData(MxSi)

                If cation("SUM") > 0 And mole(MxSi).NBmole > 0 Then
                    If cation.ContainsKey(Element) Then
                        mineralsElement += mole(MxSi).NBmole * (cation(Element) / cation("SUM"))
                    End If
                End If
            End If
        Next
        Return mineralsElement
    End Function

    Private Function GetFE3inSilicate() As Double

        'Mineral containing Fe3+
        Dim additionalFe2forOxydes As Double = (1 / 5) * mole("epidote").NBmole + (3 / 9) * mole("tschermakiteFE").NBmole + (1 / 2) *
mole("aegirine").NBmole + (2 / 15) * mole("riebeckite").NBmole

        Return additionalFe2forOxydes
    End Function
```

```
Public Sub ManageSS()

    'Manage GRENAT - Fe-Mn-Mg
    If mole("GRTpyalm").NBmole > 0 Then
        If FeMgMnRatio > 0.9 Then
            mole("pyrope").NBmole = MgFeRatio * mole("GRTpyalm").NBmole
            mole("almandine").NBmole = (1 - MgFeRatio) * mole("GRTpyalm").NBmole
        Else
            Dim PropSPESS As Double = mole("GRTpyalm").NBmole * (1 - FeMgMnRatio)
            Dim PropGRTleft As Double = mole("GRTpyalm").NBmole - PropSPESS

            mole("pyrope").NBmole = MgFeRatio * PropGRTleft
            mole("almandine").NBmole = (1 - MgFeRatio) * PropGRTleft
            mole("spessartine").NBmole = PropSPESS
        End If
        mole("GRTpyalm").NBmole = 0
    End If


    'Manage Mg-Fe SS and K-Na SS
    For Each MinName As String In mole.Keys
        If mole(MinName).NBmole > 0 And mole(MinName).typeValue = ValueMx.enumTypeValue.silicate Then

            Dim cation As New Dictionary(Of String, Double)
            cation = MineralsDefinition(MinName)

            If cation.ContainsKey("KNa") And mole(MinName).NBmole > 0 Then
                If MinName = "WhiteMica" Then
                    If KRatioForWM = 0 Then : KRatioForWM = Kratio
                    End If
                    mole("paragonite").NBmole = (1 - KRatioForWM) * mole("WhiteMica").NBmole
                    mole("muscovite").NBmole = KRatioForWM * mole("WhiteMica").NBmole
                    mole("WhiteMica").NBmole = 0
                ElseIf MinName = "Kfeld" Then
                    mole("albite").NBmole = (1 - Kratio) * mole("Kfeld").NBmole
                    mole("orthose").NBmole = Kratio * mole("Kfeld").NBmole
```

```
            mole("Kfeld").NBmole = 0
        ElseIf MinName = "NaK" Then
            mole("Na2O").NBmole = (1 - Kratio) * mole("NaK").NBmole
            mole("K2O").NBmole = Kratio * mole("NaK").NBmole
            mole("NaK").NBmole = 0
        End If

    ElseIf cation.ContainsKey("MgFe") And mole(MinName).NBmole > 0 Then
        If MinName = "OPX" Then
            mole("enstatite").NBmole = MgFeRatio * mole("OPX").NBmole
            mole("ferrosilite").NBmole = (1 - MgFeRatio) * mole("OPX").NBmole
            mole("OPX").NBmole = 0
        ElseIf MinName = "biotite" Then
            mole("annite").NBmole = MgFeRatio * mole("biotite").NBmole
            mole("phlogopite").NBmole = (1 - MgFeRatio) * mole("biotite").NBmole
            mole("biotite").NBmole = 0
        ElseIf MinName = "CPX" Then
            mole("diopside").NBmole = MgFeRatio * mole("CPX").NBmole
            mole("hedenbergite").NBmole = (1 - MgFeRatio) * mole("CPX").NBmole
            mole("CPX").NBmole = 0
        ElseIf MinName = "olivine" Then
            mole("forsterite").NBmole = MgFeRatio * mole("olivine").NBmole
            mole("fayalite").NBmole = (1 - MgFeRatio) * mole("olivine").NBmole
            mole("olivine").NBmole = 0
        ElseIf MinName = "actinolite" Then
            mole("tremoliteMg").NBmole = MgFeRatio * mole("actinolite").NBmole
            mole("tremoliteFE").NBmole = (1 - MgFeRatio) * mole("actinolite").NBmole
            mole("actinolite").NBmole = 0
        ElseIf MinName = "spinelss" Then
            mole("spinel").NBmole = MgFeRatio * mole("spinelss").NBmole
            mole("hercynite").NBmole = (1 - MgFeRatio) * mole("spinelss").NBmole
            mole("spinelss").NBmole = 0
        ElseIf MinName = "monticelliteSS" Then
            mole("monticellite").NBmole = MgFeRatio * mole("spinelss").NBmole
            mole("kirschsteinite").NBmole = (1 - MgFeRatio) * mole("spinelss").NBmole
            mole("monticelliteSS").NBmole = 0
```

```
                    Else
                        Dim MinNameMg As String = MinName & "Mg"
                        Dim MinNameFE As String = MinName & "FE"
                        mole(MinNameMg).NBmole = MgFeRatio * mole(MinName).NBmole
                        mole(MinNameFE).NBmole = (1 - MgFeRatio) * mole(MinName).NBmole
                        mole(MinName).NBmole = 0
                    End If
                End If
            End If
        Next
    End Sub

#End Region


#Region "CalculateWithTetraedre"

    Private Sub mxToBeFormed(ByVal M1 As Double, ByVal M2 As Double, ByVal M3 As Double, ByVal M4 As Double, ByVal WhichMethod As
String)

        'Set MAIN tetraedre
        Dim taille As Double = 1
        Dim MainTetra As New Dictionary(Of Integer, Dictionary(Of String, Double))
        Dim MainPt1 As New Dictionary(Of String, Double)
        Dim MainPt2 As New Dictionary(Of String, Double)
        Dim MainPt3 As New Dictionary(Of String, Double)
        Dim MainPt4 As New Dictionary(Of String, Double)
        MainPt1 = NewPointCartesian(taille / 2, taille / 2 * System.Math.Tan(Math.PI / 6), taille * Math.Cos(Math.PI / 6))
        MainPt2 = NewPointCartesian(0, 0, 0)
        MainPt3 = NewPointCartesian(taille, 0, 0)
        MainPt4 = NewPointCartesian(taille / 2, taille * Math.Sin(Math.PI / 3), 0)
        MainTetra.Add(1, MainPt1)
        MainTetra.Add(2, MainPt2)
        MainTetra.Add(3, MainPt3)
        MainTetra.Add(4, MainPt4)
        calculerCoordonneesBarycentriques(MainPt1, MainTetra)
        calculerCoordonneesBarycentriques(MainPt2, MainTetra)
```

```
calculerCoordonneesBarycentriques(MainPt3, MainTetra)
calculerCoordonneesBarycentriques(MainPt4, MainTetra)

'Test individual points in SMALL tetraedres
If M1 <= 0.0001 Then : M1 = 0.0001
End If
If M2 <= 0.0001 Then : M2 = 0.0001
End If
If M3 <= 0.0001 Then : M3 = 0.0001
End If
If M4 <= 0.0001 Then : M4 = 0.0001
End If
Dim SumBary As Double = M1 + M2 + M3 + M4

Dim tetraedreChoosen2 As String = Nothing
Dim mineralsToTestNew As New Dictionary(Of Integer, Array)

If WhichMethod = "CarbonateMethode1" Then : mineralsToTestNew = assemblageCARBONATE01
ElseIf WhichMethod = "CarbonateMethode2" Then : mineralsToTestNew = assemblageCARBONTE02
ElseIf WhichMethod = "TetraedreAFM" Then : mineralsToTestNew = ASSEMBsilicateAFM
ElseIf WhichMethod = "Oxyde" Then : mineralsToTestNew = assemblageOXYDE
ElseIf WhichMethod = "Silicate" Then : mineralsToTestNew = assemblageSILICATE
End If

'Coordonnees ACFK d'un point a tester, comme un echantillon par exemple
Dim pt As New Dictionary(Of String, Double)
pt = NewPointBarycentric(M1 / SumBary, M2 / SumBary, M3 / SumBary, M4 / SumBary)
calculerCoordonneesCartesiennes(pt, MainTetra)

'Itterates through assemblages and select minerals to be formed
Dim MxToFormDelete() As String
MxToForm0 = MxToFormDelete
DestroyCARBorNot = False

For Each i As Integer In mineralsToTestNew.Keys

    Try
```

```
                ' Built small tetraedres, one by one, and test pt each time
                Dim IndividualAssemblage() As String

                Dim cationMx1 As New Dictionary(Of String, Double)
                Dim cationMx2 As New Dictionary(Of String, Double)
                Dim cationMx3 As New Dictionary(Of String, Double)
                Dim cationMx4 As New Dictionary(Of String, Double)

                If WhichMethod <> "CarbonateMethode2" And WhichMethod <> "TetraedreAFM" Then
                    IndividualAssemblage = mineralsToTestNew(i)
                    cationMx1 = SetCationDataForTetraedre(IndividualAssemblage(0), WhichMethod)
                    cationMx2 = SetCationDataForTetraedre(IndividualAssemblage(1), WhichMethod)
                    cationMx3 = SetCationDataForTetraedre(IndividualAssemblage(2), WhichMethod)

                    If WhichMethod = "Silicate" Then
                        cationMx4 = SetCationDataForTetraedre(IndividualAssemblage(3), WhichMethod)
                    End If
                End If

                Dim SmallTetraedre As New Dictionary(Of Integer, Dictionary(Of String, Double))
                Dim pt1 As New Dictionary(Of String, Double)
                Dim pt2 As New Dictionary(Of String, Double)
                Dim pt3 As New Dictionary(Of String, Double)
                Dim pt4 As New Dictionary(Of String, Double)

                If WhichMethod = "CarbonateMethode1" Then
                    pt1 = NewPointBarycentric(cationMx1("Ca") / cationMx1("SUM"), 0, cationMx1("Mg") / cationMx1("SUM"),
cationMx1("Fe2") / cationMx1("SUM"))
                    pt2 = NewPointBarycentric(cationMx2("Ca") / cationMx2("SUM"), 0, cationMx2("Mg") / cationMx2("SUM"),
cationMx2("Fe2") / cationMx2("SUM"))
                    pt3 = NewPointBarycentric(cationMx3("Ca") / cationMx3("SUM"), 0, cationMx3("Mg") / cationMx3("SUM"),
cationMx3("Fe2") / cationMx3("SUM"))
                    pt4 = NewPointBarycentric(0, 1, 0, 0)

                ElseIf WhichMethod = "CarbonateMethode2" Then
                    Dim cc(,) As Double
                    cc = mineralsToTestNew(i)
```

```
                    pt1 = NewPointBarycentric(cc(0, 0), cc(0, 1), cc(0, 2), cc(0, 3))
                    pt2 = NewPointBarycentric(cc(1, 0), cc(1, 1), cc(1, 2), cc(1, 3))
                    pt3 = NewPointBarycentric(cc(2, 0), cc(2, 1), cc(2, 2), cc(2, 3))
                    pt4 = NewPointBarycentric(cc(3, 0), cc(3, 1), cc(3, 2), cc(3, 3))

                ElseIf WhichMethod = "TetraedreAFM" Then
                    Dim cc(,) As Double
                    cc = ASSEMBsilicateAFM(i)
                    pt1 = NewPointBarycentric(cc(0, 0), cc(0, 1), cc(0, 2), cc(0, 3))
                    pt2 = NewPointBarycentric(cc(1, 0), cc(1, 1), cc(1, 2), cc(1, 3))
                    pt3 = NewPointBarycentric(cc(2, 0), cc(2, 1), cc(2, 2), cc(2, 3))
                    pt4 = NewPointBarycentric(cc(3, 0), cc(3, 1), cc(3, 2), cc(3, 3))

                ElseIf WhichMethod = "Oxyde" Then
                    pt1 = NewPointBarycentric(cationMx1("Ti") / cationMx1("SUM"), cationMx1("Fe2") / cationMx1("SUM"),
cationMx1("Fe3") / cationMx1("SUM"), 0)
                    pt2 = NewPointBarycentric(cationMx2("Ti") / cationMx2("SUM"), cationMx2("Fe2") / cationMx2("SUM"),
cationMx2("Fe3") / cationMx2("SUM"), 0)
                    pt3 = NewPointBarycentric(cationMx3("Ti") / cationMx3("SUM"), cationMx3("Fe2") / cationMx3("SUM"),
cationMx3("Fe3") / cationMx3("SUM"), 0)
                    pt4 = NewPointBarycentric(0, 0, 0, 1)

                ElseIf WhichMethod = "Silicate" Then
                    pt1 = NewPointBarycentric(cationMx1("Al") / cationMx1("SUM"), (cationMx1("K") + cationMx1("Na") +
cationMx1("KNa")) / cationMx1("SUM"), (cationMx1("Mg") + cationMx1("Fe2") + cationMx1("MgFe")) / cationMx1("SUM"), cationMx1("Ca") /
cationMx1("SUM"))
                    pt2 = NewPointBarycentric(cationMx2("Al") / cationMx2("SUM"), (cationMx2("K") + cationMx2("Na") +
cationMx2("KNa")) / cationMx2("SUM"), (cationMx2("Mg") + cationMx2("Fe2") + cationMx2("MgFe")) / cationMx2("SUM"), cationMx2("Ca") /
cationMx2("SUM"))
                    pt3 = NewPointBarycentric(cationMx3("Al") / cationMx3("SUM"), (cationMx3("K") + cationMx3("Na") +
cationMx3("KNa")) / cationMx3("SUM"), (cationMx3("Mg") + cationMx3("Fe2") + cationMx3("MgFe")) / cationMx3("SUM"), cationMx3("Ca") /
cationMx3("SUM"))
                    pt4 = NewPointBarycentric(cationMx4("Al") / cationMx4("SUM"), (cationMx4("K") + cationMx4("Na") +
cationMx4("KNa")) / cationMx4("SUM"), (cationMx4("Mg") + cationMx4("Fe2") + cationMx4("MgFe")) / cationMx4("SUM"), cationMx4("Ca") /
cationMx4("SUM"))
                End If
```

```
                calculerCoordonneesCartesiennes(pt1, MainTetra)
                calculerCoordonneesCartesiennes(pt2, MainTetra)
                calculerCoordonneesCartesiennes(pt3, MainTetra)
                calculerCoordonneesCartesiennes(pt4, MainTetra)
                SmallTetraedre.Add(1, pt1)
                SmallTetraedre.Add(2, pt2)
                SmallTetraedre.Add(3, pt3)
                SmallTetraedre.Add(4, pt4)


                ' TEST pt
                calculerCoordonneesBarycentriques(pt, SmallTetraedre)
                If pt("coordHaut") < -0.00001 Or pt("coordArriere") < -0.00001 Or pt("coordBasGauche") < -0.00001 Or
pt("coordBasDroite") < -0.00001 Then
                Else
                    If WhichMethod = "CarbonateMethode2" Then
                        DestroyCARBorNot = True
                    ElseIf WhichMethod = "TetraedreAFM" Then
                        tetraedreChoosen2 = assemblageMxAFM(i)
                    Else
                        Dim MxToForm() As String = mineralsToTestNew(i)
                        MxToForm0 = MxToForm
                        DestroyCARBorNot = False
                    End If
                End If

            Catch ex2 As Exception
            End Try
        Next

        If WhichMethod = "TetraedreAFM" Then
            If tetraedreChoosen2 = Nothing Then : TetraLocation = enumTetraedre.TetraIII
            ElseIf tetraedreChoosen2 = "tetraedre01" Then : TetraLocation = enumTetraedre.TetraI
            ElseIf tetraedreChoosen2 = "tetraedre02" Then : TetraLocation = enumTetraedre.TetraII
            End If
        End If
    End Sub
```

```
    Private Function SetCationData(ByVal MxName As String) As Dictionary(Of String, Double)

        Dim cationMx As New Dictionary(Of String, Double)
        cationMx = MineralsDefinition(MxName)

        Dim SUM As Double = 0

        For Each Constitutants As String In cationMx.Keys
            If Constitutants <> "Si" And Constitutants <> "H" And Constitutants <> "O" And Constitutants <> "C" And Constitutants <>
"S" Then
                SUM += cationMx(Constitutants)
            End If
        Next

        cationMx.Add("SUM", SUM)
        Return cationMx
    End Function

    Private Function SetCationDataForTetraedre(ByVal MxName As String, ByVal WhichMethod As String) As Dictionary(Of String, Double)

        Dim cationMx As New Dictionary(Of String, Double)
        cationMx = MineralsDefinition(MxName)
        Dim SUM As Double = 0

        Dim ListMissingCations() As String
        If WhichMethod = "Oxyde" Then : ListMissingCations = {"Ti", "Fe2", "Fe3", "SUM"}
        ElseIf WhichMethod = "Silicate" Then : ListMissingCations = {"Al", "K", "Na", "Mg", "Fe2", "Ca", "MgFe", "KNa", "Si", "SUM"}
        ElseIf WhichMethod = "CarbonateMethode1" Then : ListMissingCations = {"Ca", "Fe2", "Mg", "SUM"}
        Else : ListMissingCations = {"SUM"}
        End If

        For Each catMiss As String In ListMissingCations
            If Not cationMx.ContainsKey(catMiss) Then : cationMx.Add(catMiss, 0)
            End If
        Next
```

```
        For Each Constitutants As String In cationMx.Keys
            If Constitutants <> "Si" And Constitutants <> "H" And Constitutants <> "O" And Constitutants <> "C" And Constitutants <>
"S" Then
                SUM += cationMx(Constitutants)
            End If
        Next

        cationMx("SUM") = SUM
        Return cationMx
    End Function

    Protected Sub inverseMatriceGetMxMole(ByVal M1 As Double, ByVal M2 As Double, ByVal M3 As Double, ByVal M4 As Double, ByVal
WhichMethod As String, Optional ByVal MxToForm2() As String = Nothing)

        MxToForm0 = Nothing

        If MxToForm2 Is Nothing Then : mxToBeFormed(M1, M2, M3, M4, WhichMethod)
        Else : MxToForm0 = MxToForm2
        End If

        If DestroyCARBorNot = False And MxToForm0 IsNot Nothing And WhichMethod <> "CarbonateMethode2" Then
            'Method from K.L. Pruseth - published in Computers and Geosciences 35 (2009) 1785-1788

            Dim Ndata As Integer = 3
            If WhichMethod = "CarbonateMethode1" Or WhichMethod = "Oxyde" Then
                Ndata = 3
            ElseIf WhichMethod = "Silicate" Then
                Ndata = 4
            End If

            Dim matrixMx As New Matrix(Ndata, Ndata)
            Dim matrixMxProp As New Matrix(Ndata, 1)
            Dim vect(0 To Ndata - 1) As Double
            Dim matB As New Matrix(Ndata, 1)
            Dim matMxinv As Matrix

            Dim factor1 As Double
```

```
Dim factor2 As Double
Dim factor3 As Double
Dim factor4 As Double

For i As Integer = 0 To (MxToForm0.Count - 1)
    Dim cationMx As New Dictionary(Of String, Double)
    cationMx = SetCationDataForTetraedre(MxToForm0(i), WhichMethod)

    If WhichMethod = "CarbonateMethode1" Or WhichMethod = "CarbonateMethode2" Then
        factor1 = cationMx("Ca") / cationMx("SUM")
        factor2 = cationMx("Mg") / cationMx("SUM")
        factor3 = cationMx("Fe2") / cationMx("SUM")
        vect = {factor1, factor2, factor3}

    ElseIf WhichMethod = "Oxyde" Then
        factor1 = cationMx("Ti") / cationMx("SUM")
        factor2 = cationMx("Fe2") / cationMx("SUM")
        factor3 = cationMx("Fe3") / cationMx("SUM")
        vect = {factor1, factor2, factor3}

    ElseIf WhichMethod = "Silicate" Then
        factor1 = cationMx("Al") / cationMx("SUM")
        factor2 = cationMx("Ca") / cationMx("SUM")
        factor3 = (cationMx("K") + cationMx("Na") + cationMx("KNa")) / cationMx("SUM")
        factor4 = (cationMx("Fe2") + cationMx("Mg") + cationMx("MgFe")) / cationMx("SUM")
        vect = {factor1, factor2, factor3, factor4}
    End If

    For l As Integer = 0 To Ndata - 1
        matrixMx(l, i) = vect(l)
    Next
Next

If WhichMethod = "CarbonateMethode1" Then
    matB(0, 0) = M1
    matB(1, 0) = M3
    matB(2, 0) = M4
```

```
            ElseIf WhichMethod = "Oxyde" Then
                matB(0, 0) = M1
                matB(1, 0) = M2
                matB(2, 0) = M3
            ElseIf WhichMethod = "Silicate" Then
                matB(0, 0) = M1
                matB(1, 0) = M4
                matB(2, 0) = M2
                matB(3, 0) = M3
            End If

            matMxinv = matrixMx.Inverse()
            matrixMxProp = matMxinv * matB

            For k As Integer = 0 To Ndata - 1
                mole(MxToForm0(k)).NBmole = matrixMxProp(k, 0)
            Next
        End If
    End Sub
#End Region

#Region "TetraedreTools"

    'Convertion of data from Cartesian to Barycentric coordinates, and vice versa.

    Private Function NewPointCartesian(ByVal coordonneeX As Double, ByVal coordonneeY As Double, ByVal coordonneeZ As Double)
        Dim coordinates As New Dictionary(Of String, Double)

        coordinates.Add("coordHaut", 0)           'Tetraedre / top
        coordinates.Add("coordBasGauche", 0)      'Tetraedre / lower left corner
        coordinates.Add("coordBasDroite", 0)      'Tetraedre / lower right corner
        coordinates.Add("coordArriere", 0)        'Tetraedre / back
        coordinates.Add("coordonneeX", coordonneeX)
        coordinates.Add("coordonneeY", coordonneeY)
        coordinates.Add("coordonneeZ", coordonneeZ)

        Return coordinates
```

```
    End Function

    Private Function NewPointBarycentric(ByVal coordHaut As Double, ByVal coordBasGauche As Double, ByVal coordBasDroite As Double,
ByVal coordArriere As Double)
        Dim coordinates As New Dictionary(Of String, Double)

        coordinates.Add("coordHaut", coordHaut)           'Tetraedre / top
        coordinates.Add("coordBasGauche", coordBasGauche)      'Tetraedre / lower left corner
        coordinates.Add("coordBasDroite", coordBasDroite)      'Tetraedre / lower right corner
        coordinates.Add("coordArriere", coordArriere)        'Tetraedre / back
        coordinates.Add("coordonneeX", 0)
        coordinates.Add("coordonneeY", 0)
        coordinates.Add("coordonneeZ", 0)

        Return coordinates
    End Function

    Friend Sub calculerCoordonneesCartesiennes(ByRef coordinates As Dictionary(Of String, Double), ByVal TetraCoord As Dictionary(Of
Integer, Dictionary(Of String, Double)))

        Dim matB As New Matrix(4, 1)
        Dim matX As New Matrix(4, 1)
        Dim matA As Matrix
        matA = monterMatriceA(TetraCoord)     'Tetraedre (its 4 defining points)

        matX(0, 0) = coordinates("coordHaut")        'Barycentric coordinates of 1 point
        matX(1, 0) = coordinates("coordBasGauche")
        matX(2, 0) = coordinates("coordBasDroite")
        matX(3, 0) = coordinates("coordArriere")

        matB = matA * matX

        coordinates("coordonneeX") = matB(0, 0)      'Cartesian coordinates of 1 point
        coordinates("coordonneeY") = matB(1, 0)
        coordinates("coordonneeZ") = matB(2, 0)
    End Sub
```

```
    Friend Sub calculerCoordonneesBarycentriques(ByRef coordinates As Dictionary(Of String, Double), ByVal TetraCoord As
Dictionary(Of Integer, Dictionary(Of String, Double)))

        Dim matB As New Matrix(4, 1)
        Dim matX As New Matrix(4, 1)
        Dim matA As Matrix
        Dim matAinv As Matrix

        matA = monterMatriceA(TetraCoord)   'Tetraedre (its 4 defining points)
        matAinv = matA.Inverse()

        matB(0, 0) = coordinates("coordonneeX")    'Cartesian coordinates of 1 point
        matB(1, 0) = coordinates("coordonneeY")
        matB(2, 0) = coordinates("coordonneeZ")
        matB(3, 0) = 1

        matX = matAinv * matB

        coordinates("coordHaut") = matX(0, 0)       'Barycentric coordinates of the 1 point in the considered tetraedre
        coordinates("coordBasGauche") = matX(1, 0)
        coordinates("coordBasDroite") = matX(2, 0)
        coordinates("coordArriere") = matX(3, 0)
    End Sub

    Protected Function monterMatriceA(ByVal TetraCoord As Dictionary(Of Integer, Dictionary(Of String, Double))) As Matrix

        Dim matA As New Matrix(4, 4)

        If TetraCoord.Count = 4 Then

            Dim Pt1 As New Dictionary(Of String, Double)
            Dim Pt2 As New Dictionary(Of String, Double)
            Dim Pt3 As New Dictionary(Of String, Double)
            Dim Pt4 As New Dictionary(Of String, Double)
            Pt1 = TetraCoord(1)
            Pt2 = TetraCoord(2)
            Pt3 = TetraCoord(3)
```

```
            Pt4 = TetraCoord(4)

            matA(0, 0) = Pt1("coordonneeX")
            matA(1, 0) = Pt1("coordonneeY")
            matA(2, 0) = Pt1("coordonneeZ")
            matA(3, 0) = 1

            matA(0, 1) = Pt2("coordonneeX")
            matA(1, 1) = Pt2("coordonneeY")
            matA(2, 1) = Pt2("coordonneeZ")
            matA(3, 1) = 1

            matA(0, 2) = Pt3("coordonneeX")
            matA(1, 2) = Pt3("coordonneeY")
            matA(2, 2) = Pt3("coordonneeZ")
            matA(3, 2) = 1

            matA(0, 3) = Pt4("coordonneeX")
            matA(1, 3) = Pt4("coordonneeY")
            matA(2, 3) = Pt4("coordonneeZ")
            matA(3, 3) = 1
        End If
        Return matA
    End Function
#End Region


#Region "CarbonateCalculation"


    Private Sub MakeCarbonate()

        Dim CMMexcess As Double
        If mole("C").NBmole > 0 Then
            If mole("C").NBmole < mole("Ca").NBmole Then
                mole("calcite").NBmole = mole("C").NBmole
```

```
            ElseIf mole("C").NBmole < (mole("Ca").NBmole + (mole("Fe2").NBmole + mole("Mg").NBmole) / 4) Then
                CMMexcess = mole("C").NBmole - (mole("Ca").NBmole + mole("Mg").NBmole / 4 + mole("Fe2").NBmole / 4)
                CalcCarbonates(mole("Ca").NBmole, 0, mole("Mg").NBmole / 4, mole("Fe2").NBmole / 4, "CarbonateMethode1", CMMexcess)

            ElseIf mole("C").NBmole < (mole("Ca").NBmole + (mole("Fe2").NBmole + mole("Mg").NBmole) / 2) Then
                CMMexcess = mole("C").NBmole - (mole("Ca").NBmole + mole("Mg").NBmole / 2 + mole("Fe2").NBmole / 2)
                CalcCarbonates(mole("Ca").NBmole, 0, mole("Mg").NBmole / 2, mole("Fe2").NBmole / 2, "CarbonateMethode1", CMMexcess)

            Else
                CMMexcess = mole("C").NBmole - (mole("Ca").NBmole + mole("Mg").NBmole + mole("Fe2").NBmole)
                CalcCarbonates(mole("Ca").NBmole, 0, mole("Mg").NBmole, mole("Fe2").NBmole, "CarbonateMethode1", CMMexcess)
            End If

            'Update CaMM, FeMM, MgMM according to amount consumed by carbonates
            Dim list1() As String = {"calcite", "magnesite", "siderite", "dolomite", "ankerite"}
            Dim factorCa() As Double = {1, 0, 0, 0.5, 0.5}
            Dim factorMg() As Double = {0, 1, 0, 0.5, 0}

            For i As Integer = 0 To list1.Count - 1
                mole("Ca").NBmole = mole("Ca").NBmole - mole(list1(i)).NBmole * factorCa(i)
                mole("Mg").NBmole = mole("Mg").NBmole - mole(list1(i)).NBmole * factorMg(i)
                mole("Fe2").NBmole = mole("Fe2").NBmole - mole(list1(i)).NBmole * (1 - factorCa(i) - factorMg(i))
            Next
        End If
    End Sub

    Sub CalcCarbonates(ByVal M1 As Double, ByVal M2 As Double, ByVal M3 As Double, ByVal M4 As Double, ByVal WhichMethod As String,
ByVal CMMexcess As String)

        inverseMatriceGetMxMole(M1, M2, M3, M4, WhichMethod)

        Dim propCC As Double = 0
        Dim propMST As Double = 0
        Dim propSID As Double = 0
        Dim propDOL As Double = 0
        Dim propANK As Double = 0
```

```
For Each minName As String In mole.Keys
    If mole(minName).typeValue = ValueMx.enumTypeValue.carbonate Then
        Dim cationMx As New Dictionary(Of String, Double)
        cationMx = MineralsDefinition(minName)

        If minName.Contains("CCmst") Or minName.Contains("MSTcc") Then
            propCC += mole(minName).NBmole * cationMx("Ca")
            propMST += mole(minName).NBmole * cationMx("Mg")
            mole(minName).NBmole = 0
        ElseIf minName.Contains("CCsid") Or minName.Contains("SIDcc") Then
            propCC += mole(minName).NBmole * cationMx("Ca")
            propSID += mole(minName).NBmole * cationMx("Fe2")
            mole(minName).NBmole = 0
        ElseIf minName.Contains("SIDmst") Or minName.Contains("CCsidmst") Then
            propCC += mole(minName).NBmole * cationMx("Ca")
            propSID += mole(minName).NBmole * cationMx("Fe2")
            propMST += mole(minName).NBmole * cationMx("Mg")
            mole(minName).NBmole = 0
        ElseIf minName.Contains("ankerite0") Or minName.Contains("ankerite1") Then
            Dim FactorFEMG As Double = cationMx("Mg") / (cationMx("Mg") + cationMx("Fe2"))
            propDOL += mole(minName).NBmole * FactorFEMG
            propANK += mole(minName).NBmole * (1 - FactorFEMG)
            mole(minName).NBmole = 0
        End If
    End If
Next

mole("calcite").NBmole += propCC
mole("magnesite").NBmole += propMST
mole("siderite").NBmole += propSID
mole("dolomite").NBmole += propDOL
mole("ankerite").NBmole += propANK


If CMMexcess > 0 Then
    mole("graphite").NBmole = CMMexcess
End If
```

```
    End Sub

#End Region

#Region "OxydeCalculation"

    Private Sub MakeOxyde(ByVal Fe3forOxydes As Double)
        If mole("Ti").NBmole > 0 Or Fe3forOxydes > 0 Then
            If Fe2forOxydes = -1 Then
                If (FEoxydeRatio * mole("Fe3").NBmole) >= mole("Fe2").NBmole Then
                    Fe2forOxydes = mole("Fe2").NBmole
                    mole("Fe2").NBmole = 0.0001
                Else
                    Fe2forOxydes = FEoxydeRatio * mole("Fe3").NBmole
                    mole("Fe2").NBmole = mole("Fe2").NBmole - Fe2forOxydes
                End If
            End If

            inverseMatriceGetMxMole(mole("Ti").NBmole, Fe2forOxydes, Fe3forOxydes, 0, "Oxyde")
            Dim propHEM As Double = 0
            Dim propILM As Double = 0
            Dim propPSB As Double = 0

            For Each minName As String In mole.Keys
                If mole(minName).typeValue = ValueMx.enumTypeValue.oxyde Then
                    If minName.Contains("HEMilm") Then
                        Dim cationMx As New Dictionary(Of String, Double)
                        cationMx = MineralsDefinition(minName)
                        propHEM += mole(minName).NBmole * (cationMx("Fe3") / 2)
                        propILM += mole(minName).NBmole * cationMx("Fe2")
                        mole(minName).NBmole = 0

                    ElseIf minName.Contains("PSBfe") Then
                        propPSB += mole(minName).NBmole
                        mole(minName).NBmole = 0
                    End If
                End If
```

```
            Next

            mole("hematite").NBmole += propHEM
            mole("ilmenite").NBmole += propILM
            mole("pseudobrookite").NBmole += propPSB
        End If
    End Sub
#End Region

#Region "SilicateCalculation"

    Private Sub MakeSilicate()

        SelectAssemblageSilicate()

        Dim Calculate As Boolean = True   'Trick to enable silicate re-calculation in cases for which Qtz+carbonates have to react
        For i As Integer = 0 To 1 Step 1
            If Calculate = True Then : Calculate = False

                mxToBeFormed(mole("Al").NBmole, mole("K").NBmole + mole("Na").NBmole, mole("Fe2").NBmole + mole("Mg").NBmole +
mole("Mn").NBmole, mole("Ca").NBmole, "Silicate")

                If Not MxToForm0 Is Nothing Then
                    If MxToForm0.Contains("NaK") Then   'The mineral assemblage cannot be in the DeadZone (cf. there is no K-Na
silicates or oxydes) / So, if it is, decrease the amount of K and Na and try again to find a realistic assemblage
                        Dim factorN As Double = (mole("K").NBmole + mole("Na").NBmole) / 20
                        For j As Double = factorN To (mole("Na").NBmole + mole("K").NBmole) Step factorN
                            mole("Na").NBmole = mole("Na").NBmole - factorN * NaRatio
                            mole("K").NBmole = mole("K").NBmole - factorN * (1 - NaRatio)

                            mxToBeFormed(mole("Al").NBmole, mole("K").NBmole + mole("Na").NBmole, mole("Fe2").NBmole +
mole("Mg").NBmole + mole("Mn").NBmole, mole("Ca").NBmole, "Silicate")

                            If MxToForm0 Is Nothing Then
                                mole("Na").NBmole = mole("Na").NBmole + factorN * NaRatio
                                mole("K").NBmole = mole("K").NBmole + factorN * (1 - NaRatio)
```

```
                                mxToBeFormed(mole("Al").NBmole, mole("K").NBmole + mole("Na").NBmole, mole("Fe2").NBmole +
mole("Mg").NBmole + mole("Mn").NBmole, mole("Ca").NBmole, "Silicate")
                                Exit For
                            ElseIf Not MxToForm0.Contains("NaK") Then
                                mole("Na2O").NBmole = j * NaRatio
                                mole("K2O").NBmole = j * (1 - NaRatio)
                                Exit For
                            End If
                        Next
                    End If
                    CalcSilicatesPart1()
                    CalcSilicatesPart2()
                End If

                'If Qtz has been made, have carbonate react with Qtz (except for the 4 lowest temperature facies)
                Dim Y As New Dictionary(Of String, Double)
                If mole("quartz").NBmole > 0 And mole("C").NBmole > 0 And i = 0 Then
                    If FaciesEnum <> FormMain.enumFacies.facies7AMP500 And FaciesEnum <> FormMain.enumFacies.facies5SV400 And
FaciesEnum <> FormMain.enumFacies.facies9SB450 And FaciesEnum <> FormMain.enumFacies.facies10SB350 Then

                        'Update CaMM, FeMM, MgMM according to amount consumed by carbonates
                        Dim list1() As String = {"calcite", "magnesite", "siderite", "dolomite", "ankerite"}
                        Dim factorCa() As Double = {1, 0, 0, 0.5, 0.5}
                        Dim factorMg() As Double = {0, 1, 0, 0.5, 0}
                        Dim CaCarb As Double
                        Dim FeCarb As Double
                        Dim MgCarb As Double

                        For k As Integer = 0 To list1.Count - 1
                            CaCarb += mole(list1(k)).NBmole * factorCa(k)
                            MgCarb += mole(list1(k)).NBmole * factorMg(k)
                            FeCarb += mole(list1(k)).NBmole * (1 - factorCa(k) - factorMg(k))
                        Next

                        mxToBeFormed(CaCarb, mole("quartz").NBmole, MgCarb, FeCarb, "CarbonateMethode2")

                        If DestroyCARBorNot = True Then
```

```
                            'clear carbonate + silicate dico
                            For Each Mx As String In mole.Keys
                                If mole(Mx).typeValue = ValueMx.enumTypeValue.silicate Or mole(Mx).typeValue =
ValueMx.enumTypeValue.carbonate Then
                                    mole(Mx).NBmole = 0
                                End If
                            Next

                            'All C goes in graphite
                            mole("graphite").NBmole = mole("C").NBmole

                            'Re-calculate silicates with additional Ca-Fe-Mg
                            Calculate = True
                            mole("Fe2").NBmole += FeCarb
                            mole("Mg").NBmole += MgCarb
                            mole("Ca").NBmole += CaCarb
                        End If
                    End If
                End If
            End If
        Next
    End Sub

    Private Sub ResetKratio()

        Dim NAmineralsMM As Double = 0
        Dim kmineralsMM As Double = 0

        If KRatioForWM > 0 And Kratio <> KRatioForWM And mole("WhiteMica").NBmole > 0 Then
            kmineralsMM = mole("WhiteMica").NBmole * (KRatioForWM / 4)                ' 4 = SumConstituantACKF
            NAmineralsMM = mole("WhiteMica").NBmole * ((1 - KRatioForWM) / 4)
        End If

        For Each Mx As String In NaKproblems.Keys
            Dim cation As New Dictionary(Of String, Double)
            cation = SetCationData(Mx)
```

```
            If mole(Mx).NBmole > 0 Then
                If NaKproblems(Mx) = "K" Then
                    kmineralsMM += mole(Mx).NBmole * (cation("K") / cation("SUM"))
                ElseIf NaKproblems(Mx) = "Na" Then
                    NAmineralsMM += mole(Mx).NBmole * (cation("Na") / cation("SUM"))
                End If
            End If
        Next

        Dim NewKratio As Double = (mole("K").NBmole - kmineralsMM) / ((mole("K").NBmole - kmineralsMM) + (mole("Na").NBmole -
NAmineralsMM))

        If Double.IsNaN(NewKratio) Then : NewKratio = Kratio
        End If
        If NewKratio < 0 Then : NewKratio = 0
        ElseIf NewKratio > 1 Then : NewKratio = 1
        End If

        Kratio = NewKratio
    End Sub

    Private Sub ResetMGratio()
        Dim FeToRemove As Double = 0
        Dim MgToRemove As Double = 0

        For Each Mx As String In MgFeproblems.Keys
            Dim cation As New Dictionary(Of String, Double)
            cation = SetCationData(Mx)

            If mole(Mx).NBmole > 0 Then
                If MgFeproblems(Mx) = "Fe" Then
                    FeToRemove += mole(Mx).NBmole * (cation("Fe2") / cation("SUM"))
                ElseIf MgFeproblems(Mx) = "Mg" Then
                    MgToRemove += mole(Mx).NBmole * (cation("Mg") / cation("SUM"))
                End If
            End If
        Next
```

```
        Dim newMgFeRatio As Double = 0

        If mole("spessartine").NBmole > 0 Or mole("MnO").NBmole > 0 Or mole("tephroite").NBmole > 0 Or mole("rhodonite").NBmole > 0
Or mole("pyroxmangite").NBmole > 0 Then
            newMgFeRatio = (mole("Mg").NBmole - MgToRemove) / ((mole("Mg").NBmole - MgToRemove) + (mole("Fe2").NBmole - FeToRemove))
        Else
            newMgFeRatio = (mole("Mg").NBmole - MgToRemove) / ((mole("Mg").NBmole - MgToRemove) + ((mole("Fe2").NBmole +
mole("Mn").NBmole) - FeToRemove))
        End If

        If Double.IsNaN(newMgFeRatio) Then : newMgFeRatio = MgFeRatio
        End If
        If newMgFeRatio < 0 Then : newMgFeRatio = 0
        ElseIf newMgFeRatio > 1 Then : newMgFeRatio = 1
        End If

        MgFeRatio = newMgFeRatio
    End Sub


    Private Sub CalcSilicatesPart1()

        If Not MxToForm0 Is Nothing Then
            inverseMatriceGetMxMole(mole("Al").NBmole, mole("K").NBmole + mole("Na").NBmole, mole("Fe2").NBmole + mole("Mg").NBmole +
mole("Mn").NBmole, mole("Ca").NBmole, "Silicate", MxToForm0)

            'MANAGE  THE  EXCEPTIONS
            For Each minname As String In mole.Keys
                If mole(minname).typeValue = ValueMx.enumTypeValue.silicate Then
                    If minname.Contains("FSP") And mole(minname).NBmole > 0 Then
                        feldsparSS(minname)
                    ElseIf minname = "GRThalf" Or minname = "GROSSss" Then
                        If mole(minname).NBmole > 0 Then
                            GRThalfSS()
                        End If
                    ElseIf minname = "CPXgrt" Or minname = "GRTss" Or minname = "GRTss05" Then
```

```
            If mole(minname).NBmole > 0 Then
                CPXgrtSS()
            End If
        End If
    End If
Next

'Sudoite contains only Mg (no Fe here)
If mole("sudoite").NBmole > 0 And MgFeRatio < 0.9 Then
    sudoitePbl()
End If

'Distribution of K (biotite problem) / Remove Na from biotite by making albite + FM
If mole("biotite").NBmole > 0 Then
    If (mole("biotite").NBmole / 5) > mole("K").NBmole Then
        biotitePbl()
    End If
End If

'Provisional albite-orthose (needed for Hornblende calculation)
KRatioForWM = 0
If mole("Kfeld").NBmole > 0 Then

    'WhiteMica exception (only if WM co-exist with Kfeld)
    If mole("WhiteMica").NBmole > 0 Then

        Dim NaLeft As Double = 0
        KRatioForWM = Kratio * Kratio * Kratio * Kratio * Kratio * Kratio
        NaLeft = mole("Na").NBmole - mole("WhiteMica").NBmole * ((1 - KRatioForWM) / 4)  '4 = SumConstituantACKF
        If NaLeft < 0 Then
            KRatioForWM = Kratio * Kratio * Kratio * Kratio
            NaLeft = mole("Na").NBmole - mole("WhiteMica").NBmole * ((1 - KRatioForWM) / 4)
            If NaLeft < 0 Then
                KRatioForWM = Kratio * Kratio
                NaLeft = mole("Na").NBmole - mole("WhiteMica").NBmole * ((1 - KRatioForWM) / 4)
                If NaLeft < 0 Then
                    KRatioForWM = Kratio
```

```
                        End If
                    End If
                End If
            End If

            ResetKratio()
            mole("albite").NBmole = (1 - Kratio) * mole("Kfeld").NBmole
            mole("orthose").NBmole = Kratio * mole("Kfeld").NBmole
            mole("Kfeld").NBmole = 0
        End If

        'Make Hornblende from FM + anorthite, etc., for amphibolite facies only
        If MgFeRatio > 0.1 Then       'No amphibole for Fe-richest rocks
            Dim MakeAmph As Boolean = False

            'only for Amphibolite facies assemblages, and adjust Mgratio
            If FaciesEnum = FormMain.enumFacies.facies3AMP675 And MgFeRatio > 0.1 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies5AMP670 And MgFeRatio > 0.15 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP700 And MgFeRatio > 0.5 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 And MgFeRatio > 0.15 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies10AMP700 And MgFeRatio > 0.6 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 And MgFeRatio > 0.1 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies4AMP600 And MgFeRatio > 0.2 Then : MakeAmph = True
            ElseIf FaciesEnum = FormMain.enumFacies.facies6AMP600 And MgFeRatio > 0.2 Then : MakeAmph = True
            End If

            If MakeAmph Then
                If mole("FM").NBmole > 0 And mole("anorthite").NBmole > 0 And mole("albite").NBmole > 0 And
mole("actinolite").NBmole > 0 Then
                    PargasiteCreation01()
                ElseIf mole("CPX").NBmole > 0 And mole("anorthite").NBmole > 0 And mole("albite").NBmole > 0 And
mole("actinolite").NBmole > 0 Then
                    PargasiteCreation02()
                ElseIf mole("CPX").NBmole > 0 And mole("anorthite").NBmole > 0 And mole("albite").NBmole > 0 Then
                    If mole("grossular").NBmole > 0 Or mole("clinozoisite").NBmole > 0 Or mole("wollastonite").NBmole > 0 Then
                        PargasiteCreation03()
                    End If
```

```
                    ElseIf mole("FM").NBmole > 0 And mole("anorthite").NBmole > 0 And mole("albite").NBmole > 0 Then
                        If mole("amesite").NBmole > 0 Or mole("cordierite").NBmole > 0 Or mole("GRTpyalm").NBmole > 0 Then
                            PargasiteCreation04()
                        End If
                    End If

                    If mole("cordierite").NBmole > 0 And mole("actinolite").NBmole > 0 And mole("Kfeld").NBmole > 0 Then
                        PargasiteCreation05()
                    ElseIf mole("amesite").NBmole > 0 And mole("actinolite").NBmole > 0 And mole("Kfeld").NBmole > 0 Then
                        PargasiteCreation06()
                    End If

                    If mole("FM").NBmole > 0 And mole("anorthite").NBmole > 0 Then    'Make tschermakite if there is FM + anorthite
left
                        FManorthiteReaction()
                    End If
                End If
            End If

            'FM: use a triangle and get proper FM for each facies
            If mole("FM").NBmole > 0 Then
                FMMminerals()
            End If
        End If

    End Sub

    Public Sub CalcSilicatesPart2()

        'Amount of Si needed?
        Dim SiMMneeded As Double = ELEMENTinSilicatesAutres("Si")
        SiToGain = SiMMneeded - mole("Si").NBmole

        'Mx Destruction to solve Si-deficit
        If SiToGain < 0 Then
            mole("quartz").NBmole = mole("Si").NBmole - SiMMneeded
        ElseIf SiToGain > 0 Then
```

```
            ResetMGratio()
            SiExcessDeficit()
        End If


        'Manage SS
        'Remove provisional orthose-albite
        If mole("albite").NBmole > 0 Or mole("orthose").NBmole > 0 Then
            mole("Kfeld").NBmole = mole("albite").NBmole + mole("orthose").NBmole
            mole("albite").NBmole = 0
            mole("orthose").NBmole = 0
        End If

        'Manage SS
        ResetMGratio()
        ResetKratio()
        ManageSS()
    End Sub


    Public Function WhichMGO() As String       'Brucite or Periclase?

        Dim NameMGO As String : NameMGO = "brucite"
        If FaciesEnum = FormMain.enumFacies.facies3GRA800 Or FaciesEnum = FormMain.enumFacies.facies5GRA900 Or FaciesEnum =
FormMain.enumFacies.facies10GRA950 Then
            NameMGO = "periclase"
        End If
        Return NameMGO
    End Function

    Public Function WhichALO() As String       'Diaspore or Corindon?

        Dim NameALO As String = "corundum"
        If FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum = FormMain.enumFacies.facies9SB450 Or FaciesEnum =
FormMain.enumFacies.facies10SB350 Then
            NameALO = "diaspore"
        End If
```

```
        Return NameALO
    End Function

    Public Function WhichMNO() As String       'pyroxmangite or rhodonite?

        Dim NameMNO As String = "rhodonite"
        If FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum =
FormMain.enumFacies.facies9SB450 Or FaciesEnum = FormMain.enumFacies.facies10SB350 Then
            NameMNO = "pyroxmangite"
        End If
        Return NameMNO
    End Function

    Public Function WhichALUMINO() As String     'Which aluminosilicate?

        Dim NameALU As String = "sillimanite"
        If FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum = FormMain.enumFacies.facies10SB350 Then
            NameALU = "pyrophyllite"
        ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 Or FaciesEnum = FormMain.enumFacies.facies10AMP700 Or FaciesEnum =
FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies9SB450 Or FaciesEnum = FormMain.enumFacies.facies6AMP600
Then
            NameALU = "kyanite"
        End If
        Return NameALU
    End Function
#End Region

#Region "SilicateExceptions"

    'Additional reactions between silicates.

    Sub GRThalfSS()
        Dim FactorA As Double = 0

        Dim propGARNETss As Double = 0
        If mole("GRThalf").NBmole > 0 Then : propGARNETss = mole("GRThalf").NBmole
            mole("GRThalf").NBmole = 0
```

```
        FactorA = 2 / 3
    ElseIf mole("GROSSss").NBmole > 0 Then : propGARNETss = mole("GROSSss").NBmole
        mole("GROSSss").NBmole = 0
        FactorA = 1 / 3
    End If

    mole("GRTpyalm").NBmole = mole("GRTpyalm").NBmole + FactorA * propGARNETss
    mole("grossular").NBmole = mole("grossular").NBmole + (1 - FactorA) * propGARNETss

End Sub

Sub CPXgrtSS()

    Dim Factor05 As Double = 1

    Dim propCPX As Double = 0
    If mole("CPXgrt").NBmole > 0 Then
        mole("CPX").NBmole = mole("CPXgrt").NBmole
        mole("CPXgrt").NBmole = 0
    End If
    Dim propGRTpyalm As Double = 0
    If mole("GRTss").NBmole > 0 Then
        mole("GRTpyalm").NBmole = mole("GRTss").NBmole
        mole("GRTss").NBmole = 0
    End If
    If mole("GRTss05").NBmole > 0 Then
        mole("GRTpyalm").NBmole = mole("GRTss05").NBmole
        mole("GRTss05").NBmole = 0
        Factor05 = 0.5
    End If

    Dim propAn As Double = 0
    Dim propAn2 As Double = 0
    If mole("anorthite").NBmole > 0 Then : propAn = (mole("anorthite").NBmole) * Factor05
        mole("anorthite").NBmole = 0
        If Factor05 = 0.5 Then : propAn2 = propAn
        End If
```

```
    End If
    Dim propFM As Double = 0
    Dim propFM2 As Double = 0
    If mole("FM").NBmole > 0 Then : propFM = (mole("FM").NBmole) * Factor05
        mole("FM").NBmole = 0
        If Factor05 = 0.5 Then : propFM2 = propFM
        End If
    End If
    Dim propGross As Double = 0
    Dim propGRT As Double = 0

    If propFM > (6 / 9) * propAn Then
        propGross = (5 / 9) * propAn
        propGRT = (4 / 9) * propAn + (6 / 9) * propAn
        propFM = propFM - (6 / 9) * propAn
        propAn = 0
    Else
        propGross = (5 / 9) * propAn
        propGRT = propFM + (2 / 9) * propFM
        propAn = (4 / 9) * propAn - (2 / 9) * propFM
        propFM = 0
    End If

    If propGRT > 0 Then : mole("GRTpyalm").NBmole = mole("GRTpyalm").NBmole + propGRT
    End If
    If propGross > 0 Then : mole("grossular").NBmole = mole("grossular").NBmole + propGross
    End If
    If propFM > 0 Or propFM2 > 0 Then : mole("FM").NBmole = propFM + propFM2
    End If
    If propAn > 0 Or propAn2 > 0 Then : mole("anorthite").NBmole = propAn + propAn2
    End If

End Sub

Sub sudoitePbl()

    mole("sudoite").NBmole = mole("sudoite").NBmole * MgFeRatio * MgFeRatio
```

```
    mole("chloritoid").NBmole = mole("sudoite").NBmole * (1 - MgFeRatio * MgFeRatio)

End Sub

Sub feldsparSS(ByVal minName As String)

    Dim cation As New Dictionary(Of String, Double)
    cation = SetCationData(minName)

    Dim propFELDss As Double = mole(minName).NBmole

    Dim propAn As Double = propFELDss * (cation("Ca") / cation("SUM")) * 3
    Dim propKfeld As Double = propFELDss * ((cation("K") + cation("Na")) / cation("SUM")) * 2

    mole("anorthite").NBmole = mole("anorthite").NBmole + propAn
    mole("Kfeld").NBmole = mole("Kfeld").NBmole + propKfeld

    mole(minName).NBmole = 0
End Sub

Sub biotitePbl()

    Dim propBT As Double = mole("biotite").NBmole
    mole("biotite").NBmole = mole("K").NBmole * 5

    Dim propKfeld As Double = mole("Kfeld").NBmole
    Dim propFM As Double = mole("FM").NBmole
    Dim propGlauco As Double = 0

    If FaciesEnum = FormMain.enumFacies.facies10SB350 Or FaciesEnum = FormMain.enumFacies.facies9SB450 Then
        propGlauco = (propBT - (mole("K").NBmole * 5)) * (3.5 / 5)
        propFM = propFM + (propBT - (mole("K").NBmole * 5)) * (1.5 / 5)
    Else
        propKfeld = propKfeld + (propBT - (mole("K").NBmole * 5)) * (2 / 5)
        propFM = propFM + (propBT - (mole("K").NBmole * 5)) * (3 / 5)
    End If
```

```
Dim nameALUMINO As String = WhichALUMINO()
If mole(nameALUMINO).NBmole > 0 And propFM > 0 Then
    Dim propALUMINO As Double = mole(nameALUMINO).NBmole

    If mole("staurolite").NBmole > 0 Then
        Dim propSTAU As Double = mole("staurolite").NBmole
        If propALUMINO > propFM * 4.5 Then
            propSTAU = propSTAU + propFM * (22 / 4)
            propALUMINO = propALUMINO - propFM * (18 / 4)
            propFM = 0
        Else
            propSTAU = propSTAU + propALUMINO * (22 / 18)
            propFM = propFM - propALUMINO * (4 / 18)
            propALUMINO = 0
        End If
        mole("staurolite").NBmole = propSTAU

    ElseIf mole("cordierite").NBmole > 0 Then
        Dim propCORD As Double = mole("cordierite").NBmole
        If propALUMINO > propFM * 2 Then
            propCORD = propCORD + propFM * 3
            propALUMINO = propALUMINO - propFM * 2
            propFM = 0
        Else
            propCORD = propCORD + propALUMINO * (6 / 4)
            propFM = propFM - propALUMINO * 0.5
            propALUMINO = 0
        End If
        mole("cordierite").NBmole = propCORD

    ElseIf mole("sudoite").NBmole > 0 Then
        Dim propSUD As Double = mole("sudoite").NBmole
        If propALUMINO > propFM * 2 Then
            propSUD = propSUD + propFM * 3
            propALUMINO = propALUMINO - propFM * 2
            propFM = 0
        Else
```

```
            propSUD = propSUD + propALUMINO * (6 / 4)
            propFM = propFM - propALUMINO * 0.5
            propALUMINO = 0
        End If
        mole("sudoite").NBmole = propSUD

    ElseIf mole("chloritoid").NBmole > 0 Then
        Dim propCTD As Double = mole("chloritoid").NBmole
        If propALUMINO > propFM * 2 Then
            propCTD = propCTD + propFM * 3
            propALUMINO = propALUMINO - propFM * 2
            propFM = 0
        Else
            propCTD = propCTD + propALUMINO * (3 / 2)
            propFM = propFM - propALUMINO * 0.5
            propALUMINO = 0
        End If
        mole("chloritoid").NBmole = propCTD

    ElseIf mole("carpholite").NBmole > 0 Then
        Dim propCAR As Double = mole("carpholite").NBmole
        If propALUMINO > propFM * 2 Then
            propCAR = propCAR + propFM * 3
            propALUMINO = propALUMINO - propFM * 2
            propFM = 0
        Else
            propCAR = propCAR + propALUMINO * (3 / 2)
            propFM = propFM - propALUMINO * 0.5
            propALUMINO = 0
        End If
        mole("carpholite").NBmole = propCAR

    ElseIf mole("spinelss").NBmole > 0 Then
        Dim propHER As Double = mole("spinelss").NBmole
        If propALUMINO > propFM * 2 Then
            propHER = propHER + propFM * 3
            propALUMINO = propALUMINO - propFM * 2
```

```
            propFM = 0
        Else
            propHER = propHER + propALUMINO * (3 / 2)
            propFM = propFM - propALUMINO * 0.5
            propALUMINO = 0
        End If
        mole("spinelss").NBmole = propHER

    ElseIf mole("daphnite").NBmole > 0 Then
        Dim propDAPH As Double = mole("daphnite").NBmole
        If propALUMINO > propFM * 2.5 Then
            propDAPH = propDAPH + propFM * (7 / 5)
            propALUMINO = propALUMINO - propFM * (2 / 5)
            propFM = 0
        Else
            propDAPH = propDAPH + propALUMINO * (7 / 2)
            propFM = propFM - propALUMINO * (5 / 2)
            propALUMINO = 0
        End If
        mole("daphnite").NBmole = propDAPH

    ElseIf mole("GRTpyalm").NBmole > 0 Then
        Dim propGRT As Double = mole("GRTpyalm").NBmole
        If propALUMINO * 1.5 > propFM Then
            propGRT = propGRT + propFM * (5 / 3)
            propALUMINO = propALUMINO - propFM * (2 / 3)
            propFM = 0
        Else
            propGRT = propGRT + propALUMINO * (5 / 2)
            propFM = propFM - propALUMINO * (3 / 2)
            propALUMINO = 0
        End If
        mole("GRTpyalm").NBmole = propGRT

    ElseIf mole("amesite").NBmole > 0 Then
        Dim propAME As Double = mole("amesite").NBmole
        If propALUMINO > propFM Then
```

```
                propAME = propAME + propFM * 2
                propALUMINO = propALUMINO - propFM
                propFM = 0
            Else
                propAME = propAME + propALUMINO * 2
                propFM = propFM - propALUMINO
                propALUMINO = 0
            End If
            mole("amesite").NBmole = propAME
        End If

        mole(nameALUMINO).NBmole = propALUMINO
    End If

    mole("Kfeld").NBmole = propKfeld
    mole("FM").NBmole = propFM
    mole("glaucophane").NBmole = propGlauco
End Sub

Sub FMMminerals()
    Dim FMMprop As Double = mole("FM").NBmole
    mole("FM").NBmole = 0

    Dim MgPercent As Double = mole("Mg").NBmole / (mole("Mn").NBmole + mole("Fe2").NBmole + mole("Mg").NBmole)
    Dim FePercent As Double = mole("Fe2").NBmole / (mole("Mn").NBmole + mole("Fe2").NBmole + mole("Mg").NBmole)
    Dim MnPercent As Double = mole("Mn").NBmole / (mole("Mn").NBmole + mole("Fe2").NBmole + mole("Mg").NBmole)

    Dim MgMineralName As String = ""
    Dim MnMineralName As String = "rhodonite"
    Dim FePercent2 As Double = 0
    Dim nMx As Integer = 0
    Dim PxmnThere As Boolean = False
    Dim Limit01 As Double = 0
    Dim Limit02 As Double = 0
    Dim Limit03 As Double = 0
    Dim Limit04 As Double = 0
```

```
        If FaciesEnum = FormMain.enumFacies.facies9GRA800 Or FaciesEnum = FormMain.enumFacies.facies4GRA750 Then
            If MgMnRatio < 0.65 Then
                mole("rhodonite").NBmole = MnPercent * FMMprop
                mole("anthophyllite").NBmole = MgPercent * FMMprop
                mole("olivine").NBmole = FePercent * FMMprop
            Else
                mole("anthophyllite").NBmole = MgPercent * FMMprop
                mole("olivine").NBmole = (FePercent + MnPercent) * FMMprop
            End If

        Else
            If FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum =
FormMain.enumFacies.facies9SB450 Then
                MgMineralName = "talc"
                Limit01 = 0.8
                Limit02 = 0.3
                Limit03 = 0.3
                Limit04 = 0.9
                nMx = 2
                PxmnThere = True
                MnMineralName = "pyroxmangite"
            ElseIf FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                MgMineralName = "talc"
                Limit01 = 0.79
                Limit02 = 0.23
                Limit03 = 0.01
                Limit04 = 0.8
                nMx = 2
                PxmnThere = True
                MnMineralName = "pyroxmangite"
            ElseIf FaciesEnum = FormMain.enumFacies.facies3AMP675 Then
                MgMineralName = "talc"
                Limit01 = 0.97
                Limit02 = 0.87
                Limit03 = 0.67
                nMx = 2
            ElseIf FaciesEnum = FormMain.enumFacies.facies3GRA800 Then
```

```
        MgMineralName = "OPX"
        Limit01 = 0.99
        Limit02 = 0.98
        Limit03 = 0.85
        nMx = 1
    ElseIf FaciesEnum = FormMain.enumFacies.facies5GRA900 Then
        MgMineralName = "OPX"
        Limit01 = 0.99
        Limit02 = 0.92
        Limit03 = 0.85
        nMx = 1
    ElseIf FaciesEnum = FormMain.enumFacies.facies5AMP670 Or FaciesEnum = FormMain.enumFacies.facies7AMP700 Then
        MgMineralName = "talc"
        Limit01 = 0.95
        Limit02 = 0.81
        Limit03 = 0.5
        nMx = 2
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 Then
        MgMineralName = "talc"
        Limit01 = 0.9
        Limit02 = 0.6
        Limit03 = 0.65
        nMx = 2
    ElseIf FaciesEnum = FormMain.enumFacies.facies10GRA950 Then
        MgMineralName = "OPX"
        Limit01 = 0.99
        Limit02 = 0.9
        Limit03 = 0.72
        nMx = 1
    ElseIf FaciesEnum = FormMain.enumFacies.facies10AMP700 Then
        MgMineralName = "talc"
        Limit01 = 0.94
        Limit02 = 0.79
        Limit03 = 0.47
        nMx = 2
    ElseIf FaciesEnum = FormMain.enumFacies.facies4AMP600 Then
        MgMineralName = "talc"
```

```
    Limit01 = 0.92
    Limit02 = 0.67
    Limit03 = 0.35
    nMx = 2
ElseIf FaciesEnum = FormMain.enumFacies.facies6GRA750 Then
    MgMineralName = "talc"
    Limit01 = 0.99
    Limit02 = 0.97
    Limit03 = 0.61
    nMx = 1
ElseIf FaciesEnum = FormMain.enumFacies.facies6AMP600 Then
    MgMineralName = "talc"
    Limit01 = 0.9
    Limit02 = 0.64
    Limit03 = 0.35
    nMx = 2
End If

If PxmnThere = True And MgFeRatio > Limit04 And Limit04 <> 0 Then
    mole("talc").NBmole = (FePercent + MnPercent) * FMMprop
    mole(MnMineralName).NBmole = MnPercent * FMMprop
Else
    If MgMnRatio < Limit03 Then
        mole(MnMineralName).NBmole = MnPercent * FMMprop
    Else
        FePercent2 = FePercent + MnPercent
    End If

    If MgFeRatio > Limit01 Then
        mole(MgMineralName).NBmole = (MgPercent + FePercent2) * FMMprop
    ElseIf MgFeRatio > Limit02 Then
        mole(MgMineralName).NBmole = MgPercent * FMMprop
        mole("olivine").NBmole = FePercent2 * FMMprop
    Else
        If nMx = 1 Then
            mole("olivine").NBmole = (MgPercent + FePercent2) * FMMprop
        ElseIf nMx = 2 Then
```

```
                        mole("olivine").NBmole = FePercent2 * FMMprop
                        mole("anthophyllite").NBmole = MgPercent * FMMprop
                    End If
                End If
            End If
        End If

    End Sub

    Sub FManorthiteReaction()

        'FM + anorthite = tschermakite (only for amphibolite facies)
        Dim TSCHprop As Double = mole("tschermakite").NBmole
        Dim ANprop As Double = mole("anorthite").NBmole
        Dim FMprop As Double = mole("FM").NBmole
        Dim Factor As Double = 1       'Factor: used to destroy only a certain % of An/FM (to be adjusted using natural cases) / the
higher, the more An-Fm are destroyed

        If FMprop > 0.5 * ANprop Then
            TSCHprop = TSCHprop + (ANprop * Factor) * 1.5
            FMprop = FMprop - 0.5 * (ANprop * Factor)
            ANprop = ANprop * (1 - Factor)
        Else
            TSCHprop = TSCHprop + (FMprop * Factor) * 3
            ANprop = ANprop - 2 * (FMprop * Factor)
            FMprop = FMprop * (1 - Factor)
        End If

        mole("FM").NBmole = FMprop
        mole("anorthite").NBmole = ANprop
        mole("tschermakite").NBmole = TSCHprop
    End Sub

    Sub PargasiteCreation01()

        Dim FMprop As Double = mole("FM").NBmole
        Dim ANprop As Double = mole("anorthite").NBmole
```

```
    Dim ALBprop As Double = mole("albite").NBmole
    Dim TRprop As Double = mole("actinolite").NBmole

    Dim ParagProp As Double = 0

    If FMprop * (3 / 1.5) < ANprop And FMprop * (2 / 1.5) < ALBprop And FMprop * (3.5 / 1.5) < TRprop Then
        ParagProp = FMprop * (10 / 1.5)
        ANprop = ANprop - ParagProp * (3 / 10)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        TRprop = TRprop - ParagProp * (3.5 / 10)
        FMprop = 0
    ElseIf TRprop * (1.5 / 3.5) < FMprop And TRprop * (2 / 3.5) < ALBprop And TRprop * (3 / 3.5) < ANprop Then
        ParagProp = TRprop * (10 / 3.5)
        ANprop = ANprop - ParagProp * (3 / 10)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        FMprop = FMprop - ParagProp * (1.5 / 10)
        TRprop = 0
    ElseIf ALBprop * (3 / 2) < ANprop And ALBprop * (1.5 / 2) < FMprop And ALBprop * (3.5 / 2) < TRprop Then
        ParagProp = ALBprop * (10 / 2)
        ANprop = ANprop - ParagProp * (3 / 10)
        FMprop = FMprop - ParagProp * (1.5 / 10)
        TRprop = TRprop - ParagProp * (3.5 / 10)
        ALBprop = 0
    Else           'anorthite out because it is the limiting factor
        ParagProp = ANprop * (10 / 3)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        TRprop = TRprop - ParagProp * (3.5 / 10)
        FMprop = FMprop - ParagProp * (1.5 / 10)
        ANprop = 0
    End If

    mole("FM").NBmole = FMprop
    mole("albite").NBmole = ALBprop
    mole("anorthite").NBmole = ANprop
    mole("actinolite").NBmole = TRprop
    mole("pargasite").NBmole = ParagProp
End Sub
```

```
Sub PargasiteCreation02()
    Dim CPXprop As Double = mole("CPX").NBmole
    Dim ANprop As Double = mole("anorthite").NBmole
    Dim ALBprop As Double = mole("albite").NBmole
    Dim TRprop As Double = mole("actinolite").NBmole

    Dim ParagProp As Double = 0

    If TRprop * (3 / 7) < ANprop And TRprop * (2 / 7) < ALBprop Then
        ParagProp = TRprop * (10 / 5) * (5 / 7)
        ANprop = ANprop - ParagProp * (3 / 10)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        CPXprop = CPXprop + TRprop * (2 / 7)
        TRprop = 0
    ElseIf ANprop * (15 / 7) < TRprop And ANprop * (2 / 3) < ALBprop Then
        ParagProp = ANprop * (10 / 3)
        TRprop = TRprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (2 / 7)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        CPXprop = CPXprop + ParagProp * (5 / 10) * (2 / 7)
        ANprop = 0
    Else
        ParagProp = ALBprop * (10 / 2)
        TRprop = TRprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (2 / 7)
        ANprop = ANprop - ParagProp * (3 / 10)
        CPXprop = CPXprop + ParagProp * (5 / 10) * (2 / 7)
        ALBprop = 0
    End If

    mole("CPX").NBmole = CPXprop
    mole("albite").NBmole = ALBprop
    mole("anorthite").NBmole = ANprop
    mole("actinolite").NBmole = TRprop
    mole("pargasite").NBmole = ParagProp
End Sub

Sub PargasiteCreation03()
```

```
Dim CPXprop As Double = mole("CPX").NBmole
Dim ANprop As Double = mole("anorthite").NBmole
Dim ALBprop As Double = mole("albite").NBmole
Dim WOprop As Double = mole("wollastonite").NBmole

Dim ParagProp As Double = 0

If CPXprop * (3 / 8) < ANprop And CPXprop * (2 / 8) < ALBprop Then
    ParagProp = CPXprop * (10 / 5) * (5 / 8)
    ANprop = ANprop - ParagProp * (3 / 10)
    ALBprop = ALBprop - ParagProp * (2 / 10)
    WOprop = WOprop + CPXprop * (3 / 8)
    CPXprop = 0
ElseIf ANprop * (71 / 48) < CPXprop And ANprop * (2 / 3) < ALBprop Then
    ParagProp = ANprop * (10 / 3)
    CPXprop = CPXprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (3 / 8)
    ALBprop = ALBprop - ParagProp * (2 / 10)
    WOprop = WOprop + ParagProp * (5 / 10) * (3 / 8)
    ANprop = 0
Else
    ParagProp = ALBprop * (10 / 2)
    CPXprop = CPXprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (3 / 8)
    ANprop = ANprop - ParagProp * (3 / 10)
    WOprop = WOprop + ParagProp * (5 / 10) * (3 / 8)
    ALBprop = 0
End If

mole("CPX").NBmole = CPXprop
mole("albite").NBmole = ALBprop
mole("pargasite").NBmole = ParagProp

If mole("grossular").NBmole > 0 And ANprop > 0 And WOprop > 0 Then
    Dim GROSSprop As Double = mole("grossular").NBmole
    If ANprop > WOprop * (3 / 2) Then
        GROSSprop = GROSSprop + WOprop * (5 / 2)
        ANprop = ANprop - WOprop * (3 / 2)
        WOprop = 0
```

```
        Else
            GROSSprop = GROSSprop + ANprop * (5 / 3)
            WOprop = WOprop - ANprop * (2 / 3)
            ANprop = 0
        End If
        mole("grossular").NBmole = GROSSprop

    ElseIf mole("clinozoisite").NBmole > 0 And ANprop > 0 And WOprop > 0 Then
        Dim CZOprop As Double = mole("clinozoisite").NBmole
        If ANprop > WOprop * 9 Then
            CZOprop = CZOprop + WOprop * 10
            ANprop = ANprop - WOprop * 9
            WOprop = 0
        Else
            CZOprop = CZOprop + ANprop * (10 / 9)
            WOprop = WOprop - ANprop * (1 / 9)
            ANprop = 0
        End If
        mole("clinozoisite").NBmole = CZOprop
    End If

    mole("wollastonite").NBmole = WOprop
    mole("anorthite").NBmole = ANprop
End Sub

Sub PargasiteCreation04()
    Dim FMprop As Double = mole("FM").NBmole
    Dim ANprop As Double = mole("anorthite").NBmole
    Dim ALBprop As Double = mole("albite").NBmole

    Dim NameALU As String = WhichALUMINO()
    Dim ALUMprop As Double = mole(NameALU).NBmole

    Dim ParagProp As Double = 0

    If ANprop * (1 / 3) < ALBprop And ANprop * (5 / 12) < FMprop Then
        ParagProp = ANprop * (10 / 4) * (2 / 3)
```

```
    FMprop = FMprop - ParagProp * (4 / 10)
    ALBprop = ALBprop - ParagProp * (2 / 10)
    ALUMprop = ALUMprop + ANprop * (1 / 3)
    ANprop = 0
ElseIf FMprop * (1 / 2) < ALBprop And FMprop * (4 / 3) < ANprop Then
    ParagProp = FMprop * (10 / 4)
    ANprop = ANprop - ParagProp * (4 / 10) - ParagProp * (4 / 10) * (1 / 3)
    ALBprop = ALBprop - ParagProp * (2 / 10)
    ALUMprop = ALUMprop + ParagProp * (4 / 10) * (1 / 3)
    FMprop = 0
Else
    ParagProp = ALBprop * (10 / 2)
    ANprop = ANprop - ParagProp * (4 / 10) - ParagProp * (4 / 10) * (1 / 3)
    FMprop = FMprop - ParagProp * (4 / 10)
    ALUMprop = ALUMprop + ParagProp * (4 / 10) * (1 / 3)
    ALBprop = 0
End If

mole("albite").NBmole = ALBprop
mole("anorthite").NBmole = ANprop
mole("pargasite").NBmole = ParagProp

If mole("cordierite").NBmole > 0 And ALUMprop > 0 And FMprop > 0 Then
    Dim CORDprop As Double = mole("cordierite").NBmole

    If ALUMprop > FMprop * 2 Then
        ALUMprop = ALUMprop - FMprop * 2
        CORDprop = CORDprop + FMprop * (6 / 2)
        FMprop = 0
    Else
        FMprop = FMprop - ALUMprop / 2
        CORDprop = CORDprop + ALUMprop * (6 / 4)
        ALUMprop = 0
    End If
    mole("cordierite").NBmole = CORDprop

ElseIf mole("amesite").NBmole > 0 And ALUMprop > 0 And FMprop > 0 Then
```

```
        Dim AMEprop As Double = mole("amesite").NBmole

        If ALUMprop > FMprop Then
            ALUMprop = ALUMprop - FMprop
            AMEprop = AMEprop + FMprop * 2
            FMprop = 0
        Else
            FMprop = FMprop - ALUMprop
            AMEprop = AMEprop + ALUMprop * 2
            ALUMprop = 0
        End If
        mole("amesite").NBmole = AMEprop

    ElseIf mole("GRTpyalm").NBmole > 0 And ALUMprop > 0 And FMprop > 0 Then
        Dim GRTprop As Double = mole("GRTpyalm").NBmole

        If ALUMprop > FMprop * (5 / 3) Then
            ALUMprop = ALUMprop - FMprop * (2 / 3)
            GRTprop = GRTprop + FMprop * (5 / 3)
            FMprop = 0
        Else
            FMprop = FMprop - ALUMprop * (3 / 2)
            GRTprop = GRTprop + ALUMprop * (5 / 2)
            ALUMprop = 0
        End If
        mole("GRTpyalm").NBmole = GRTprop
    End If

    mole("FM").NBmole = FMprop
    mole(NameALU).NBmole = ALUMprop
End Sub

Sub PargasiteCreation05()
    Dim TRprop As Double = mole("actinolite").NBmole
    Dim CORDprop As Double = mole("cordierite").NBmole
    Dim ALBprop As Double = mole("albite").NBmole
    Dim FMprop As Double = mole("FM").NBmole
```

```
    Dim ParagProp As Double = mole("pargasite").NBmole

    If TRprop * (3 / 7) < CORDprop And TRprop * (2 / 7) < ALBprop Then
        ParagProp = ParagProp + TRprop * (10 / 5) * (5 / 7)
        CORDprop = CORDprop - ParagProp * (3 / 10)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        FMprop = FMprop + TRprop * (2 / 7)
        TRprop = 0
    ElseIf CORDprop * (15 / 7) < TRprop And CORDprop * (2 / 3) < ALBprop Then
        ParagProp = ParagProp + CORDprop * (10 / 3)
        TRprop = TRprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (2 / 7)
        ALBprop = ALBprop - ParagProp * (2 / 10)
        FMprop = FMprop + ParagProp * (5 / 10) * (2 / 7)
        CORDprop = 0
    Else
        ParagProp = ParagProp + ALBprop * (10 / 2)
        TRprop = TRprop - ParagProp * (5 / 10) - ParagProp * (5 / 10) * (2 / 7)
        CORDprop = CORDprop - ParagProp * (3 / 10)
        FMprop = FMprop + ParagProp * (5 / 10) * (2 / 7)
        ALBprop = 0
    End If

    mole("actinolite").NBmole = TRprop
    mole("cordierite").NBmole = CORDprop
    mole("albite").NBmole = ALBprop
    mole("FM").NBmole = FMprop
    mole("pargasite").NBmole = ParagProp

End Sub

Sub PargasiteCreation06()
    Dim TRprop As Double = mole("actinolite").NBmole
    Dim AMEprop As Double = mole("amesite").NBmole
    Dim ALBprop As Double = mole("albite").NBmole
    Dim FMprop As Double = mole("FM").NBmole
```

```
        Dim ParagProp As Double = mole("pargasite").NBmole

        If TRprop * (4 / 7) < AMEprop And TRprop * (2 / 7) < ALBprop Then
            ParagProp = ParagProp + TRprop * (10 / 4) * (4 / 7)
            AMEprop = AMEprop - ParagProp * (4 / 10)
            ALBprop = ALBprop - ParagProp * (2 / 10)
            FMprop = FMprop + TRprop * (3 / 7)
            TRprop = 0
        ElseIf AMEprop * (10 / 7) < TRprop And AMEprop * (1 / 2) < ALBprop Then
            ParagProp = ParagProp + AMEprop * (10 / 4)
            TRprop = TRprop - ParagProp * (4 / 10) - ParagProp * (4 / 10) * (3 / 7)
            ALBprop = ALBprop - ParagProp * (2 / 10)
            FMprop = FMprop + ParagProp * (4 / 10) * (3 / 7)
            AMEprop = 0
        Else
            ParagProp = ParagProp + ALBprop * (10 / 2)
            TRprop = TRprop - ParagProp * (4 / 10) - ParagProp * (4 / 10) * (3 / 7)
            AMEprop = AMEprop - ParagProp * (4 / 10)
            FMprop = FMprop + ParagProp * (4 / 10) * (3 / 7)
            ALBprop = 0
        End If

        mole("actinolite").NBmole = TRprop
        mole("amesite").NBmole = AMEprop
        mole("albite").NBmole = ALBprop
        mole("FM").NBmole = FMprop
        mole("pargasite").NBmole = ParagProp
    End Sub

#End Region

#Region "SiDeficit"

    'Reactions used to solve the SI-deficit

    Private Sub SiExcessDeficit()
```

```
        Dim nameAlum As String = WhichALUMINO()
        Dim nameALO As String = WhichALO()
        Dim nameMGO As String = WhichMGO()
        Dim nameMNO As String = WhichMNO()

        Try
            If mole("kyanite").NBmole > 0 Or mole("sillimanite").NBmole > 0 Or mole("pyrophyllite").NBmole > 0 Then            '01 -
Aluminosilicate
                ForONE(nameAlum, nameALO)
            End If

            If mole("cordierite").NBmole > 0 And SiToGain > 0 Then                                                  '02 - cordierite
                If FaciesEnum = FormMain.enumFacies.facies10GRA950 Or FaciesEnum = FormMain.enumFacies.facies9GRA800 Then
                    ForTWO("cordierite", "OPX", nameALO)
                ElseIf FaciesEnum = FormMain.enumFacies.facies3AMP675 Or FaciesEnum = FormMain.enumFacies.facies5AMP670 Or FaciesEnum
= FormMain.enumFacies.facies7AMP700 Then
                    ForTWO("cordierite", "amesite", nameALO)
                Else
                    ForONE("cordierite", "spinelss")
                End If
            End If

            If mole("carpholite").NBmole > 0 And SiToGain > 0 Then                                                '02 - carpholite
                ForTWO("carpholite", "amesite", nameALO)
            End If

            If mole("glaucophane").NBmole > 0 And SiToGain > 0 Then                                                '02 - glaucophane
                ResetMGratio()
                ForTWO("glaucophane", "albite", nameMGO, "FeO", MgFeRatio)
            End If

            If mole("orthose").NBmole > 0 And SiToGain > 0 Then                                                '03 - orthose
                ForONE("orthose", "leucite")
            End If

            If mole("albite").NBmole > 0 And SiToGain > 0 Then                                                '04 - albite
```

```
                If FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum =
FormMain.enumFacies.facies9SB450 Or FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                    ForONE("albite", "jadeite")
                Else
                    ForONE("albite", "nepheline")
                End If
            End If

            If mole("leucite").NBmole > 0 And SiToGain > 0 Then                                          '05 - leucite
                ForONE("leucite", "kalsilite")
            End If

            If mole("WhiteMica").NBmole > 0 And SiToGain > 0 Then                                        '06 - WhiteMica
                destroyWhiteMica(nameALO)
            End If

            If mole("biotite").NBmole > 0 And SiToGain > 0 Then                                          '06 - biotite
                ResetMGratio()
                ForTWO("biotite", "kalsilite", nameMGO, "FeO", MgFeRatio)
            End If

            If mole("jadeite").NBmole > 0 And SiToGain > 0 Then                                          '07 - jadeite
                ForONE("jadeite", "nepheline")
            End If

            If mole("lawsonite").NBmole > 0 And SiToGain > 0 Then                                        '08 - lawsonite
                ForTWO("lawsonite", "grossular", nameALO)
            End If

            If mole("anorthite").NBmole > 0 And SiToGain > 0 Then                                      '08 - anorthite
                If FaciesEnum = FormMain.enumFacies.facies7AMP600 Or FaciesEnum = FormMain.enumFacies.facies10AMP700 Or FaciesEnum =
FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum = FormMain.enumFacies.facies9SB450
Or FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                    ForTHREE("anorthite", "grossular", "margarite", nameALO)
                Else
                    ForTWO("anorthite", "grossular", nameALO)
                End If
```

```
            End If

            If mole("margarite").NBmole > 0 And SiToGain > 0 Then                                   '10 - margarite
                ForTWO("margarite", "clinozoisite", nameALO)
            End If

            If mole("talc").NBmole > 0 And SiToGain > 0 Then                                        '11 - talc
                If FaciesEnum = FormMain.enumFacies.facies10AMP700 Then
                    ForONE("talc", "OPX")
                ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies4AMP600 Or FaciesEnum
= FormMain.enumFacies.facies5SV400 Or FaciesEnum = FormMain.enumFacies.facies9SB450 Or FaciesEnum = FormMain.enumFacies.facies10SB350
Then
                    ForONE("talc", "antigorite")
                Else
                    ForONE("talc", "anthophyllite")
                End If
            End If

            If mole("anthophyllite").NBmole > 0 And SiToGain > 0 Then                               '12 - anthophyllite
                ForONE("anthophyllite", "OPX")
            End If

            If mole("actinolite").NBmole > 0 And SiToGain > 0 Then                                   '12 - tremolite
                If FaciesEnum = FormMain.enumFacies.facies5GRA900 Or FaciesEnum = FormMain.enumFacies.facies10GRA950 Then
                    ForTWO("actinolite", "OPX", "CPX")
                ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum = FormMain.enumFacies.facies5SV400 Or FaciesEnum
= FormMain.enumFacies.facies9SB450 Or FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                    ForTWO("actinolite", "CPX", "antigorite")
                Else
                    ForTWO("actinolite", "olivine", "CPX")
                End If
            End If

            If mole("CPX").NBmole > 0 And SiToGain > 0 Then                                         '09 - CPX
                destroyCPX(nameMGO)
            End If
```

```
            If mole("tschermakite").NBmole > 0 And SiToGain > 0 Then                                    '12 - tschermakite
                ForTHREE("tschermakite", "grossular", "penninite", nameALO)
            End If

            If mole("epidote").NBmole > 0 And SiToGain > 0 Then                                         '13 - epidote
                ForTHREE("epidote", "grossular", "penninite", nameALO)
            End If

            If mole("staurolite").NBmole > 0 And SiToGain > 0 Then                                       '14 - staurolite
                If FaciesEnum = FormMain.enumFacies.facies4AMP600 Or FaciesEnum = FormMain.enumFacies.facies7AMP600 Or FaciesEnum =
FormMain.enumFacies.facies6AMP600 Then
                    ForTWO("staurolite", "spinelss", nameALO)
                ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Then
                    ForTWO("staurolite", "chloritoid", nameALO)
                Else
                    ForTWO("staurolite", "GRTpyalm", nameALO)
                End If
            End If

            If mole("clinozoisite").NBmole > 0 And SiToGain > 0 Then                                    '15 - clinozoisite
                ForTWO("clinozoisite", "grossular", nameALO)
            End If

            If mole("antigorite").NBmole > 0 And SiToGain > 0 Then                                      '16 - antigorite
                If FaciesEnum = FormMain.enumFacies.facies4AMP600 Or FaciesEnum = FormMain.enumFacies.facies7AMP500 Then
                    ForONE("antigorite", "olivine")
                Else
                    ResetMGratio()
                    ForONE("antigorite", nameMGO, "FeO", MgFeRatio)
                End If
            End If

            If mole("akermanite").NBmole > 0 And SiToGain > 0 Then                                      '17 - akermanite
                ForTWO("akermanite", "merwinite", "monticelliteSS")
            End If

            If mole("OPX").NBmole > 0 And SiToGain > 0 Then                                             '18 - OPX
```

```
            ForONE("OPX", "olivine")
        End If

        If mole("grossular").NBmole > 0 And SiToGain > 0 Then                                   '19 - grossular
            ForTWO("grossular", "gehlenite", "wollastonite")
        End If

        If mole("GRTpyalm").NBmole > 0 And SiToGain > 0 Then                                     '19 - GRTpyalm,
spessartine
            If FeMgMnRatio < 0.9 Then                          'Provisional pyrope-almandine-Spessartine amount
                Dim propGRT As Double = mole("GRTpyalm").NBmole
                mole("GRTpyalm").NBmole = propGRT * FeMgMnRatio
                mole("spessartine").NBmole = propGRT * (1 - FeMgMnRatio)
            End If
            ForTWO("GRTpyalm", "spinelss", "olivine")
        End If

        If mole("spessartine").NBmole > 0 And SiToGain > 0 Then
            ForTWO("spessartine", nameALO, nameMNO)
        End If
        If mole("spessartine").NBmole > 0 Then     'Remove provisional Spessartine
            mole("GRTpyalm").NBmole = mole("GRTpyalm").NBmole + mole("spessartine").NBmole
            mole("spessartine").NBmole = 0
        End If

        If mole("chloritoid").NBmole > 0 And SiToGain > 0 Then                                  '20 - chloritoide
            If FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                ForTWO("chloritoid", "amesite", nameALO)
            Else
                ForONE("chloritoid", "spinelss")
            End If
        End If

        If mole("daphnite").NBmole > 0 And SiToGain > 0 Then                                    '22 - chlorite = DAPH
            ForTWO("daphnite", "olivine", "spinelss")
        End If
```

```
            If mole("sudoite").NBmole > 0 And SiToGain > 0 Then                                    '22 - chlorite = SUD
                ForTWO("sudoite", "amesite", nameALO)
            End If

            If mole("amesite").NBmole > 0 And SiToGain > 0 Then                                    '22 - chlorite = AME
                If FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                    ForTWO("amesite", nameMGO, nameALO)
                Else
                    If FaciesEnum = FormMain.enumFacies.facies7AMP600 Or FaciesEnum = FormMain.enumFacies.facies7AMP500 Or FaciesEnum
= FormMain.enumFacies.facies4AMP600 Or FaciesEnum = FormMain.enumFacies.facies9SB450 Or FaciesEnum =
FormMain.enumFacies.facies6AMP600 Then
                        ForTWO("amesite", "spinelss", nameMGO)
                    Else
                        ForTWO("amesite", "spinelss", "olivine")
                    End If
                End If
            End If

            If mole("penninite").NBmole > 0 And SiToGain > 0 Then                                  '22 - chlorite = PEN
                ForTWO("penninite", "olivine", "spinelss")
            End If

            If mole("monticelliteSS").NBmole > 0 And SiToGain > 0 Then                               '23 - Monticellite
                ResetMGratio()
                ForTWO("monticelliteSS", "merwinite", nameMGO, "FeO", MgFeRatio)
                mole(nameMGO).NBmole = mole(nameMGO).NBmole - 0.25 * mole("merwinite").NBmole
                mole("FeO").NBmole = mole("FeO").NBmole + 0.25 * mole("merwinite").NBmole
            End If

            If mole("gehlenite").NBmole > 0 And SiToGain > 0 Then                                  '24 - gehlenite
                ForTWO("gehlenite", "lime", nameALO)
            End If

            If mole("merwinite").NBmole > 0 And SiToGain > 0 Then                                  '25 - merwinite
                ForTWO("merwinite", "lime", nameMGO)
            End If
```

```
            If mole("olivine").NBmole > 0 And SiToGain > 0 Then                                         '26 - olivine
                ForTWO("olivine", "FeO", nameMGO)
            End If

            If mole("wollastonite").NBmole > 0 And SiToGain > 0 Then                                    '27 - wollastonite
                ForONE("wollastonite", "lime")
            End If

            If mole("rhodonite").NBmole > 0 And SiToGain > 0 Or mole("pyroxmangite").NBmole > 0 And SiToGain > 0 Then  '28 -
pyroxmangite, rhodonite
                ForONE(nameMNO, "tephroite")
            End If

            If mole("tephroite").NBmole > 0 And SiToGain > 0 Then                                   '29 - tephroite
                ForONE("tephroite", "MnO")
            End If

        Catch ex As Exception
            ' MsgBox(ex.Message & "Error in solving Si deficit", MsgBoxStyle.Critical)
        End Try
    End Sub


    Private Sub destroyWhiteMica(ByVal nameALO As String)

        'Provisional muscovite-paragonite
        Dim WMratio As Double = 0
        If KRatioForWM = 0 Then : WMratio = Kratio
        Else : WMratio = KRatioForWM
        End If
        mole("muscovite").NBmole = mole("WhiteMica").NBmole * WMratio
        mole("paragonite").NBmole = mole("WhiteMica").NBmole * (1 - WMratio)
        mole("WhiteMica").NBmole = 0

        ForTWO("muscovite", "kalsilite", nameALO)

        If FaciesEnum = FormMain.enumFacies.facies10SB350 Then
```

```
            ForTWO("paragonite", "jadeite", nameALO)
        Else
            ForTWO("paragonite", "nepheline", nameALO)
        End If

        'Remove provisional muscovite-paragonite
        mole("WhiteMica").NBmole = mole("muscovite").NBmole + mole("paragonite").NBmole
        mole("muscovite").NBmole = 0
        mole("paragonite").NBmole = 0
    End Sub

    Private Sub destroyCPX(ByVal nameMGO As String)           '09 - CPX

        ResetMGratio()

        If mole("wollastonite").NBmole > 0 Then  'Diopside + Wollastonite + Hedenbergite = Merwinite OR Akermanite + (Wo) + fayalite

            'Provisional diopside-hedenbergite
            mole("diopside").NBmole = mole("CPX").NBmole * MgFeRatio
            mole("hedenbergite").NBmole = mole("CPX").NBmole * (1 - MgFeRatio)
            mole("CPX").NBmole = 0

            ForTWO("hedenbergite", "olivine", "wollastonite")

            If FaciesEnum = FormMain.enumFacies.facies3GRA800 Or FaciesEnum = FormMain.enumFacies.facies5GRA900 Then
                ForTWO("diopside", "akermanite", "wollastonite")
            Else
                ForTWO("diopside", "merwinite", "wollastonite")
            End If

            'Remove provisional diopside-hedenbergite
            mole("CPX").NBmole = mole("hedenbergite").NBmole + mole("diopside").NBmole
            mole("diopside").NBmole = 0
            mole("hedenbergite").NBmole = 0

        Else
            If MgFeRatio < 0.1 Then
```

```
                ForTWO("CPX", "olivine", "wollastonite")
            Else
                If FaciesEnum = FormMain.enumFacies.facies10SB350 Then
                    ForTWO("CPX", "wollastonite", nameMGO, "FeO", MgFeRatio)
                    ResetMGratio()
                Else
                    ForONE("CPX", "monticelliteSS")
                End If
            End If
        End If
    End Sub

    Private Sub ForONE(ByVal Mx1 As String, ByVal Mx2 As String, Optional ByVal Mx3Fe As String = "", Optional ByVal MgRatio As
Double = 1)

        Dim cationMx1 As New Dictionary(Of String, Double)
        cationMx1 = SetCationDataForTetraedre(Mx1, "Silicate")
        Dim cationMx2 As New Dictionary(Of String, Double)
        cationMx2 = SetCationDataForTetraedre(Mx2, "Silicate")

        Dim SiInMx1 As Double = cationMx1("Si") / cationMx1("SUM")
        Dim SiInMx2 As Double = cationMx2("Si") / cationMx2("SUM")

        Dim SiGainedMax As Double = mole(Mx1).NBmole * (SiInMx1 - SiInMx2)

        If SiGainedMax <= SiToGain Then
            mole(Mx2).NBmole = mole(Mx2).NBmole + mole(Mx1).NBmole * MgRatio
            If Mx3Fe <> "" Then
                mole(Mx3Fe).NBmole = mole(Mx3Fe).NBmole + mole(Mx1).NBmole * (1 - MgRatio)
            End If
            mole(Mx1).NBmole = 0
            SiToGain = SiToGain - SiGainedMax
        Else
            mole(Mx1).NBmole = mole(Mx1).NBmole - SiToGain * (1 / (SiInMx1 - SiInMx2))
            mole(Mx2).NBmole = mole(Mx2).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx2)) * MgRatio
            If Mx3Fe <> "" Then
                mole(Mx3Fe).NBmole = mole(Mx3Fe).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx2)) * (1 - MgRatio)
```

```
            End If
            SiToGain = 0
        End If

    End Sub

    Private Sub ForTWO(ByVal Mx1 As String, ByVal Mx2 As String, ByVal Mx3Mg As String, Optional ByVal Mx4Fe As String = "", Optional
ByVal MgRatio As Double = 1)

        Dim cationMx1 As New Dictionary(Of String, Double)
        cationMx1 = SetCationDataForTetraedre(Mx1, "Silicate")
        Dim cationMx2 As New Dictionary(Of String, Double)
        cationMx2 = SetCationDataForTetraedre(Mx2, "Silicate")
        Dim cationMx3 As New Dictionary(Of String, Double)
        cationMx3 = SetCationDataForTetraedre(Mx3Mg, "Silicate")

        Dim SiInMx1 As Double = cationMx1("Si") / cationMx1("SUM")
        Dim SiInMx23 As Double = (cationMx2("Si") + cationMx3("Si")) / (cationMx2("SUM") + cationMx3("SUM"))
        Dim Factor1 As Double = cationMx2("SUM") / (cationMx2("SUM") + cationMx3("SUM"))

        Dim SiGainedMax As Double = mole(Mx1).NBmole * (SiInMx1 - SiInMx23)

        If SiGainedMax <= SiToGain Then
            mole(Mx2).NBmole = mole(Mx2).NBmole + mole(Mx1).NBmole * Factor1
            mole(Mx3Mg).NBmole = mole(Mx3Mg).NBmole + mole(Mx1).NBmole * (1 - Factor1) * MgRatio
            If Mx4Fe <> "" Then
                mole(Mx4Fe).NBmole = mole(Mx4Fe).NBmole + mole(Mx1).NBmole * (1 - Factor1) * (1 - MgRatio)
            End If
            mole(Mx1).NBmole = 0
            SiToGain = SiToGain - SiGainedMax
        Else
            mole(Mx1).NBmole = mole(Mx1).NBmole - SiToGain * (1 / (SiInMx1 - SiInMx23))
            mole(Mx2).NBmole = mole(Mx2).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx23)) * Factor1
            mole(Mx3Mg).NBmole = mole(Mx3Mg).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx23)) * (1 - Factor1) * MgRatio
            If Mx4Fe <> "" Then
                mole(Mx4Fe).NBmole = mole(Mx4Fe).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx23)) * (1 - Factor1) * (1 - MgRatio)
            End If
```

```
            SiToGain = 0
        End If

    End Sub

    Private Sub ForTHREE(ByVal Mx1 As String, ByVal Mx2 As String, ByVal Mx3 As String, ByVal Mx4 As String)

        Dim cationMx1 As New Dictionary(Of String, Double)
        cationMx1 = SetCationDataForTetraedre(Mx1, "Silicate")
        Dim cationMx2 As New Dictionary(Of String, Double)
        cationMx2 = SetCationDataForTetraedre(Mx2, "Silicate")
        Dim cationMx3 As New Dictionary(Of String, Double)
        cationMx3 = SetCationDataForTetraedre(Mx3, "Silicate")
        Dim cationMx4 As New Dictionary(Of String, Double)
        cationMx4 = SetCationDataForTetraedre(Mx4, "Silicate")

        Dim SiInMx1 As Double = cationMx1("Si") / cationMx1("SUM")
        Dim SiInMx234 As Double = (cationMx2("Si") + cationMx3("Si") + cationMx4("Si")) / (cationMx2("SUM") + cationMx3("SUM") +
cationMx4("SUM"))
        Dim Factor1 As Double = cationMx2("SUM") / (cationMx2("SUM") + cationMx3("SUM") + cationMx4("SUM"))
        Dim Factor2 As Double = cationMx3("SUM") / (cationMx2("SUM") + cationMx3("SUM") + cationMx4("SUM"))

        Dim SiGainedMax As Double = mole(Mx1).NBmole * (SiInMx1 - SiInMx234)

        If SiGainedMax <= SiToGain Then
            mole(Mx2).NBmole = mole(Mx2).NBmole + mole(Mx1).NBmole * Factor1
            mole(Mx3).NBmole = mole(Mx3).NBmole + mole(Mx1).NBmole * Factor2
            mole(Mx4).NBmole = mole(Mx4).NBmole + mole(Mx1).NBmole * (1 - Factor1 - Factor2)
            mole(Mx1).NBmole = 0
            SiToGain = SiToGain - SiGainedMax
        Else
            mole(Mx1).NBmole = mole(Mx1).NBmole - SiToGain * (1 / (SiInMx1 - SiInMx234))
            mole(Mx2).NBmole = mole(Mx2).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx234)) * Factor1
            mole(Mx3).NBmole = mole(Mx3).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx234)) * Factor2
            mole(Mx4).NBmole = mole(Mx4).NBmole + SiToGain * (1 / (SiInMx1 - SiInMx234)) * (1 - Factor1 - Factor2)
            SiToGain = 0
        End If
```

```
    End Sub

#End Region


#Region "MineralsDefinition"

    'Simplified formula of minerals modelled.

    Public Function MineralsDefinition(ByVal mineral As String) As Dictionary(Of String, Double)
        Dim cation As New Dictionary(Of String, Double)

        If mineral = "apatiteH" Then
            cation.Add("Ca", 5)
            cation.Add("P", 3)
            cation.Add("O", 13)
            cation.Add("H", 1)
        ElseIf mineral = "apatiteF" Then
            cation.Add("Ca", 5)
            cation.Add("P", 3)
            cation.Add("O", 12)
            cation.Add("F", 1)
        ElseIf mineral = "apatiteCl" Then
            cation.Add("Ca", 5)
            cation.Add("P", 3)
            cation.Add("O", 12)
            cation.Add("Cl", 1)
        ElseIf mineral = "chromite" Then
            cation.Add("Fe2", 1)
            cation.Add("Cr", 2)
            cation.Add("O", 4)
        ElseIf mineral = "zircon" Then
            cation.Add("Zr", 1)
            cation.Add("Si", 1)
            cation.Add("O", 4)
        ElseIf mineral = "schorl" Then
            cation.Add("Fe2", 3)
```

```
    cation.Add("Na", 1)
    cation.Add("Al", 6)
    cation.Add("B", 3)
    cation.Add("H", 4)
    cation.Add("Si", 6)
    cation.Add("O", 31)
ElseIf mineral = "dravite" Then
    cation.Add("Mg", 3)
    cation.Add("Na", 1)
    cation.Add("Al", 6)
    cation.Add("B", 3)
    cation.Add("H", 4)
    cation.Add("Si", 6)
    cation.Add("O", 31)
ElseIf mineral = "foitite" Then
    cation.Add("Fe2", 3)
    cation.Add("Al", 6)
    cation.Add("B", 3)
    cation.Add("H", 4)
    cation.Add("Si", 6)
    cation.Add("O", 31)
ElseIf mineral = "foititeMg" Then
    cation.Add("Mg", 3)
    cation.Add("Al", 6)
    cation.Add("B", 3)
    cation.Add("H", 4)
    cation.Add("Si", 6)
    cation.Add("O", 31)
ElseIf mineral = "barite" Then
    cation.Add("Ba", 1)
    cation.Add("S", 1)
    cation.Add("O", 4)
ElseIf mineral = "pyrite" Then
    cation.Add("Fe2", 1)
    cation.Add("S", 2)
ElseIf mineral = "pyrrhotite" Then
    cation.Add("Fe2", 1)
```

```
    cation.Add("S", 1)
ElseIf mineral = "galena" Then
    cation.Add("Pb", 1)
    cation.Add("S", 1)
ElseIf mineral = "sphalerite" Then
    cation.Add("Zn", 1)
    cation.Add("S", 1)
ElseIf mineral = "millerite" Then
    cation.Add("Ni", 1)
    cation.Add("S", 1)
ElseIf mineral = "molybdenite" Then
    cation.Add("Mo", 1)
    cation.Add("S", 2)
ElseIf mineral = "chalcopyrite" Then
    cation.Add("Fe2", 1)
    cation.Add("Cu", 1)
    cation.Add("S", 2)
ElseIf mineral = "arsenopyrite" Then
    cation.Add("Fe2", 1)
    cation.Add("As", 1)
    cation.Add("S", 1)

ElseIf mineral = "albite" Then
    cation.Add("Na", 1)
    cation.Add("Al", 1)
    cation.Add("Si", 3)
    cation.Add("O", 8)
ElseIf mineral = "anorthite" Then
    cation.Add("Ca", 1)
    cation.Add("Al", 2)
    cation.Add("Si", 2)
    cation.Add("O", 8)
ElseIf mineral = "orthose" Then
    cation.Add("K", 1)
    cation.Add("Al", 1)
    cation.Add("Si", 3)
    cation.Add("O", 8)
```

```
ElseIf mineral = "Kfeld" Then
    cation.Add("KNa", 1)
    cation.Add("Al", 1)
    cation.Add("Si", 3)
    cation.Add("O", 8)
ElseIf mineral.Contains("FSP0") Or mineral.Contains("FSP1") Then : Dim nb As Double = 0
    If mineral = "FSP04" Then : nb = 0.4 + Kratio * 0.6 - 0.001
    ElseIf mineral = "FSP05_B" Then : nb = 0.05 + Kratio * 0.95 - 0.001
    ElseIf mineral = "FSP05_A" Then : nb = 0.7 + Kratio * 0.3
    ElseIf mineral = "FSP06" Then : nb = 0.55 + Kratio * 0.45 - 0.001
    ElseIf mineral = "FSP07" Then : nb = 0.1 + Kratio * 0.9 - 0.001
    ElseIf mineral = "FSP08" Then : nb = 0.5 + Kratio * 0.5 - 0.001
    ElseIf mineral = "FSP09" Then : nb = 0.5 + Kratio * 0.5 - 0.001
    ElseIf mineral = "FSP11" Then : nb = 0.7 + Kratio * 0.3
    ElseIf mineral = "FSP17" Then : nb = 0.5 + Kratio * 0.5 - 0.001
    End If
    cation.Add("Al", (2 - nb) / (3 - nb))
    cation.Add("Ca", (1 - nb) / (3 - nb))
    cation.Add("K", (nb / (3 - nb)) * Kratio)
    cation.Add("Na", (nb / (3 - nb)) * (1 - Kratio))
ElseIf mineral = "amesite" Then
    cation.Add("Al", 4)
    cation.Add("MgFe", 4)
    cation.Add("Si", 2)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "amesiteMg" Then
    cation.Add("Al", 4)
    cation.Add("Mg", 4)
    cation.Add("Si", 2)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "amesiteFE" Then
    cation.Add("Al", 4)
    cation.Add("Fe2", 4)
    cation.Add("Si", 2)
    cation.Add("H", 8)
```

```
    cation.Add("O", 18)
ElseIf mineral = "daphnite" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 5)
    cation.Add("Si", 3)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "daphniteFE" Then
    cation.Add("Al", 2)
    cation.Add("Fe2", 5)
    cation.Add("Si", 3)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "daphniteMg" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 5)
    cation.Add("Si", 3)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "sudoite" Then
    cation.Add("Al", 4)
    cation.Add("Mg", 2)
    cation.Add("Si", 3)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "penninite" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 11)
    cation.Add("Si", 7)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "penniniteMg" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 11)
    cation.Add("Si", 7)
    cation.Add("H", 8)
    cation.Add("O", 18)
```

```
ElseIf mineral = "penniniteFE" Then
    cation.Add("Al", 2)
    cation.Add("Fe2", 11)
    cation.Add("Si", 7)
    cation.Add("H", 8)
    cation.Add("O", 18)
ElseIf mineral = "OPX" Then
    cation.Add("MgFe", 2)
    cation.Add("Si", 2)
    cation.Add("O", 6)
ElseIf mineral = "OPXtalc" Then
    cation.Add("MgFe", 2)
    cation.Add("Si", 2)
    cation.Add("O", 6)
ElseIf mineral = "enstatite" Then
    cation.Add("Mg", 2)
    cation.Add("Si", 2)
    cation.Add("O", 6)
ElseIf mineral = "ferrosilite" Then
    cation.Add("Fe2", 2)
    cation.Add("Si", 2)
    cation.Add("O", 6)
ElseIf mineral = "biotite" Then
    cation.Add("Al", 1)
    cation.Add("K", 1)
    cation.Add("MgFe", 3)
    cation.Add("Si", 3)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "annite" Then
    cation.Add("Al", 1)
    cation.Add("K", 1)
    cation.Add("Fe2", 3)
    cation.Add("Si", 3)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "phlogopite" Then
```

```
        cation.Add("Al", 1)
        cation.Add("K", 1)
        cation.Add("Mg", 3)
        cation.Add("Si", 3)
        cation.Add("H", 2)
        cation.Add("O", 12)
    ElseIf mineral = "diopside" Then
        cation.Add("Ca", 1)
        cation.Add("Mg", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "hedenbergite" Then
        cation.Add("Ca", 1)
        cation.Add("Fe2", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "CPX" Then
        cation.Add("Ca", 1)
        cation.Add("MgFe", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "CPXgrt" Then
        cation.Add("Ca", 1)
        cation.Add("MgFe", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "jadeite" Then
        cation.Add("Al", 1)
        cation.Add("Na", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "aegirine" Then
        cation.Add("Na", 1)
        cation.Add("Fe3", 1)
        cation.Add("Si", 2)
        cation.Add("O", 6)
    ElseIf mineral = "olivine" Then
```

```
    cation.Add("MgFe", 2)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "forsterite" Then
    cation.Add("Mg", 2)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "fayalite" Then
    cation.Add("Fe2", 2)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "grossular" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "spessartine" Then
    cation.Add("Al", 2)
    cation.Add("Mn", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "GRTpyalm" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "GRTss" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "GRTss05" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "almandine" Then
```

```
    cation.Add("Al", 2)
    cation.Add("Fe2", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "pyrope" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 3)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "GRThalf" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 1)
    cation.Add("MgFe", 2)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "GROSSss" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 2)
    cation.Add("MgFe", 1)
    cation.Add("Si", 3)
    cation.Add("O", 12)
ElseIf mineral = "anthophyllite" Then
    cation.Add("MgFe", 7)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "anthophylliteMg" Then
    cation.Add("Mg", 7)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "anthophylliteFE" Then
    cation.Add("Fe2", 7)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "actinolite" Then
```

```
    cation.Add("Ca", 2)
    cation.Add("MgFe", 5)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "tremoliteMg" Then
    cation.Add("Ca", 2)
    cation.Add("Mg", 5)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "tremoliteFE" Then
    cation.Add("Ca", 2)
    cation.Add("Fe2", 5)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "tschermakite" Then
    cation.Add("Al", 4)
    cation.Add("Ca", 2)
    cation.Add("MgFe", 3)
    cation.Add("Si", 6)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "tschermakiteMg" Then
    cation.Add("Al", 4)
    cation.Add("Ca", 2)
    cation.Add("Mg", 3)
    cation.Add("Si", 6)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "tschermakiteFE" Then
    cation.Add("Al", 4)
    cation.Add("Ca", 2)
    cation.Add("Fe2", 3)
    cation.Add("Si", 6)
    cation.Add("H", 2)
```

```
    cation.Add("O", 24)
ElseIf mineral = "pargasite" Then
    cation.Add("Al", 3)
    cation.Add("Ca", 2)
    cation.Add("Na", 1)
    cation.Add("MgFe", 4)
    cation.Add("Si", 6)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "pargasiteMg" Then
    cation.Add("Al", 3)
    cation.Add("Ca", 2)
    cation.Add("Na", 1)
    cation.Add("Mg", 4)
    cation.Add("Si", 6)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "pargasiteFE" Then
    cation.Add("Al", 3)
    cation.Add("Ca", 2)
    cation.Add("Na", 1)
    cation.Add("Fe2", 4)
    cation.Add("Si", 6)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "glaucophane" Then
    cation.Add("Al", 2)
    cation.Add("Na", 2)
    cation.Add("MgFe", 3)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "glaucophaneMg" Then
    cation.Add("Al", 2)
    cation.Add("Na", 2)
    cation.Add("Mg", 3)
    cation.Add("Si", 8)
```

```
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "glaucophaneFE" Then
    cation.Add("Al", 2)
    cation.Add("Na", 2)
    cation.Add("Fe2", 3)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "riebeckite" Then
    cation.Add("Na", 2)
    cation.Add("Fe2", 5)
    cation.Add("Si", 8)
    cation.Add("H", 2)
    cation.Add("O", 24)
ElseIf mineral = "WhiteMica" Then
    cation.Add("Al", 3)
    cation.Add("KNa", 1)
    cation.Add("Si", 3)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "muscovite" Then
    cation.Add("Al", 3)
    cation.Add("K", 1)
    cation.Add("Si", 3)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "paragonite" Then
    cation.Add("Al", 3)
    cation.Add("Na", 1)
    cation.Add("Si", 3)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "margarite" Then
    cation.Add("Al", 4)
    cation.Add("Ca", 1)
    cation.Add("Si", 2)
```

```
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "leucite" Then
    cation.Add("Al", 1)
    cation.Add("K", 1)
    cation.Add("Si", 2)
    cation.Add("O", 6)
ElseIf mineral = "nepheline" Then
    cation.Add("Al", 1)
    cation.Add("Na", 1)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "kalsilite" Then
    cation.Add("Al", 1)
    cation.Add("K", 1)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "gehlenite" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 2)
    cation.Add("Si", 1)
    cation.Add("O", 7)
ElseIf mineral = "kyanite" Then
    cation.Add("Al", 2)
    cation.Add("Si", 1)
    cation.Add("O", 5)
ElseIf mineral = "sillimanite" Then
    cation.Add("Al", 2)
    cation.Add("Si", 1)
    cation.Add("O", 5)
ElseIf mineral = "staurolite" Then
    cation.Add("Al", 36)
    cation.Add("MgFe", 8)
    cation.Add("Si", 15)
    cation.Add("H", 8)
    cation.Add("O", 96)
ElseIf mineral = "stauroliteMg" Then
```

```
    cation.Add("Al", 36)
    cation.Add("Mg", 8)
    cation.Add("Si", 15)
    cation.Add("H", 8)
    cation.Add("O", 96)
ElseIf mineral = "staurolitеFE" Then
    cation.Add("Al", 36)
    cation.Add("Fe2", 8)
    cation.Add("Si", 15)
    cation.Add("H", 8)
    cation.Add("O", 96)
ElseIf mineral = "talc" Then
    cation.Add("Mg", 3)
    cation.Add("Si", 4)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "cordierite" Then
    cation.Add("Al", 4)
    cation.Add("MgFe", 2)
    cation.Add("Si", 5)
    cation.Add("O", 18)
ElseIf mineral = "cordieriteMg" Then
    cation.Add("Al", 4)
    cation.Add("Mg", 2)
    cation.Add("Si", 5)
    cation.Add("O", 18)
ElseIf mineral = "cordieriteFE" Then
    cation.Add("Al", 4)
    cation.Add("Fe2", 2)
    cation.Add("Si", 5)
    cation.Add("O", 18)
ElseIf mineral = "chloritoid" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 1)
    cation.Add("Si", 1)
    cation.Add("H", 2)
    cation.Add("O", 7)
```

```
ElseIf mineral = "chloritoidMg" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 1)
    cation.Add("Si", 1)
    cation.Add("H", 2)
    cation.Add("O", 7)
ElseIf mineral = "chloritoidFE" Then
    cation.Add("Al", 2)
    cation.Add("Fe2", 1)
    cation.Add("Si", 1)
    cation.Add("H", 2)
    cation.Add("O", 7)
ElseIf mineral = "wollastonite" Then
    cation.Add("Ca", 1)
    cation.Add("Si", 1)
    cation.Add("O", 3)
ElseIf mineral = "epidote" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 2)
    cation.Add("Fe2", 1)
    cation.Add("Si", 3)
    cation.Add("H", 1)
    cation.Add("O", 13)
ElseIf mineral = "clinozoisite" Then
    cation.Add("Al", 3)
    cation.Add("Ca", 2)
    cation.Add("Si", 3)
    cation.Add("H", 1)
    cation.Add("O", 13)
ElseIf mineral = "spinelss" Then
    cation.Add("Al", 2)
    cation.Add("MgFe", 1)
    cation.Add("O", 4)
ElseIf mineral = "spinel" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 1)
    cation.Add("O", 4)
```

```
ElseIf mineral = "hercynite" Then
    cation.Add("Al", 2)
    cation.Add("Fe2", 1)
    cation.Add("O", 4)
ElseIf mineral = "pyrophyllite" Then
    cation.Add("Al", 2)
    cation.Add("Si", 4)
    cation.Add("H", 2)
    cation.Add("O", 12)
ElseIf mineral = "carpholite" Then
    cation.Add("Al", 2)
    cation.Add("Mg", 1)
    cation.Add("Si", 2)
    cation.Add("H", 4)
    cation.Add("O", 10)
ElseIf mineral = "lawsonite" Then
    cation.Add("Al", 2)
    cation.Add("Ca", 1)
    cation.Add("Si", 2)
    cation.Add("H", 4)
    cation.Add("O", 10)
ElseIf mineral = "brucite" Then
    cation.Add("Mg", 1)
    cation.Add("H", 2)
    cation.Add("O", 2)
ElseIf mineral = "lime" Then
    cation.Add("Ca", 1)
    cation.Add("O", 1)
ElseIf mineral = "corundum" Then
    cation.Add("Al", 2)
    cation.Add("O", 3)
ElseIf mineral = "periclase" Then
    cation.Add("Mg", 1)
    cation.Add("O", 1)
ElseIf mineral = "merwinite" Then
    cation.Add("Ca", 3)
    cation.Add("Mg", 1)
```

```
    cation.Add("Si", 2)
    cation.Add("O", 8)
ElseIf mineral = "akermanite" Then
    cation.Add("Ca", 2)
    cation.Add("Mg", 1)
    cation.Add("Si", 2)
    cation.Add("O", 7)
ElseIf mineral = "monticelliteSS" Then
    cation.Add("Ca", 1)
    cation.Add("MgFe", 1)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "monticellite" Then
    cation.Add("Ca", 1)
    cation.Add("Mg", 1)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "kirschsteinite" Then
    cation.Add("Ca", 1)
    cation.Add("Fe2", 1)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "antigorite" Then
    cation.Add("MgFe", 3)
    cation.Add("Si", 2)
    cation.Add("H", 4)
    cation.Add("O", 9)
ElseIf mineral = "antigoriteMg" Then
    cation.Add("Mg", 3)
    cation.Add("Si", 2)
    cation.Add("H", 4)
    cation.Add("O", 9)
ElseIf mineral = "antigoriteFE" Then
    cation.Add("Fe2", 3)
    cation.Add("Si", 2)
    cation.Add("H", 4)
    cation.Add("O", 9)
```

```
ElseIf mineral = "quartz" Then
    cation.Add("Si", 1)
    cation.Add("O", 2)
ElseIf mineral = "pyroxmangite" Then
    cation.Add("Mn", 1)
    cation.Add("Si", 1)
    cation.Add("O", 3)
ElseIf mineral = "rhodonite" Then
    cation.Add("Mn", 1)
    cation.Add("Si", 1)
    cation.Add("O", 3)
ElseIf mineral = "tephroite" Then
    cation.Add("Mn", 2)
    cation.Add("Si", 1)
    cation.Add("O", 4)
ElseIf mineral = "diaspore" Then
    cation.Add("Al", 1)
    cation.Add("H", 1)
    cation.Add("O", 2)
ElseIf mineral = "FM" Then
    cation.Add("MgFe", 1)
ElseIf mineral = "NaK" Then
    cation.Add("KNa", 1)
ElseIf mineral = "Na2O" Then
    cation.Add("Na", 2)
    cation.Add("O", 1)
ElseIf mineral = "K2O" Then
    cation.Add("K", 2)
    cation.Add("O", 1)
ElseIf mineral = "FeO" Then
    cation.Add("Fe2", 1)
    cation.Add("O", 1)
ElseIf mineral = "MnO" Then
    cation.Add("Mn", 1)
    cation.Add("O", 1)
ElseIf mineral = "ilmenite" Then
    cation.Add("Ti", 1)
```

```
    cation.Add("Fe2", 1)
    cation.Add("O", 3)
ElseIf mineral = "rutile" Then
    cation.Add("Ti", 1)
    cation.Add("O", 2)
ElseIf mineral = "ulvospinel" Then
    cation.Add("Ti", 1)
    cation.Add("Fe2", 2)
    cation.Add("O", 4)
ElseIf mineral = "magnetite" Then
    cation.Add("Fe2", 1)
    cation.Add("Fe3", 2)
    cation.Add("O", 4)
ElseIf mineral = "hematite" Then
    cation.Add("Fe3", 2)
    cation.Add("O", 3)
ElseIf mineral = "pseudobrookite" Then
    cation.Add("Ti", 1)
    cation.Add("Fe3", 2)
    cation.Add("O", 5)
ElseIf mineral = "wustite" Then
    cation.Add("Fe2", 1)
    cation.Add("O", 1)
ElseIf mineral = "HEMilm01" Then
    cation.Add("Ti", 0.417)
    cation.Add("Fe2", 0.417)
    cation.Add("Fe3", 1.166)
    cation.Add("O", 3)
ElseIf mineral = "HEMilm06" Then
    cation.Add("Ti", 0.15)
    cation.Add("Fe2", 0.15)
    cation.Add("Fe3", 1.7)
    cation.Add("O", 3)
ElseIf mineral = "HEMilm07" Then
    cation.Add("Ti", 0.94)
    cation.Add("Fe2", 0.94)
    cation.Add("Fe3", 0.12)
```

```
    cation.Add("O", 3)
ElseIf mineral = "PSBfe01" Then
    cation.Add("Ti", 1.125)
    cation.Add("Fe2", 0.125)
    cation.Add("Fe3", 1.75)
    cation.Add("O", 5)
ElseIf mineral = "PSBfe02" Then
    cation.Add("Ti", 1.54)
    cation.Add("Fe2", 0.54)
    cation.Add("Fe3", 0.92)
    cation.Add("O", 5)
ElseIf mineral = "PSBfe03" Then
    cation.Add("Ti", 1.68)
    cation.Add("Fe2", 0.68)
    cation.Add("Fe3", 0.64)
    cation.Add("O", 5)
ElseIf mineral = "PSBfe06" Then
    cation.Add("Ti", 1.05)
    cation.Add("Fe2", 0.05)
    cation.Add("Fe3", 1.9)
    cation.Add("O", 5)
ElseIf mineral = "PSBfe07" Then
    cation.Add("Ti", 1.784)
    cation.Add("Fe2", 0.784)
    cation.Add("Fe3", 0.432)
    cation.Add("O", 5)
ElseIf mineral = "graphite" Then
    cation.Add("C", 1)
ElseIf mineral = "siderite" Then
    cation.Add("Fe2", 1)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "magnesite" Then
    cation.Add("Mg", 1)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "calcite" Then
```

```
    cation.Add("Ca", 1)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "dolomite" Then
    cation.Add("Ca", 1)
    cation.Add("Mg", 1)
    cation.Add("C", 2)
    cation.Add("O", 6)
ElseIf mineral = "ankerite" Then
    cation.Add("Ca", 1)
    cation.Add("MgFe", 1)
    cation.Add("C", 2)
    cation.Add("O", 6)
ElseIf mineral = "ankerite05" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.0925)
    cation.Add("Fe2", 0.4075)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "ankerite06" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.1295)
    cation.Add("Fe2", 0.3705)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "ankerite09" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.1785)
    cation.Add("Fe2", 0.3215)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "ankerite10" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.125)
    cation.Add("Fe2", 0.375)
    cation.Add("C", 1)
    cation.Add("O", 3)
```

```
ElseIf mineral = "ankerite12" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.148)
    cation.Add("Fe2", 0.352)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "ankerite14" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.21)
    cation.Add("Fe2", 0.29)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "ankerite03" Then
    cation.Add("Ca", 0.5)
    cation.Add("Mg", 0.03779)
    cation.Add("Fe2", 0.46221)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst05" Then
    cation.Add("Ca", 0.8777)
    cation.Add("Mg", 0.1223)
    cation.Add("Fe2", 0)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst06" Then
    cation.Add("Ca", 0.944)
    cation.Add("Mg", 0.056)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst09" Then
    cation.Add("Ca", 0.8839)
    cation.Add("Mg", 0.1161)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst10" Then
    cation.Add("Ca", 0.9455)
    cation.Add("Mg", 0.0545)
```

```
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst12" Then
    cation.Add("Ca", 0.9636)
    cation.Add("Mg", 0.0364)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCmst14" Then
    cation.Add("Ca", 0.89)
    cation.Add("Mg", 0.11)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "MSTcc05" Then
    cation.Add("Ca", 0.0189)
    cation.Add("Mg", 0.9811)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "MSTcc06" Then
    cation.Add("Ca", 0.0187)
    cation.Add("Mg", 0.9813)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "MSTcc09" Then
    cation.Add("Ca", 0.0268)
    cation.Add("Mg", 0.9732)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "MSTcc12" Then
    cation.Add("Ca", 0.0364)
    cation.Add("Mg", 0.9636)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "MSTcc14" Then
    cation.Add("Ca", 0.024)
    cation.Add("Mg", 0.976)
    cation.Add("C", 1)
    cation.Add("O", 3)
```

```
ElseIf mineral = "CCsid05" Then
    cation.Add("Ca", 0.676)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.324)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsid06" Then
    cation.Add("Ca", 0.793)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.207)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsid09" Then
    cation.Add("Ca", 0.8182)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.1818)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsid10" Then
    cation.Add("Ca", 0.9266)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.0734)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsid12" Then
    cation.Add("Ca", 0.9074)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.0926)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsid14" Then
    cation.Add("Ca", 0.9)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.1)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsidB09" Then
```

```
    cation.Add("Ca", 0.6636)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.3364)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc05" Then
    cation.Add("Ca", 0.095)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.905)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc06" Then
    cation.Add("Ca", 0.0943)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.9057)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc09" Then
    cation.Add("Ca", 0.0909)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.9091)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc10" Then
    cation.Add("Ca", 0.055)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.945)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc12" Then
    cation.Add("Ca", 0.0741)
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.9259)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDcc14" Then
    cation.Add("Ca", 0.085)
```

```
    cation.Add("Mg", 0)
    cation.Add("Fe2", 0.915)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDmst05" Then
    cation.Add("Ca", 0.091400469)
    cation.Add("Mg", 0.068677)
    cation.Add("Fe2", 0.839922531)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDmst06" Then
    cation.Add("Ca", 0.082383777)
    cation.Add("Mg", 0.1746714)
    cation.Add("Fe2", 0.742944823)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDmst10" Then
    cation.Add("Ca", 0.049764)
    cation.Add("Mg", 0.0952)
    cation.Add("Fe2", 0.855036)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDmst12" Then
    cation.Add("Ca", 0.072384205)
    cation.Add("Mg", 0.11119944)
    cation.Add("Fe2", 0.816416355)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "SIDmst14" Then
    cation.Add("Ca", 0.078027664)
    cation.Add("Mg", 0.14152)
    cation.Add("Fe2", 0.780452336)
    cation.Add("C", 1)
    cation.Add("O", 3)
ElseIf mineral = "CCsidmst05" Then
    cation.Add("Ca", 0.66)
    cation.Add("Mg", 0.025699089)
```

```
            cation.Add("Fe2", 0.314300911)
            cation.Add("C", 1)
            cation.Add("O", 3)
        ElseIf mineral = "CCsidmst06" Then
            cation.Add("Ca", 0.693717865)
            cation.Add("Mg", 0.058301856)
            cation.Add("Fe2", 0.247980279)
            cation.Add("C", 1)
            cation.Add("O", 3)
        ElseIf mineral = "CCsidmst10" Then
            cation.Add("Ca", 0.769686641)
            cation.Add("Mg", 0.017036426)
            cation.Add("Fe2", 0.213276933)
            cation.Add("C", 1)
            cation.Add("O", 3)
        ElseIf mineral = "CCsidmst12" Then
            cation.Add("Ca", 0.785161115)
            cation.Add("Mg", 0.163462569)
            cation.Add("Fe2", 0.051376316)
            cation.Add("C", 1)
            cation.Add("O", 3)
        ElseIf mineral = "CCsidmst14" Then
            cation.Add("Ca", 0.715115932)
            cation.Add("Mg", 0.03183172)
            cation.Add("Fe2", 0.253052348)
            cation.Add("C", 1)
            cation.Add("O", 3)
        End If

        Return cation
    End Function
#End Region

#Region "CarbonatesOxydeData"

    'This REGION contains the definition of the ternary diagrams of carbonates and oxydes
```

```
'These declarations are used to avoid typing mistakes in the following classes (cf. tetraedre definition)
Private calcite As String = "calcite"
Private ankerite As String = "ankerite"
Private magnesite As String = "magnesite"
Private siderite As String = "siderite"
Private dolomite As String = "dolomite"
Private SIDcc As String = "SIDcc"
Private MSTcc As String = "MSTcc"
Private CCsid As String = "CCsid"
Private CCsidB As String = "CCsidB"
Private CCmst As String = "CCmst"
Private CCsidmst As String = "CCsidmst"
Private SIDmst As String = "SIDmst"
Private ilmenite As String = "ilmenite"
Private pseudobrookite As String = "pseudobrookite"
Private hematite As String = "hematite"
Private magnetite As String = "magnetite"
Private ulvospinel As String = "ulvospinel"
Private wustite As String = "wustite"
Private rutile As String = "rutile"
Private HEMilm As String = "HEMilm"
Private PSBfe As String = "PSBfe"

Private Sub Assemblage01()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})    '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})   '        2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})    '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '       6
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})     '       7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})    '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11
```

```
    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0.25, 0.5, 0, 0.25}, {0, 1 / 3, 2 / 3, 0}, {0, 1 / 3, 0, 2 / 3}})
    assemblageCARBONTE02.Add(1, {{0, 1, 0, 0}, {0.25, 0.5, 0.25, 0}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0, 0.25}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "01", PSBfe & "01", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "01", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "01", rutile, pseudobrookite})
    assemblageOXYDE.Add(5, {HEMilm & "01", PSBfe & "01", hematite})
    assemblageOXYDE.Add(6, {PSBfe & "01", hematite, pseudobrookite})
    assemblageOXYDE.Add(7, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(8, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(9, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage02()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})    '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})   '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})    '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '       6
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})     '       7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})    '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0.25, 0.5, 0.25, 0}, {0.25, 0.5, 0, 0.25}, {0.5, 0.5, 0, 0}})
    assemblageCARBONTE02.Add(1, {{0, 1, 0, 0}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}, {0, 1 / 3, 0, 2 / 3}})
    assemblageCARBONTE02.Add(2, {{0, 1, 0, 0}, {0, 1 / 3, 0, 2 / 3}, {0.25, 0.5, 0, 0.25}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
```

```
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "02", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "02", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "02", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "02", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage03()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})    '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})   '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})    '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '        6
    assemblageCARBONATE01.Add(7, {ankerite & "03", SIDmst & "05", MSTcc & "05"})     '       7
    assemblageCARBONATE01.Add(8, {ankerite & "03", dolomite, MSTcc & "05"})     '      8
    assemblageCARBONATE01.Add(9, {ankerite & "03", CCmst & "05", CCsidmst & "05"})       '    9
    assemblageCARBONATE01.Add(10, {ankerite & "03", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "03"})     '          11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0.25, 0.5, 0.25, 0}, {0.25, 0.5, 0, 0.25}, {0.5, 0.5, 0, 0}})
    assemblageCARBONTE02.Add(1, {{0, 1, 0, 0}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}, {0, 1 / 3, 0, 2 / 3}})
    assemblageCARBONTE02.Add(2, {{0, 1, 0, 0}, {0, 1 / 3, 0, 2 / 3}, {0.25, 0.5, 0, 0.25}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "03", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "03", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "03", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "03", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
```

```
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage04()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})    '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})   '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})    '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '        6
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})     '       7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})     '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})     '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11


    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0, 1 / 3, 0, 2 / 3}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "01", PSBfe & "01", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "01", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "01", rutile, pseudobrookite})
    assemblageOXYDE.Add(5, {HEMilm & "01", PSBfe & "01", hematite})
    assemblageOXYDE.Add(6, {PSBfe & "01", hematite, pseudobrookite})
    assemblageOXYDE.Add(7, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(8, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(9, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage05()

    'ASSEMBLAGES DEFINITION : CARBONATES
```

```
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})     '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})    '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})     '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})       '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})      '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})      '       6
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})      '       7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})     '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})       '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11


    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {2 / 15, 8 / 15, 5 / 15, 0}, {0, 1 / 3, 0, 2 / 3}, {0, 1 / 3, 2 / 3, 0}})
    assemblageCARBONTE02.Add(1, {{0, 1, 0, 0}, {2 / 15, 8 / 15, 5 / 15, 0}, {0, 1 / 3, 0, 2 / 3}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "01", PSBfe & "01", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "01", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "01", rutile, pseudobrookite})
    assemblageOXYDE.Add(5, {HEMilm & "01", PSBfe & "01", hematite})
    assemblageOXYDE.Add(6, {PSBfe & "01", hematite, pseudobrookite})
    assemblageOXYDE.Add(7, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(8, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(9, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage06()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "06", magnesite, siderite})     '    1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "06", MSTcc & "06"})    '         2
    assemblageCARBONATE01.Add(3, {CCsid & "06", CCsidmst & "06", calcite})     '      3
    assemblageCARBONATE01.Add(4, {CCmst & "06", CCsidmst & "06", calcite})     '      4
    assemblageCARBONATE01.Add(5, {CCsidmst & "06", SIDcc & "06", SIDmst & "06"})      '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "06", CCsid & "06", SIDcc & "06"})      '       6
```

```
    assemblageCARBONATE01.Add(7, {ankerite & "06", SIDmst & "06", MSTcc & "06"})      '      7
    assemblageCARBONATE01.Add(8, {ankerite & "06", dolomite, MSTcc & "06"})       '    8
    assemblageCARBONATE01.Add(9, {ankerite & "06", CCmst & "06", CCsidmst & "06"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "06", dolomite, CCmst & "06"})      '    10
    assemblageCARBONATE01.Add(11, {CCsidmst & "06", SIDmst & "06", ankerite & "06"})      '         11


    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {2 / 15, 8 / 15, 5 / 15, 0}, {0, 4 / 7, 3 / 7, 0}, {0, 1 / 3, 0, 2 / 3}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "06", PSBfe & "06", rutile})      '       1
    assemblageOXYDE.Add(2, {HEMilm & "06", ilmenite, rutile})       '          2
    assemblageOXYDE.Add(3, {PSBfe & "06", rutile, pseudobrookite})       '     3
    assemblageOXYDE.Add(4, {HEMilm & "06", PSBfe & "06", hematite})      '     4
    assemblageOXYDE.Add(5, {PSBfe & "06", hematite, pseudobrookite})       '          5
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})       '      6
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})      '    7
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})      '     8
End Sub

Private Sub Assemblage07()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {magnesite, SIDcc & "09", siderite})     '    1
    assemblageCARBONATE01.Add(2, {MSTcc & "09", SIDcc & "09", magnesite})     '       2
    assemblageCARBONATE01.Add(3, {CCmst & "09", CCsid & "09", calcite})     '          3
    assemblageCARBONATE01.Add(4, {CCsid & "09", CCsidB & "09", ankerite & "09"})      '      4
    assemblageCARBONATE01.Add(5, {ankerite & "09", CCsidB & "09", SIDcc & "09"})      '      5
    assemblageCARBONATE01.Add(6, {ankerite & "09", CCsid & "09", CCmst & "09"})     '       6
    assemblageCARBONATE01.Add(7, {CCmst & "09", ankerite & "09", dolomite})      '    7
    assemblageCARBONATE01.Add(8, {ankerite & "09", dolomite, SIDcc & "09"})      '    8
    assemblageCARBONATE01.Add(9, {dolomite, MSTcc & "09", SIDcc & "09"})     '       9

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}, {0, 1 / 3, 0, 2 / 3}})
    assemblageCARBONTE02.Add(1, {{0, 1, 0, 0}, {0.25, 0.5, 0.25, 0}, {0.25, 0.5, 0, 0.25}, {0, 1 / 3, 0, 2 / 3}})
```

```
    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "07", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "07", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "07", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "07", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage08()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {magnesite, SIDcc & "09", siderite})     '    1
    assemblageCARBONATE01.Add(2, {MSTcc & "09", SIDcc & "09", magnesite})     '       2
    assemblageCARBONATE01.Add(3, {CCmst & "09", CCsid & "09", calcite})    '          3
    assemblageCARBONATE01.Add(4, {CCsid & "09", CCsidB & "09", ankerite & "09"})      '      4
    assemblageCARBONATE01.Add(5, {ankerite & "09", CCsidB & "09", SIDcc & "09"})      '      5
    assemblageCARBONATE01.Add(6, {ankerite & "09", CCsid & "09", CCmst & "09"})     '       6
    assemblageCARBONATE01.Add(7, {CCmst & "09", ankerite & "09", dolomite})      '    7
    assemblageCARBONATE01.Add(8, {ankerite & "09", dolomite, SIDcc & "09"})      '    8
    assemblageCARBONATE01.Add(9, {dolomite, MSTcc & "09", SIDcc & "09"})     '       9

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0, 0.5, 0.5, 0}, {0.25, 0.5, 0.25, 0}, {0, 0.5, 0, 0.5}})
    assemblageCARBONTE02.Add(1, {{0, 1 / 3, 0, 2 / 3}, {0, 0.5, 0, 0.5}, {0, 0.5, 0.5, 0}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "02", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "02", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "02", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "02", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
```

```
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage09()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {magnesite, SIDcc & "09", siderite})     '    1
    assemblageCARBONATE01.Add(2, {MSTcc & "09", SIDcc & "09", magnesite})     '       2
    assemblageCARBONATE01.Add(3, {CCmst & "09", CCsid & "09", calcite})    '          3
    assemblageCARBONATE01.Add(4, {CCsid & "09", CCsidB & "09", ankerite & "09"})      '      4
    assemblageCARBONATE01.Add(5, {ankerite & "09", CCsidB & "09", SIDcc & "09"})      '      5
    assemblageCARBONATE01.Add(6, {ankerite & "09", CCsid & "09", CCmst & "09"})     '       6
    assemblageCARBONATE01.Add(7, {CCmst & "09", ankerite & "09", dolomite})      '    7
    assemblageCARBONATE01.Add(8, {ankerite & "09", dolomite, SIDcc & "09"})      '    8
    assemblageCARBONATE01.Add(9, {dolomite, MSTcc & "09", SIDcc & "09"})     '       9

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0.25, 0.5, 0.25, 0}, {0, 0.5, 0, 0.5}, {0, 0.5, 0.5, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "01", PSBfe & "01", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "01", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "01", rutile, pseudobrookite})
    assemblageOXYDE.Add(5, {HEMilm & "01", PSBfe & "01", hematite})
    assemblageOXYDE.Add(6, {PSBfe & "01", hematite, pseudobrookite})
    assemblageOXYDE.Add(7, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(8, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(9, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage10()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {siderite, magnesite, SIDcc & "10"})     '    1
    assemblageCARBONATE01.Add(2, {calcite, CCsid & "10", CCsidmst & "10"})     '      2
    assemblageCARBONATE01.Add(3, {calcite, CCmst & "10", CCsidmst & "10"})    '       3
    assemblageCARBONATE01.Add(4, {CCsid & "10", SIDmst & "10", SIDcc & "10"})    '    4
```

```
    assemblageCARBONATE01.Add(5, {ankerite & "10", CCmst & "10", CCsidmst & "10"})      '    5
    assemblageCARBONATE01.Add(6, {CCmst & "10", ankerite & "10", dolomite})      '    6
    assemblageCARBONATE01.Add(7, {magnesite, dolomite, SIDmst & "10"})      '         7
    assemblageCARBONATE01.Add(8, {dolomite, ankerite & "10", SIDmst & "10"})      '          8
    assemblageCARBONATE01.Add(9, {ankerite & "10", SIDmst & "10", CCsidmst & "10"})      '          9

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {ilmenite, magnetite, wustite})      '       1
    assemblageOXYDE.Add(2, {ilmenite, hematite, magnetite})   '         2
    assemblageOXYDE.Add(3, {hematite, ilmenite, rutile})    '    3
End Sub

Private Sub Assemblage11()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "06", magnesite, siderite})     '    1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "06", MSTcc & "06"})    '        2
    assemblageCARBONATE01.Add(3, {CCsid & "06", CCsidmst & "06", calcite})     '      3
    assemblageCARBONATE01.Add(4, {CCmst & "06", CCsidmst & "06", calcite})     '      4
    assemblageCARBONATE01.Add(5, {CCsidmst & "06", SIDcc & "06", SIDmst & "06"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "06", CCsid & "06", SIDcc & "06"})     '        6
    assemblageCARBONATE01.Add(7, {ankerite & "06", SIDmst & "06", MSTcc & "06"})      '     7
    assemblageCARBONATE01.Add(8, {ankerite & "06", dolomite, MSTcc & "06"})      '    8
    assemblageCARBONATE01.Add(9, {ankerite & "06", CCmst & "06", CCsidmst & "06"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "06", dolomite, CCmst & "06"})     '    10
    assemblageCARBONATE01.Add(11, {CCsidmst & "06", SIDmst & "06", ankerite & "06"})      '         11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {2 / 15, 8 / 15, 5 / 15, 0}, {0, 4 / 7, 3 / 7, 0}, {0, 1 / 3, 0, 2 / 3}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "06", PSBfe & "06", rutile})      '       1
    assemblageOXYDE.Add(2, {HEMilm & "06", ilmenite, rutile})       '          2
    assemblageOXYDE.Add(3, {PSBfe & "06", rutile, pseudobrookite})       '     3
    assemblageOXYDE.Add(4, {HEMilm & "06", PSBfe & "06", hematite})      '     4
    assemblageOXYDE.Add(5, {PSBfe & "06", hematite, pseudobrookite})       '          5
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})      '      6
```

```
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})      '    7
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})      '     8
End Sub

Private Sub Assemblage12()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {magnesite, SIDcc & "12", MSTcc & "12"})     '       1
    assemblageCARBONATE01.Add(2, {siderite, SIDcc & "12", magnesite})      '    2
    assemblageCARBONATE01.Add(3, {CCsidmst & "12", CCsid & "12", calcite})     '      3
    assemblageCARBONATE01.Add(4, {CCsidmst & "12", CCmst & "12", calcite})     '      4
    assemblageCARBONATE01.Add(5, {CCsidmst & "12", SIDmst & "12", CCsid & "12"})     '       5
    assemblageCARBONATE01.Add(6, {SIDcc & "12", SIDmst & "12", CCsid & "12"})     '          6
    assemblageCARBONATE01.Add(7, {ankerite & "12", CCsidmst & "12", SIDmst & "12"})      '          7
    assemblageCARBONATE01.Add(8, {dolomite, ankerite & "12", CCmst & "12"})      '    8
    assemblageCARBONATE01.Add(9, {CCsidmst & "12", ankerite & "12", CCmst & "12"})      '    8-bis
    assemblageCARBONATE01.Add(10, {dolomite, ankerite & "12", SIDmst & "12"})     '          9
    assemblageCARBONATE01.Add(11, {dolomite, SIDmst & "12", MSTcc & "12"})     '      10

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {ilmenite, magnetite, wustite})      '       1
    assemblageOXYDE.Add(2, {ilmenite, hematite, magnetite})   '         2
    assemblageOXYDE.Add(3, {hematite, ilmenite, rutile})    '    3
End Sub

Private Sub Assemblage13()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {MSTcc & "14", SIDcc & "14", magnesite})     '       1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "14", siderite})     '     1-bis
    assemblageCARBONATE01.Add(3, {CCmst & "14", calcite, CCsidmst & "14"})     '      2
    assemblageCARBONATE01.Add(4, {CCsidmst & "14", calcite, CCsid & "14"})     '      3
    assemblageCARBONATE01.Add(5, {CCsid & "14", SIDmst & "14", SIDcc & "14"})     '          4
    assemblageCARBONATE01.Add(6, {CCsidmst & "14", ankerite & "14", SIDmst & "14"})      '          5
    assemblageCARBONATE01.Add(7, {dolomite, ankerite & "14", CCmst & "14"})       '          6
    assemblageCARBONATE01.Add(8, {CCsidmst & "14", ankerite & "14", CCmst & "14"})       '          7
    assemblageCARBONATE01.Add(9, {dolomite, ankerite & "14", SIDmst & "14"})      '          8
```

```
    assemblageCARBONATE01.Add(10, {dolomite, MSTcc & "14", SIDmst & "14"})          '       9

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {ilmenite, magnetite, wustite})      '       1
    assemblageOXYDE.Add(2, {ilmenite, hematite, magnetite})   '         2
    assemblageOXYDE.Add(3, {hematite, ilmenite, rutile})    '    3
End Sub

Private Sub Assemblage14()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {MSTcc & "14", SIDcc & "14", magnesite})     '       1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "14", siderite})     '     1-bis
    assemblageCARBONATE01.Add(3, {CCmst & "14", calcite, CCsidmst & "14"})     '      2
    assemblageCARBONATE01.Add(4, {CCsidmst & "14", calcite, CCsid & "14"})     '      3
    assemblageCARBONATE01.Add(5, {CCsid & "14", SIDmst & "14", SIDcc & "14"})     '          4
    assemblageCARBONATE01.Add(6, {CCsidmst & "14", ankerite & "14", SIDmst & "14"})      '          5
    assemblageCARBONATE01.Add(7, {dolomite, ankerite & "14", CCmst & "14"})       '          6
    assemblageCARBONATE01.Add(8, {CCsidmst & "14", ankerite & "14", CCmst & "14"})       '          7
    assemblageCARBONATE01.Add(9, {dolomite, ankerite & "14", SIDmst & "14"})      '          8
    assemblageCARBONATE01.Add(10, {dolomite, MSTcc & "14", SIDmst & "14"})          '       9

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {ilmenite, magnetite, wustite})      '       1
    assemblageOXYDE.Add(2, {ilmenite, hematite, magnetite})   '         2
    assemblageOXYDE.Add(3, {hematite, ilmenite, rutile})    '    3
End Sub

Private Sub Assemblage15()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})     '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})   '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})     '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '        6
```

```
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})     '      7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})     '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0, 1 / 3, 0, 2 / 3}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "02", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "02", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "02", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "02", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage16()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "05", magnesite, siderite})     '     1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "05", MSTcc & "05"})    '         2
    assemblageCARBONATE01.Add(3, {CCsid & "05", CCsidmst & "05", calcite})     '       3
    assemblageCARBONATE01.Add(4, {CCmst & "05", CCsidmst & "05", calcite})      '     4
    assemblageCARBONATE01.Add(5, {CCsidmst & "05", SIDcc & "05", SIDmst & "05"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "05", CCsid & "05", SIDcc & "05"})     '        6
    assemblageCARBONATE01.Add(7, {ankerite & "05", SIDmst & "05", MSTcc & "05"})     '       7
    assemblageCARBONATE01.Add(8, {ankerite & "05", dolomite, MSTcc & "05"})     '      8
    assemblageCARBONATE01.Add(9, {ankerite & "05", CCmst & "05", CCsidmst & "05"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "05", dolomite, CCmst & "05"})      '          10
    assemblageCARBONATE01.Add(11, {CCsidmst & "05", SIDmst & "05", ankerite & "05"})     '          11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {0, 1 / 3, 0, 2 / 3}, {0, 1 / 3, 2 / 3, 0}, {0.25, 0.5, 0.25, 0}})
```

```
    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "07", PSBfe & "02", rutile})
    assemblageOXYDE.Add(2, {HEMilm & "07", ilmenite, rutile})
    assemblageOXYDE.Add(3, {PSBfe & "02", rutile, pseudobrookite})
    assemblageOXYDE.Add(4, {HEMilm & "07", PSBfe & "02", hematite})
    assemblageOXYDE.Add(5, {PSBfe & "02", hematite, pseudobrookite})
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})
    assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})
    assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})
End Sub

Private Sub Assemblage17()

    'ASSEMBLAGES DEFINITION : CARBONATES
    assemblageCARBONATE01.Add(1, {SIDcc & "06", magnesite, siderite})     '    1
    assemblageCARBONATE01.Add(2, {magnesite, SIDcc & "06", MSTcc & "06"})    '        2
    assemblageCARBONATE01.Add(3, {CCsid & "06", CCsidmst & "06", calcite})     '      3
    assemblageCARBONATE01.Add(4, {CCmst & "06", CCsidmst & "06", calcite})     '      4
    assemblageCARBONATE01.Add(5, {CCsidmst & "06", SIDcc & "06", SIDmst & "06"})     '       5
    assemblageCARBONATE01.Add(6, {CCsidmst & "06", CCsid & "06", SIDcc & "06"})     '       6
    assemblageCARBONATE01.Add(7, {ankerite & "06", SIDmst & "06", MSTcc & "06"})      '     7
    assemblageCARBONATE01.Add(8, {ankerite & "06", dolomite, MSTcc & "06"})      '    8
    assemblageCARBONATE01.Add(9, {ankerite & "06", CCmst & "06", CCsidmst & "06"})      '    9
    assemblageCARBONATE01.Add(10, {ankerite & "06", dolomite, CCmst & "06"})     '    10
    assemblageCARBONATE01.Add(11, {CCsidmst & "06", SIDmst & "06", ankerite & "06"})      '         11

    'ASSEMBLAGES COORDINATES : CARBONATES
    assemblageCARBONTE02.Add(0, {{0, 1, 0, 0}, {2 / 15, 8 / 15, 5 / 15, 0}, {0, 4 / 7, 3 / 7, 0}, {0, 1 / 3, 0, 2 / 3}})

    'ASSEMBLAGES DEFINITION : OXYDES
    assemblageOXYDE.Add(1, {HEMilm & "06", PSBfe & "06", rutile})      '       1
    assemblageOXYDE.Add(2, {HEMilm & "06", ilmenite, rutile})       '          2
    assemblageOXYDE.Add(3, {PSBfe & "06", rutile, pseudobrookite})       '     3
    assemblageOXYDE.Add(4, {HEMilm & "06", PSBfe & "06", hematite})      '     4
    assemblageOXYDE.Add(5, {PSBfe & "06", hematite, pseudobrookite})       '          5
    assemblageOXYDE.Add(6, {ilmenite, magnetite, hematite})      '      6
```

```
        assemblageOXYDE.Add(7, {magnetite, ulvospinel, ilmenite})      '    7
        assemblageOXYDE.Add(8, {ulvospinel, magnetite, wustite})      '     8
    End Sub


#End Region

#Region "SilicateData"

    'These classes define the Al-Ca-KNa-FeMg silicate tetraedres

#Region "MxNamesSilicates"

    'These declarations are used to avoid typing mistakes in the following classes (cf. tetraedre definition)
    Public actinote As String = "actinolite"
    Public amesite As String = "amesite"
    Public anorthite As String = "anorthite"
    Public biotite As String = "biotite"
    Public carpholite As String = "carpholite"
    Public chloritoide As String = "chloritoid"
    Public clinozoisite As String = "clinozoisite"
    Public CPX As String = "CPX"
    Public CPXgrt As String = "CPXgrt"
    Public cordierite As String = "cordierite"
    Public disthene As String = "kyanite"
    Public daphnite As String = "daphnite"
    Public FM As String = "FM"
    Public FSP04 As String = "FSP04"
    Public FSP05_A As String = "FSP05_A"
    Public FSP05_B As String = "FSP05_B"
    Public FSP06 As String = "FSP06"
    Public FSP07 As String = "FSP07"
    Public FSP08 As String = "FSP08"
    Public FSP09 As String = "FSP09"
    Public FSP11 As String = "FSP11"
    Public FSP17 As String = "FSP17"
    Public grossular As String = "grossular"
```

```
    Public GROSSss As String = "GROSSss"
    Public GRThalf As String = "GRThalf"
    Public GRTss As String = "GRTss"
    Public GRTss05 As String = "GRTss05"
    Public GRTpyalm As String = "GRTpyalm"
    Public Kfeld As String = "Kfeld"
    Public lawsonite As String = "lawsonite"
    Public margarite As String = "margarite"
    Public NaK As String = "NaK"
    Public pyrophyllite As String = "pyrophyllite"
    Public sillimanite As String = "sillimanite"
    Public staurolite As String = "staurolite"
    Public sudoite As String = "sudoite"
    Public spinelss As String = "spinelss"
    Public WhiteMica As String = "WhiteMica"
    Public wollastonite As String = "wollastonite"
#End Region

    Public Function Silicates3AMP675() As Dictionary(Of Integer, Array)

        Dim assList As New Dictionary(Of Integer, Array)

        assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
        assList.Add(2, {CPX, anorthite, Kfeld, wollastonite})     '  IIIA

        If TetraLocation = enumTetraedre.TetraI Or TetraLocation = enumTetraedre.TetraIII Then
            assList.Add(3, {cordierite, anorthite, Kfeld, sillimanite})     'IA
            assList.Add(4, {FM, anorthite, actinote, Kfeld})     'ID
            assList.Add(5, {anorthite, CPX, actinote, Kfeld})     'IF

            If TetraLocation = enumTetraedre.TetraI Then
                If NaRatio < 0.9 And MgFeRatio > 0.63 And MgMnRatio > 0.35 Then
                    assList.Add(6, {cordierite, anorthite, biotite, FM})     'IB
                    assList.Add(7, {cordierite, anorthite, biotite, Kfeld})     'IC
                Else
                    assList.Add(6, {cordierite, anorthite, FM, Kfeld})     'IBC
                End If
```

```
        ElseIf TetraLocation = enumTetraedre.TetraIII Then
            If NaRatio < 0.9 And MgFeRatio > 0.63 And MgMnRatio > 0.35 Then
                assList.Add(6, {cordierite, GRTpyalm, biotite, anorthite})     'IVB
                assList.Add(7, {cordierite, Kfeld, biotite, anorthite})     'IVC
                assList.Add(8, {FM, anorthite, biotite, GRTss05})     'IVD
            Else
                assList.Add(6, {cordierite, GRTpyalm, anorthite, Kfeld})     'IVB
                assList.Add(7, {FM, GRTss05, anorthite, Kfeld})     'IVC
            End If
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(3, {GRTpyalm, anorthite, Kfeld, sillimanite})     ' IIA
        assList.Add(4, {anorthite, FM, Kfeld, CPX})     'IID
        If NaRatio < 0.9 And MgFeRatio > 0.63 And MgMnRatio > 0.35 Then
            assList.Add(5, {GRTpyalm, biotite, Kfeld, anorthite})     ' IIB
            assList.Add(6, {GRTss05, biotite, FM, anorthite})     '     IIC
        Else
            assList.Add(5, {GRTss05, FM, Kfeld, anorthite})     '       IIBC
        End If
    End If

    Return assList
End Function

Public Function Silicates3GRA800() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {CPX, anorthite, Kfeld, wollastonite})     '  IIIA

    If TetraLocation = enumTetraedre.TetraI Or TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(3, {cordierite, anorthite, Kfeld, sillimanite})     'IA
        If MgFeRatio > 0.8 Then                'Reduce stability field of tremolite
            assList.Add(4, {FM, anorthite, actinote, Kfeld})     'ID
```

```
            assList.Add(5, {anorthite, CPX, actinote, Kfeld})     'IE
        Else
            assList.Add(4, {anorthite, CPX, FM, Kfeld})     'IE
            assList.Add(5, {anorthite, CPX, FM, Kfeld})     'IE    'Doublon, it's fine
        End If

        If TetraLocation = enumTetraedre.TetraI Then
            If NaRatio < 0.85 And MgFeRatio > 0.8 And MgMnRatio > 0.5 Then
                assList.Add(6, {cordierite, anorthite, biotite, FM})     'IB
                assList.Add(7, {cordierite, anorthite, biotite, Kfeld})     'IC
            Else
                assList.Add(6, {cordierite, anorthite, Kfeld, FM})     'IBC
            End If
        ElseIf TetraLocation = enumTetraedre.TetraIII Then
            If NaRatio < 0.85 And MgFeRatio > 0.8 And MgMnRatio > 0.5 Then
                assList.Add(6, {GRTss05, FM, biotite, anorthite})     'IVB
                assList.Add(7, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
                assList.Add(8, {biotite, cordierite, Kfeld, anorthite})     'IVF
            Else
                assList.Add(6, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
                assList.Add(7, {GRTss05, FM, Kfeld, anorthite})     'IVB
            End If
        End If
    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(3, {spinelss, anorthite, Kfeld, sillimanite})     'IIA
        assList.Add(4, {spinelss, anorthite, Kfeld, GRTpyalm})     'IIB
        assList.Add(5, {GRTss05, anorthite, Kfeld, FM})     'IIC
        assList.Add(6, {CPX, anorthite, Kfeld, FM})     'IID
    End If

    Return assList
End Function

Public Function Silicates5GRA900() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)
```

```
    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {CPX, anorthite, Kfeld, wollastonite})     '  IIIA

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(3, {cordierite, anorthite, Kfeld, sillimanite})     '      IA
        assList.Add(4, {FM, anorthite, CPX, Kfeld})     ' ID
        If NaRatio < 0.2 And MgFeRatio > 0.85 And MgMnRatio > 0.6 Then
            assList.Add(5, {cordierite, anorthite, biotite, FM})     '  IB
            assList.Add(6, {cordierite, anorthite, biotite, Kfeld})     '      IC
        Else
            assList.Add(5, {cordierite, anorthite, Kfeld, FM})     '    IBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(3, {GRTss, anorthite, Kfeld, FM})     '      IIC
        assList.Add(4, {FM, anorthite, CPX, Kfeld})     ' IID
        If MgFeRatio < 0.05 And FeMnRatio > 0.95 Then
            assList.Add(5, {spinelss, anorthite, Kfeld, sillimanite})     '    IIA
            assList.Add(6, {spinelss, anorthite, Kfeld, GRTpyalm})     '       IIB
        Else
            assList.Add(5, {sillimanite, anorthite, Kfeld, GRTpyalm})     '    IIAB
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(3, {FM, anorthite, CPX, Kfeld})     'IIA
        assList.Add(4, {cordierite, anorthite, Kfeld, sillimanite})     'IIB
        assList.Add(5, {cordierite, anorthite, Kfeld, GRTpyalm})     'IIC
        assList.Add(6, {GRTss, anorthite, Kfeld, FM})     'IID
    End If

    Return assList
End Function

Public Function Silicates5AMP670() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)
```

```
Dim BTorNOT As Boolean = True
If MgMnRatio < 0.2 And FeMnRatio < 0.9 Then
    BTorNOT = False
ElseIf NaRatio > 0.9 Then
    BTorNOT = False
End If

Dim WMorNOT As Boolean = True
If NaRatio > 0.3 Then
    WMorNOT = False
End If

assList.Add(1, {NaK, Kfeld, FM, wollastonite})   '    impossible
assList.Add(2, {CPX, anorthite, FSP04, grossular})     '     IIIA
assList.Add(3, {CPX, wollastonite, FSP04, grossular})     '  IIIB
assList.Add(4, {CPX, wollastonite, FSP04, Kfeld})     '      IIIC

If TetraLocation = enumTetraedre.TetraI Then
    assList.Add(5, {actinote, anorthite, FM, Kfeld})     'IFG
    assList.Add(6, {anorthite, CPX, actinote, Kfeld})     '  IH

    If BTorNOT = True And WMorNOT = True Then
        assList.Add(7, {cordierite, anorthite, sillimanite, WhiteMica})     '     IA
        assList.Add(8, {biotite, anorthite, Kfeld, WhiteMica})     '       IB
        assList.Add(9, {biotite, anorthite, cordierite, WhiteMica})     '  IC
        assList.Add(10, {cordierite, anorthite, biotite, FM})     'IDE
    Else
        If WMorNOT = True Then
            assList.Add(11, {cordierite, anorthite, sillimanite, WhiteMica})     '      IA-bis
            assList.Add(12, {cordierite, anorthite, Kfeld, WhiteMica})     '      IB-bis
            If BTorNOT = True Then
                assList.Add(13, {cordierite, biotite, anorthite, Kfeld})     '    IC-bis
                assList.Add(14, {cordierite, biotite, anorthite, FM})     'IDE-bis
            Else
                assList.Add(13, {cordierite, FM, anorthite, Kfeld})     'ICDE-bis
            End If
        Else
```

```
            assList.Add(11, {cordierite, sillimanite, anorthite, Kfeld})     '    IAB-bis
            If BTorNOT = True Then
                assList.Add(12, {cordierite, biotite, anorthite, Kfeld})     '    IC-bis
                assList.Add(13, {cordierite, biotite, anorthite, FM})     'IDE-bis
            Else
                assList.Add(12, {cordierite, FM, anorthite, Kfeld})     'ICDE-bis
            End If
        End If
    End If

ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(5, {CPXgrt, FM, anorthite, Kfeld})     '     IIE
    If BTorNOT = False Then
        assList.Add(6, {GRTpyalm, sillimanite, anorthite, Kfeld})     '    IIAB
        assList.Add(7, {GRTss, FM, anorthite, Kfeld})     '  IICD
    Else
        assList.Add(6, {GRTpyalm, anorthite, sillimanite, WhiteMica})     '       IIA
        assList.Add(7, {GRTpyalm, anorthite, Kfeld, WhiteMica})     '      IIB
        assList.Add(8, {GRTpyalm, biotite, anorthite, Kfeld})     ' IIC
        assList.Add(9, {GRTss, biotite, anorthite, FM})     '       IID
    End If

ElseIf TetraLocation = enumTetraedre.TetraIII Then
    assList.Add(5, {actinote, anorthite, FM, Kfeld})     'IFG
    assList.Add(6, {anorthite, CPX, actinote, Kfeld})     'IH
    If BTorNOT = True Then
        assList.Add(7, {GRTss, FM, biotite, anorthite})     'IVD
        assList.Add(8, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
        assList.Add(9, {biotite, cordierite, Kfeld, anorthite})     'IVF
        If WMorNOT = True Then
            assList.Add(10, {cordierite, anorthite, sillimanite, WhiteMica})     'IIA
            assList.Add(11, {cordierite, anorthite, Kfeld, WhiteMica})     'IIB
        Else
            assList.Add(10, {cordierite, sillimanite, anorthite, Kfeld})     'IIAB
        End If
    Else
        assList.Add(7, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
```

```
            assList.Add(8, {GRTss, FM, Kfeld, anorthite})     'IVD
            If WMorNOT = True Then
                assList.Add(9, {cordierite, anorthite, sillimanite, WhiteMica})     'IIA
                assList.Add(10, {cordierite, anorthite, Kfeld, WhiteMica})     'IIB
            Else
                assList.Add(9, {cordierite, sillimanite, anorthite, Kfeld})     'IIAB
            End If
        End If
    End If

    Return assList
End Function

Public Function Silicates7AMP700() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If MgMnRatio < 0.25 And FeMnRatio < 0.9 Then
        BTorNOT = False
    ElseIf NaRatio > 0.9 Then
        BTorNOT = False
    End If
    Dim WMorNOT As Boolean = True
    If NaRatio > 0.6 Then
        WMorNOT = False
    End If

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})   '    impossible
    assList.Add(2, {FSP05_A, Kfeld, CPX, wollastonite})     '    IIIA
    assList.Add(3, {FSP05_A, grossular, CPX, wollastonite})     '       IIIB
    assList.Add(4, {FSP05_B, grossular, CPX, clinozoisite})     '       IIIC
    assList.Add(5, {FSP05_B, grossular, CPX, FSP05_A})     '     IIID
    assList.Add(6, {FSP05_B, anorthite, CPX, clinozoisite})     '       IIIE

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(7, {actinote, anorthite, FM, Kfeld})     'IFG
```

```
        assList.Add(8, {anorthite, CPX, actinote, Kfeld})     '  IH

        If BTorNOT = True And WMorNOT = True Then
            assList.Add(9, {cordierite, anorthite, sillimanite, WhiteMica})     '     IA
            assList.Add(10, {biotite, anorthite, Kfeld, WhiteMica})     '      IB
            assList.Add(11, {biotite, anorthite, cordierite, WhiteMica})     ' IC
            assList.Add(12, {cordierite, anorthite, biotite, FM})     'IDE
        Else
            If WMorNOT = True Then
                assList.Add(13, {cordierite, anorthite, sillimanite, WhiteMica})     '      IA-bis
                assList.Add(14, {cordierite, anorthite, Kfeld, WhiteMica})     '      IB-bis
                If BTorNOT = True Then
                    assList.Add(15, {cordierite, biotite, anorthite, Kfeld})     '    IC-bis
                    assList.Add(16, {cordierite, biotite, anorthite, FM})     'IDE-bis
                Else
                    assList.Add(17, {cordierite, FM, anorthite, Kfeld})     'ICDE-bis
                End If
            Else
                assList.Add(13, {cordierite, sillimanite, anorthite, Kfeld})     '    IAB-bis
                If BTorNOT = True Then
                    assList.Add(14, {cordierite, biotite, anorthite, Kfeld})     '    IC-bis
                    assList.Add(15, {cordierite, biotite, anorthite, FM})     'IDE-bis
                Else
                    assList.Add(14, {cordierite, FM, anorthite, Kfeld})     'ICDE-bis
                End If
            End If
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(7, {CPXgrt, FM, anorthite, Kfeld})     '     IIE
        If WMorNOT = True Then
            assList.Add(8, {GRTpyalm, anorthite, sillimanite, WhiteMica})     '       IIA
            assList.Add(9, {GRTpyalm, anorthite, Kfeld, WhiteMica})     '      IIB
            If BTorNOT = True Then
                assList.Add(10, {GRTpyalm, biotite, anorthite, Kfeld})     '   IIC
                assList.Add(11, {GRTss, biotite, anorthite, FM})     '  IID
            Else
```

```
            assList.Add(10, {GRTss, FM, anorthite, Kfeld})     '    IICD
        End If
    Else
        assList.Add(8, {GRTpyalm, sillimanite, anorthite, Kfeld})     '    IIAB
        If BTorNOT = True Then
            assList.Add(9, {GRTpyalm, biotite, anorthite, Kfeld})     '    IIC
            assList.Add(10, {GRTss, biotite, anorthite, FM})     '  IID
        Else
            assList.Add(9, {GRTss, FM, anorthite, Kfeld})     '     IICD
        End If
    End If

ElseIf TetraLocation = enumTetraedre.TetraIII Then
    assList.Add(7, {actinote, anorthite, FM, Kfeld})     'IFG
    assList.Add(8, {anorthite, CPX, actinote, Kfeld})     'IH
    If BTorNOT = True Then
        assList.Add(9, {GRTss, FM, biotite, anorthite})     'IVB
        assList.Add(10, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
        assList.Add(11, {biotite, cordierite, Kfeld, anorthite})     'IVF
        If WMorNOT = True Then
            assList.Add(12, {cordierite, anorthite, sillimanite, WhiteMica})     'IIA
            assList.Add(13, {cordierite, anorthite, Kfeld, WhiteMica})     'IIB
        Else
            assList.Add(12, {cordierite, sillimanite, anorthite, Kfeld})     'IIAB
        End If
    Else
        assList.Add(9, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
        assList.Add(10, {GRTss, FM, Kfeld, anorthite})     'IVB
        If WMorNOT = True Then
            assList.Add(11, {cordierite, anorthite, sillimanite, WhiteMica})     'IIA
            assList.Add(12, {cordierite, anorthite, Kfeld, WhiteMica})     'IIB
        Else
            assList.Add(11, {cordierite, sillimanite, anorthite, Kfeld})     'IIAB
        End If
    End If
End If
```

```
    Return assList
End Function

Public Function Silicates7AMP600() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If MgMnRatio < 0.05 And FeMnRatio < 0.7 Then
        BTorNOT = False
    ElseIf NaRatio > 0.9 Then
        BTorNOT = False
    End If

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {grossular, clinozoisite, FSP06, CPX})     '  IIIA
    assList.Add(3, {grossular, wollastonite, Kfeld, CPX})     '  IIIB
    assList.Add(4, {grossular, FSP06, Kfeld, CPX})     '  IIIC
    assList.Add(5, {anorthite, FSP06, clinozoisite, actinote})     '    IIID
    assList.Add(6, {CPX, FSP06, clinozoisite, actinote})     '   IIIE
    assList.Add(7, {CPX, FSP06, Kfeld, actinote})     '   IIIF-inner

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(8, {amesite, margarite, WhiteMica, anorthite})     '       IF (for both)
        assList.Add(9, {margarite, WhiteMica, amesite, disthene})     ' IAB

        If BTorNOT = True Then
            assList.Add(10, {biotite, amesite, WhiteMica, anorthite})     '    IC
            assList.Add(11, {biotite, amesite, FM, actinote})     '     ID
            assList.Add(12, {biotite, WhiteMica, Kfeld, anorthite})     '      IE
            assList.Add(13, {amesite, actinote, biotite, anorthite})     '     IG
            assList.Add(14, {biotite, Kfeld, actinote, anorthite})     '       IG-inner
        Else
            assList.Add(10, {amesite, WhiteMica, Kfeld, anorthite})     '      IC-bis
            assList.Add(11, {amesite, FM, Kfeld, actinote})     '       ID-bis
            assList.Add(12, {amesite, actinote, Kfeld, anorthite})     '       IG-bis
        End If
```

```
ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(8, {FSP06, Kfeld, FM, CPX})     'IIE-bis-inner
    assList.Add(9, {anorthite, margarite, WhiteMica, GRThalf})     'IIG
    assList.Add(10, {WhiteMica, margarite, GRTpyalm, GRThalf})     'IIH
    assList.Add(11, {FSP06, FM, CPX, GRThalf})     'II-I
    assList.Add(12, {FSP06, clinozoisite, CPX, GRThalf})     'IIJ
    assList.Add(13, {FSP06, clinozoisite, anorthite, GRThalf})     'IIK

    If BTorNOT = True Then
        assList.Add(14, {biotite, GRTpyalm, WhiteMica, GRThalf})     'IIC
        assList.Add(15, {biotite, GRTpyalm, FM, GRThalf})     'IID
        assList.Add(16, {biotite, WhiteMica, Kfeld, anorthite})     'IIE
        assList.Add(17, {GRThalf, biotite, WhiteMica, anorthite})     'IIJ-inner
        If FeMnRatio > 0.95 Then
            assList.Add(18, {margarite, WhiteMica, disthene, staurolite})     'IIA
            assList.Add(19, {margarite, WhiteMica, GRTpyalm, staurolite})     'IIB
        Else
            assList.Add(18, {margarite, WhiteMica, GRTpyalm, disthene})     'IIAB
        End If
    Else
        assList.Add(14, {GRThalf, WhiteMica, Kfeld, GRTpyalm})     'IIC-bis-2
        assList.Add(15, {GRThalf, WhiteMica, Kfeld, anorthite})     'IIC-bis
        assList.Add(16, {GRTpyalm, FM, Kfeld, GRThalf})     'IID-bis
        If FeMnRatio > 0.95 Then
            assList.Add(17, {margarite, WhiteMica, disthene, staurolite})     'IIA
            assList.Add(18, {margarite, WhiteMica, GRTpyalm, staurolite})     'IIB
        Else
            assList.Add(17, {margarite, WhiteMica, GRTpyalm, disthene})     'IIAB
        End If
    End If

ElseIf TetraLocation = enumTetraedre.TetraIII Then
    assList.Add(8, {FSP06, FM, CPX, GRThalf})     'II-I
    assList.Add(9, {FSP06, clinozoisite, CPX, GRThalf})     'IIJ
    assList.Add(10, {FSP06, clinozoisite, anorthite, GRThalf})     'IIK
```

```
        If BTorNOT = True Then
            assList.Add(11, {margarite, WhiteMica, amesite, disthene})     'IIAB
            assList.Add(12, {GRTpyalm, amesite, margarite, biotite})     'IIC
            assList.Add(13, {margarite, amesite, biotite, WhiteMica})     'IIBC
            assList.Add(14, {biotite, GRTpyalm, FM, GRThalf})     'IID
            assList.Add(15, {biotite, WhiteMica, Kfeld, anorthite})     'IIE
            assList.Add(16, {GRThalf, biotite, WhiteMica, anorthite})     'IIJ-inner
            assList.Add(17, {FSP06, Kfeld, FM, CPX})     'IIE-bis-inner
            assList.Add(18, {anorthite, margarite, WhiteMica, GRThalf})     'IIG
            assList.Add(19, {biotite, margarite, GRTpyalm, GRThalf})     'IIH
            assList.Add(20, {GRThalf, biotite, margarite, WhiteMica})     'IIE-inner
        Else
            assList.Add(11, {margarite, WhiteMica, amesite, disthene})     'IIAB
            assList.Add(12, {FSP06, Kfeld, FM, CPX})     'IIE-bis-inner
            assList.Add(13, {anorthite, margarite, WhiteMica, GRThalf})     'IIG
            assList.Add(14, {WhiteMica, margarite, amesite, GRThalf})     'IIH
            assList.Add(15, {GRThalf, WhiteMica, Kfeld, amesite})     'IIC-bis-2
            assList.Add(16, {GRThalf, WhiteMica, Kfeld, anorthite})     'IIC-bis
            assList.Add(17, {GRTpyalm, amesite, Kfeld, GRThalf})     'IID-bis
            assList.Add(18, {GRTpyalm, FM, Kfeld, GRThalf})     'IID-bis
        End If
    End If

    Return assList
End Function

Public Function Silicates10GRA950() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {FSP07, grossular, anorthite, CPX})     '     IIIA
    assList.Add(3, {FSP07, grossular, wollastonite, CPX})     '  IIIB
    assList.Add(4, {FSP07, Kfeld, wollastonite, CPX})     '      IIIC

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(5, {cordierite, Kfeld, anorthite, sillimanite})     '      IA
```

```
        assList.Add(6, {anorthite, Kfeld, CPX, FM})     ' ID
        If NaRatio < 0.25 And MgMnRatio > 0.6 And MgFeRatio > 0.82 Then
            assList.Add(7, {cordierite, biotite, Kfeld, anorthite})     '      IB
            assList.Add(8, {cordierite, biotite, FM, anorthite})     '  IC
        Else
            assList.Add(7, {cordierite, FM, Kfeld, anorthite})     '    IBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(5, {GRTpyalm, Kfeld, anorthite, sillimanite})     ' IIA
        assList.Add(6, {anorthite, Kfeld, CPXgrt, FM})     '     IID
        If NaRatio < 0.25 And MgMnRatio > 0.6 And MgFeRatio > 0.82 Then
            assList.Add(7, {GRTpyalm, biotite, Kfeld, anorthite})     ' IIB
            assList.Add(8, {GRTss, biotite, FM, anorthite})     '       IIC
        Else
            assList.Add(7, {GRTss, FM, Kfeld, anorthite})     '  IIBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(5, {cordierite, Kfeld, anorthite, sillimanite})     'IA
        assList.Add(6, {anorthite, Kfeld, CPX, FM})     'ID

        If NaRatio < 0.25 And MgMnRatio > 0.6 And MgFeRatio > 0.82 Then
            assList.Add(7, {GRTss, FM, biotite, anorthite})     'IVB - avec Bt
            assList.Add(8, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
            assList.Add(9, {biotite, cordierite, Kfeld, anorthite})     'IVF
        Else
            assList.Add(7, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC - sans Bt
            assList.Add(8, {GRTss, FM, Kfeld, anorthite})     'IVB
        End If
    End If

    Return assList
End Function

Public Function Silicates9GRA800() As Dictionary(Of Integer, Array)
```

```
Dim assList As New Dictionary(Of Integer, Array)

assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
assList.Add(2, {FSP08, grossular, anorthite, CPX})     'IIIA
assList.Add(3, {FSP08, grossular, wollastonite, CPX})     'IIIB
assList.Add(4, {FSP08, Kfeld, wollastonite, CPX})     'IIIC

If TetraLocation = enumTetraedre.TetraI Then
    assList.Add(5, {actinote, CPX, Kfeld, anorthite})     'IC

    If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.4 Then
        assList.Add(6, {sillimanite, Kfeld, anorthite, biotite})     'IA
        assList.Add(7, {biotite, cordierite, FM, actinote})     'IB
        assList.Add(8, {sillimanite, anorthite, biotite, cordierite})     'ID
        assList.Add(9, {actinote, cordierite, anorthite, biotite})     'IE
        assList.Add(10, {biotite, Kfeld, actinote, anorthite})     'IE-inner
    Else
        assList.Add(6, {actinote, cordierite, Kfeld, anorthite})     'IE-bis
        assList.Add(7, {Kfeld, anorthite, sillimanite, cordierite})     'ID-bis
        assList.Add(8, {actinote, Kfeld, FM, cordierite})     'IB-bis
    End If

ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(5, {GRTpyalm, Kfeld, anorthite, sillimanite})     'IIA
    assList.Add(6, {anorthite, Kfeld, CPXgrt, FM})     'IID
    If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.4 Then
        assList.Add(7, {GRTss, biotite, Kfeld, anorthite})     'IIB
        assList.Add(8, {GRTss, biotite, FM, anorthite})     'IIC
    Else
        assList.Add(7, {GRTss, FM, Kfeld, anorthite})     'IIBC
    End If

ElseIf TetraLocation = enumTetraedre.TetraIII Then
    assList.Add(5, {cordierite, Kfeld, anorthite, sillimanite})     'IIA
    assList.Add(6, {anorthite, Kfeld, CPX, actinote})     'IID-1
    assList.Add(7, {anorthite, Kfeld, actinote, FM})     'IID-2
```

```
        If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.4 Then
            assList.Add(8, {GRTss, FM, biotite, anorthite})     'IVB
            assList.Add(9, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
            assList.Add(10, {biotite, cordierite, Kfeld, anorthite})     'IVF
        Else
            assList.Add(8, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
            assList.Add(9, {GRTss, FM, Kfeld, anorthite})     'IVB
        End If
    End If

    Return assList
End Function

Public Function Silicates10AMP700() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If NaRatio > 0.9 And MgMnRatio < 0.05 And MgFeRatio < 0.55 Then
        BTorNOT = False
    End If
    Dim WMorNOT As Boolean = True
    If NaRatio > 0.2 Then
        WMorNOT = False
    End If

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {grossular, clinozoisite, FSP09, CPX})     'IIIC
    assList.Add(3, {grossular, wollastonite, Kfeld, CPX})     'IIIE
    assList.Add(4, {grossular, FSP09, Kfeld, CPX})     'IIID

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(5, {CPX, FSP09, Kfeld, actinote})     'IIIF-inner
        assList.Add(6, {anorthite, FSP09, clinozoisite, actinote})     'IE
        assList.Add(7, {CPX, FSP09, clinozoisite, actinote})     'IF

        If BTorNOT = False Then
```

```
        assList.Add(8, {disthene, anorthite, Kfeld, actinote})     'IA-bis
        assList.Add(9, {FM, disthene, Kfeld, actinote})     'IBC-bis
    Else
        assList.Add(8, {FM, actinote, disthene, biotite})     'IC
        assList.Add(9, {actinote, anorthite, disthene, biotite})     'ID
        assList.Add(10, {biotite, actinote, Kfeld, anorthite})     'ID-inner
        If WMorNOT = True Then
            assList.Add(11, {WhiteMica, biotite, disthene, anorthite})     'IA
            assList.Add(12, {WhiteMica, biotite, Kfeld, anorthite})     'IB
        Else
            assList.Add(11, {Kfeld, biotite, disthene, anorthite})     'IAB
        End If
    End If

Else                        'tetraedre02
    assList.Add(5, {CPXgrt, FM, Kfeld, anorthite})     'IIE
    assList.Add(6, {GROSSss, clinozoisite, FSP09, CPX})     'IIF
    assList.Add(7, {GROSSss, clinozoisite, FSP09, anorthite})     'IIG
    If WMorNOT = True Then
        assList.Add(8, {disthene, WhiteMica, anorthite, GRTpyalm})     'IIA
        assList.Add(9, {Kfeld, WhiteMica, anorthite, GRTpyalm})     'IIB
        If BTorNOT = True Then
            assList.Add(10, {Kfeld, biotite, GRTss, anorthite})     'IIC
            assList.Add(11, {FM, biotite, GRTss, anorthite})     'IID
        Else
            assList.Add(10, {Kfeld, FM, GRTss, anorthite})     'IICD
        End If
    Else
        assList.Add(8, {Kfeld, disthene, anorthite, GRTpyalm})     'IIAB
        If BTorNOT = True Then
            assList.Add(9, {Kfeld, biotite, GRTss, anorthite})     'IIC
            assList.Add(10, {FM, biotite, GRTss, anorthite})     'IID
        Else
            assList.Add(9, {Kfeld, FM, GRTss, anorthite})     'IICD
        End If
    End If
End If
```

```
    Return assList
End Function

Public Function Silicates7AMP500() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If MgMnRatio < 0.05 And FeMnRatio < 0.45 Then
        BTorNOT = False
    ElseIf NaRatio > 0.9 Then
        BTorNOT = False
    End If

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {clinozoisite, Kfeld, grossular, CPX})     'IIIA
    assList.Add(3, {wollastonite, Kfeld, grossular, CPX})     'IIIB

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(4, {margarite, WhiteMica, disthene, amesite})     'IA
        assList.Add(5, {clinozoisite, Kfeld, actinote, CPX})     'IG
        assList.Add(6, {clinozoisite, WhiteMica, margarite, amesite})     'IE

        If BTorNOT = True Then
            assList.Add(7, {FM, actinote, biotite, amesite})     'IB
            assList.Add(8, {clinozoisite, WhiteMica, biotite, amesite})     'IC
            assList.Add(9, {clinozoisite, WhiteMica, biotite, Kfeld})     'ID
            assList.Add(10, {clinozoisite, biotite, actinote, amesite})     'IF
            assList.Add(11, {actinote, biotite, Kfeld, clinozoisite})     'IF-inner
        Else
            assList.Add(7, {clinozoisite, Kfeld, actinote, amesite})     'IF-bis
            assList.Add(8, {clinozoisite, WhiteMica, Kfeld, amesite})     'IB-bis
            assList.Add(9, {actinote, FM, Kfeld, amesite})     'ID-bis
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
```

```
        assList.Add(4, {chloritoide, disthene, WhiteMica, margarite})     'IIA
        assList.Add(5, {WhiteMica, chloritoide, daphnite, GRThalf})     'IIB
        assList.Add(6, {daphnite, GRThalf, FM, WhiteMica})     'IIC
        assList.Add(7, {clinozoisite, chloritoide, WhiteMica, margarite})     'IIF
        assList.Add(8, {clinozoisite, GRThalf, chloritoide, WhiteMica})     'IIG

        If BTorNOT = True Then
            assList.Add(9, {GRThalf, FM, biotite, WhiteMica})     'IID
            assList.Add(10, {clinozoisite, Kfeld, biotite, WhiteMica})     'IIE
            assList.Add(11, {CPX, GRThalf, FM, biotite})     'IIH
            assList.Add(12, {CPX, GRThalf, clinozoisite, biotite})     'II-I
            assList.Add(13, {GRThalf, clinozoisite, biotite, WhiteMica})     'IIC-inner
            assList.Add(14, {biotite, CPX, Kfeld, clinozoisite})     'IIB-inner
        Else
            assList.Add(9, {GRThalf, Kfeld, FM, WhiteMica})     'IID-bis
            assList.Add(10, {Kfeld, clinozoisite, WhiteMica, GRThalf})     'IIE-bis
            assList.Add(11, {Kfeld, GRThalf, CPX, clinozoisite})     'II-I-bis
            assList.Add(12, {Kfeld, GRThalf, CPX, FM})     'IIH-bis
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(4, {GRTpyalm, disthene, WhiteMica, margarite})     'IIA
        assList.Add(5, {clinozoisite, FM, GRTpyalm, WhiteMica})     'IIGCB
        assList.Add(6, {clinozoisite, GRTpyalm, WhiteMica, margarite})     'IIF

        If BTorNOT = True Then
            assList.Add(7, {clinozoisite, FM, biotite, WhiteMica})     'IID
            assList.Add(8, {clinozoisite, Kfeld, biotite, WhiteMica})     'IIE
            assList.Add(9, {CPX, actinote, clinozoisite, biotite})     'II-HI
            assList.Add(10, {FM, actinote, clinozoisite, biotite})     'II-HI-2
            assList.Add(11, {biotite, CPX, Kfeld, clinozoisite})     'IIB-inner
        Else
            assList.Add(12, {clinozoisite, Kfeld, FM, WhiteMica})     'IID-bis
            assList.Add(13, {Kfeld, actinote, CPX, clinozoisite})     'II-HI-bis
            assList.Add(14, {Kfeld, FM, actinote, clinozoisite})     'II-HI-bis-2
        End If
    End If
```

```
    Return assList
End Function

Public Function Silicates4AMP600() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If MgMnRatio < 0.05 And FeMnRatio < 0.8 Then
        BTorNOT = False
    ElseIf NaRatio > 0.9 Then
        BTorNOT = False
    End If

    Dim WMorNOT As Boolean = True
    If NaRatio > 0.2 Then
        WMorNOT = False
    End If

    Dim STAUorNOT As Boolean = False
    If MgFeRatio < 0.12 And FeMnRatio > 0.94 Then
        STAUorNOT = True
    End If

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})      ' impossible
    assList.Add(2, {FSP11, CPX, anorthite, grossular})      'IIIA
    assList.Add(3, {FSP11, CPX, wollastonite, grossular})      'IIIB
    assList.Add(4, {FSP11, CPX, wollastonite, Kfeld})      'IIIC

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(5, {actinote, CPX, anorthite, Kfeld})      'IH-all
        assList.Add(6, {actinote, anorthite, FM, Kfeld})      'IFG-all
        If BTorNOT = True And WMorNOT = True Then
            assList.Add(7, {cordierite, anorthite, WhiteMica, sillimanite})      'IA
            assList.Add(8, {WhiteMica, Kfeld, anorthite, biotite})      'IB
            assList.Add(9, {biotite, cordierite, WhiteMica, anorthite})      'IC
```

```
            assList.Add(10, {biotite, cordierite, anorthite, FM})      'IDE
        Else
            If WMorNOT = True Then
                assList.Add(7, {cordierite, anorthite, WhiteMica, sillimanite})  'IA-bis
                assList.Add(8, {WhiteMica, Kfeld, anorthite, cordierite})      'IB-biss
                If BTorNOT = True Then
                    assList.Add(9, {cordierite, biotite, Kfeld, anorthite})      'IC-bis
                    assList.Add(10, {cordierite, biotite, anorthite, FM})      'IDE-bis
                Else
                    assList.Add(9, {cordierite, Kfeld, FM, anorthite})      'ICDE-bis
                End If
            Else
                assList.Add(7, {cordierite, Kfeld, sillimanite, anorthite})      'IAB-bis
                If BTorNOT = True Then
                    assList.Add(8, {cordierite, biotite, Kfeld, anorthite})      'IC-bis
                    assList.Add(9, {cordierite, biotite, anorthite, FM})      'IDE-bis
                Else
                    assList.Add(8, {cordierite, Kfeld, FM, anorthite})      'ICDE-bis
                End If
            End If
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(5, {CPX, anorthite, FM, Kfeld})      'IIF-all
        If BTorNOT = True And WMorNOT = True Then
            assList.Add(6, {GRTss, anorthite, FM, biotite})      'IIC
            assList.Add(7, {biotite, GRTpyalm, WhiteMica, anorthite})      'IID
            assList.Add(8, {biotite, Kfeld, WhiteMica, anorthite})      'IIE
            If STAUorNOT = True Then
                assList.Add(9, {staurolite, anorthite, WhiteMica, sillimanite})      'IIA
                assList.Add(10, {staurolite, GRTpyalm, WhiteMica, anorthite})      'IIB
            Else
                assList.Add(9, {sillimanite, GRTpyalm, WhiteMica, anorthite})      'IIAB
            End If
        Else
            If STAUorNOT = True Then
                If BTorNOT = True Then
```

```
            assList.Add(6, {staurolite, GRTpyalm, Kfeld, anorthite})      'IIC-bis
            assList.Add(7, {FM, GRTss, biotite, anorthite})      'IID-bis
            assList.Add(8, {biotite, GRTpyalm, Kfeld, anorthite})      'IIE-bis
            If WMorNOT = True Then
                assList.Add(9, {staurolite, anorthite, WhiteMica, sillimanite})      'IIA-bis
                assList.Add(10, {staurolite, anorthite, WhiteMica, Kfeld})      'IIB-bis
            Else
                assList.Add(9, {GRTpyalm, anorthite, Kfeld, sillimanite})      'IIABC-bis
            End If
        Else
            assList.Add(6, {FM, GRTss, Kfeld, anorthite})      'IIED-bis
            assList.Add(7, {staurolite, GRTpyalm, Kfeld, anorthite})      'IIC-bis2
            If WMorNOT = True Then
                assList.Add(8, {staurolite, anorthite, WhiteMica, sillimanite})      'IIA-bis
                assList.Add(9, {staurolite, anorthite, WhiteMica, Kfeld})      'IIB-bis
            Else
                assList.Add(8, {GRTpyalm, anorthite, Kfeld, sillimanite})      'IIABC-bis
            End If
        End If
    Else
        If WMorNOT = True Then
            assList.Add(6, {GRTpyalm, anorthite, WhiteMica, sillimanite})      'IIA-bis-No STAU
            assList.Add(7, {GRTpyalm, anorthite, WhiteMica, Kfeld})      'IIB-bis-No STAU
            If BTorNOT = True Then
                assList.Add(8, {FM, GRTss, biotite, anorthite})      'IID-bis
                assList.Add(9, {biotite, GRTpyalm, Kfeld, anorthite})      'IIE-bis
            Else
                assList.Add(8, {FM, GRTss, Kfeld, anorthite})      'IIED-bis
            End If
        Else
            assList.Add(6, {GRTpyalm, anorthite, Kfeld, sillimanite})      'IIABC-bis
            If BTorNOT = True Then
                assList.Add(7, {FM, GRTss, biotite, anorthite})      'IID-bis
                assList.Add(8, {biotite, GRTpyalm, Kfeld, anorthite})      'IIE-bis
            Else
                assList.Add(7, {FM, GRTss, Kfeld, anorthite})      'IIED-bis
            End If
```

```
                End If
            End If
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(5, {actinote, CPX, anorthite, Kfeld})      'IH-all
        assList.Add(6, {actinote, anorthite, FM, Kfeld})      'IFG-all
        If BTorNOT = True Then
            assList.Add(7, {GRTss, FM, biotite, anorthite})      'IVB
            assList.Add(8, {GRTpyalm, cordierite, biotite, anorthite})      'IVC
            assList.Add(9, {biotite, cordierite, Kfeld, anorthite})      'IVF
            If WMorNOT = True Then
                assList.Add(10, {cordierite, anorthite, WhiteMica, sillimanite})      'IIA-bis
                assList.Add(11, {cordierite, anorthite, WhiteMica, Kfeld})      'IIB-bis
            Else
                assList.Add(10, {cordierite, anorthite, Kfeld, sillimanite})      'IIAB-bis- no WM
            End If
        Else
            assList.Add(7, {GRTpyalm, cordierite, Kfeld, anorthite})      'IVC  - no Bt
            assList.Add(8, {GRTss, FM, Kfeld, anorthite})     'IVB - no Bt
            If WMorNOT = True Then
                assList.Add(9, {cordierite, anorthite, WhiteMica, sillimanite})      'IIA-bis
                assList.Add(10, {cordierite, anorthite, WhiteMica, Kfeld})      'IIB-bis
            Else
                assList.Add(9, {cordierite, anorthite, Kfeld, sillimanite})      'IIAB-bis- no WM
            End If
        End If
    End If

    Return assList
End Function

Public Function Silicates5SV400() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
```

```
If MgMnRatio < 0.05 And FeMnRatio < 0.3 Then
    BTorNOT = False
ElseIf NaRatio > 0.9 Then
    BTorNOT = False
End If

assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
assList.Add(2, {grossular, Kfeld, CPX, wollastonite})      'IIIA
assList.Add(3, {grossular, Kfeld, CPX, clinozoisite})      'IIIB

If TetraLocation = enumTetraedre.TetraI Then
    assList.Add(4, {actinote, Kfeld, CPX, clinozoisite})      'IH-all
    assList.Add(5, {margarite, WhiteMica, sudoite, pyrophyllite})      'IA-all
    assList.Add(6, {margarite, WhiteMica, sudoite, amesite})      'IB-all
    assList.Add(7, {amesite, margarite, WhiteMica, clinozoisite})      'IF-all

    If BTorNOT = True Then
        assList.Add(8, {biotite, amesite, FM, actinote})      'IC
        assList.Add(9, {biotite, amesite, WhiteMica, actinote})      'ID
        assList.Add(10, {biotite, Kfeld, WhiteMica, clinozoisite})      'IE
        assList.Add(11, {amesite, actinote, WhiteMica, clinozoisite})      'IG
        assList.Add(12, {biotite, actinote, Kfeld, clinozoisite})      'IG-inner  - HOLE HERE
    Else
        assList.Add(8, {Kfeld, amesite, FM, actinote})      'IC-bis
        assList.Add(9, {Kfeld, WhiteMica, amesite, clinozoisite})      'IE-bis
        assList.Add(10, {amesite, actinote, Kfeld, clinozoisite})      'IG-bis
    End If

ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(4, {chloritoide, margarite, WhiteMica, pyrophyllite})      'IIA-all
    assList.Add(5, {chloritoide, daphnite, clinozoisite, WhiteMica})      'IIB-all
    assList.Add(6, {chloritoide, margarite, clinozoisite, WhiteMica})      'IIF-all

    If BTorNOT = True Then
        assList.Add(7, {daphnite, biotite, CPX, FM})      'IIC
        assList.Add(8, {daphnite, biotite, WhiteMica, clinozoisite})      'IID
        assList.Add(9, {Kfeld, biotite, WhiteMica, clinozoisite})      'IIE
```

```
            assList.Add(10, {daphnite, CPX, biotite, clinozoisite})      'IIG
            assList.Add(11, {CPX, biotite, Kfeld, clinozoisite})      'IIG-inner
        Else
            assList.Add(7, {daphnite, Kfeld, FM, CPX})      'IIC-bis
            assList.Add(8, {daphnite, Kfeld, WhiteMica, clinozoisite})      'IIE-bis
            assList.Add(9, {daphnite, CPX, Kfeld, clinozoisite})      'IIG-bis
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(4, {GRTpyalm, margarite, WhiteMica, pyrophyllite})      'IIA-all
        assList.Add(5, {GRTpyalm, margarite, clinozoisite, WhiteMica})      'IIF-all
        If BTorNOT = True Then
            assList.Add(6, {GRTpyalm, biotite, CPX, FM})      'IIC
            assList.Add(7, {GRTpyalm, biotite, WhiteMica, clinozoisite})      'IID
            assList.Add(8, {Kfeld, biotite, WhiteMica, clinozoisite})      'IIE
            assList.Add(9, {GRTpyalm, CPX, biotite, clinozoisite})      'IIG
            assList.Add(10, {CPX, biotite, Kfeld, clinozoisite})      'IIG-inner
        Else
            assList.Add(6, {GRTpyalm, Kfeld, FM, CPX})      'IIC-bis
            assList.Add(7, {GRTpyalm, Kfeld, WhiteMica, clinozoisite})      'IIE-bis
            assList.Add(8, {GRTpyalm, CPX, Kfeld, clinozoisite})      'IIG-bis
        End If
    End If

    Return assList
End Function

Public Function Silicates9SB450() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {grossular, Kfeld, CPX, wollastonite})     'IIIA
    assList.Add(3, {grossular, Kfeld, CPX, clinozoisite})     'IIIB

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(4, {actinote, Kfeld, CPX, clinozoisite})     'IH
```

```
        assList.Add(5, {margarite, WhiteMica, sudoite, disthene})     'IA
        assList.Add(6, {margarite, WhiteMica, sudoite, amesite})     'IB
        assList.Add(7, {amesite, margarite, WhiteMica, clinozoisite})     'IF
        assList.Add(8, {biotite, amesite, FM, actinote})     'IC
        assList.Add(9, {biotite, amesite, WhiteMica, clinozoisite})     'IE
        assList.Add(10, {biotite, Kfeld, WhiteMica, clinozoisite})     'ID
        assList.Add(11, {amesite, actinote, biotite, clinozoisite})     'IG
        assList.Add(12, {biotite, actinote, Kfeld, clinozoisite})     'IG-inner

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(4, {margarite, WhiteMica, chloritoide, disthene})     'IIA
        assList.Add(5, {daphnite, WhiteMica, chloritoide, clinozoisite})     'IIB
        assList.Add(6, {daphnite, biotite, FM, CPX})     'IIC
        assList.Add(7, {biotite, WhiteMica, Kfeld, clinozoisite})     'IID
        assList.Add(8, {biotite, daphnite, WhiteMica, clinozoisite})     'IIE
        assList.Add(9, {chloritoide, margarite, WhiteMica, clinozoisite})     'IIF
        assList.Add(10, {daphnite, GRThalf, clinozoisite, biotite})     'IIG
        assList.Add(11, {daphnite, GRThalf, CPX, biotite})     'IIH
        assList.Add(12, {clinozoisite, GRThalf, CPX, biotite})     'II-I
        assList.Add(13, {biotite, CPX, Kfeld, clinozoisite})     'IIA-inner

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(4, {margarite, WhiteMica, GRTpyalm, disthene})     'IIA
        assList.Add(5, {GRTpyalm, biotite, FM, CPX})     'IIC
        assList.Add(6, {biotite, WhiteMica, Kfeld, clinozoisite})     'IID
        assList.Add(7, {biotite, GRTpyalm, WhiteMica, clinozoisite})     'IIE
        assList.Add(8, {GRTpyalm, margarite, WhiteMica, clinozoisite})     'IIF
        assList.Add(9, {biotite, CPX, Kfeld, clinozoisite})     'IIA-inner
        assList.Add(10, {GRTpyalm, GRThalf, clinozoisite, biotite})     'IIG
        assList.Add(11, {GRTpyalm, GRThalf, CPX, biotite})     'IIH
        assList.Add(12, {clinozoisite, GRThalf, CPX, biotite})     'II-I
    End If

    Return assList
End Function

Public Function Silicates10SB350() As Dictionary(Of Integer, Array)
```

```
Dim assList As New Dictionary(Of Integer, Array)

assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
assList.Add(2, {grossular, Kfeld, CPX, wollastonite})     'IIIA
assList.Add(3, {grossular, Kfeld, CPX, lawsonite})     'IIIB

If TetraLocation = enumTetraedre.TetraI Then
    assList.Add(4, {actinote, Kfeld, CPX, lawsonite})     'IH
    assList.Add(5, {margarite, WhiteMica, carpholite, pyrophyllite})     'IA
    assList.Add(6, {margarite, WhiteMica, carpholite, amesite})     'IB
    assList.Add(7, {amesite, margarite, WhiteMica, lawsonite})     'IF
    assList.Add(8, {biotite, amesite, FM, actinote})     'IC
    assList.Add(9, {biotite, amesite, WhiteMica, lawsonite})     'IE
    assList.Add(10, {biotite, Kfeld, WhiteMica, lawsonite})     'ID
    assList.Add(11, {amesite, actinote, biotite, lawsonite})     'IG
    assList.Add(12, {biotite, actinote, Kfeld, lawsonite})     'IG-inner

ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(4, {margarite, WhiteMica, chloritoide, pyrophyllite})     'IIA
    assList.Add(5, {daphnite, WhiteMica, chloritoide, lawsonite})     'IIB
    assList.Add(6, {daphnite, biotite, FM, CPX})     'IIC
    assList.Add(7, {biotite, WhiteMica, Kfeld, lawsonite})     'IID
    assList.Add(8, {biotite, daphnite, WhiteMica, lawsonite})     'IIE
    assList.Add(9, {chloritoide, margarite, WhiteMica, lawsonite})     'IIF
    assList.Add(10, {daphnite, CPX, lawsonite, biotite})     'IIG
    assList.Add(11, {biotite, CPX, Kfeld, lawsonite})     'IIA-inner

ElseIf TetraLocation = enumTetraedre.TetraIII Then
    assList.Add(4, {margarite, WhiteMica, GRTpyalm, pyrophyllite})     'IIA
    assList.Add(5, {GRTpyalm, biotite, FM, CPX})     'IIC
    assList.Add(6, {biotite, WhiteMica, Kfeld, lawsonite})     'IID
    assList.Add(7, {biotite, GRTpyalm, WhiteMica, lawsonite})     'IIE
    assList.Add(8, {GRTpyalm, margarite, WhiteMica, lawsonite})     'IIF
    assList.Add(9, {GRTpyalm, CPX, lawsonite, biotite})     'IIG
    assList.Add(10, {biotite, CPX, Kfeld, lawsonite})     'IIA-inner
End If
```

```
    Return assList
End Function

Public Function Silicates4GRA750() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {anorthite, wollastonite, Kfeld, CPX})     'IIIA

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(3, {cordierite, anorthite, Kfeld, sillimanite})     'IA
        assList.Add(4, {CPX, actinote, anorthite, Kfeld})     'IE

        If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.5 Then
            assList.Add(5, {cordierite, actinote, anorthite, biotite})     'ID
            assList.Add(6, {cordierite, anorthite, Kfeld, biotite})     'IB
            assList.Add(7, {cordierite, actinote, FM, biotite})     'IC
            assList.Add(8, {biotite, actinote, Kfeld, anorthite})     'IC-inner
        Else
            assList.Add(5, {cordierite, actinote, Kfeld, FM})     'ICB
            assList.Add(6, {cordierite, actinote, anorthite, Kfeld})     'ID
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(3, {GRTpyalm, anorthite, Kfeld, sillimanite})     'IIA
        assList.Add(4, {FM, CPX, anorthite, Kfeld})     'IID
        If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.5 Then
            assList.Add(5, {GRTpyalm, anorthite, Kfeld, biotite})     'IIB
            assList.Add(6, {GRTss, anorthite, FM, biotite})     'IIC
        Else
            assList.Add(5, {GRTss, anorthite, FM, Kfeld})     'IBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(3, {cordierite, anorthite, Kfeld, sillimanite})     'IIA
```

```
        assList.Add(4, {FM, actinote, anorthite, Kfeld})     'IID
        assList.Add(5, {actinote, CPX, anorthite, Kfeld})     'IID-2

        If NaRatio < 0.9 And MgFeRatio > 0.7 And MgMnRatio > 0.5 Then
            assList.Add(6, {GRTss, FM, biotite, anorthite})     'IVB
            assList.Add(7, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
            assList.Add(8, {biotite, cordierite, Kfeld, anorthite})     'IVF
        Else
            assList.Add(6, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
            assList.Add(7, {GRTss, FM, Kfeld, anorthite})     'IVB
        End If
    End If

    Return assList
End Function

Public Function Silicates6GRA750() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
    assList.Add(2, {anorthite, grossular, Kfeld, CPX})     'IIIA
    assList.Add(3, {grossular, wollastonite, Kfeld, CPX})     'IIIB

    If TetraLocation = enumTetraedre.TetraI Then
        assList.Add(4, {cordierite, anorthite, Kfeld, sillimanite})     'IA
        assList.Add(5, {FM, actinote, anorthite, Kfeld})     'ID
        assList.Add(6, {CPX, actinote, anorthite, Kfeld})     'IE
        If NaRatio < 0.9 And MgFeRatio > 0.4 And MgMnRatio > 0.67 Then
            assList.Add(7, {cordierite, anorthite, Kfeld, biotite})     'IB
            assList.Add(8, {cordierite, anorthite, FM, biotite})     'IC
        Else
            assList.Add(7, {cordierite, anorthite, FM, Kfeld})     'IBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraII Then
        assList.Add(4, {GRTpyalm, anorthite, Kfeld, sillimanite})     'IIA
```

```
        assList.Add(5, {FM, CPXgrt, anorthite, Kfeld})     'IID
        If NaRatio < 0.9 And MgFeRatio > 0.4 And MgMnRatio > 0.67 Then
            assList.Add(6, {GRTpyalm, anorthite, Kfeld, biotite})     'IIB
            assList.Add(7, {GRTss, anorthite, FM, biotite})     'IIC
        Else
            assList.Add(6, {GRTss, anorthite, FM, Kfeld})     'IBC
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(4, {cordierite, anorthite, Kfeld, sillimanite})     'IIA
        assList.Add(5, {FM, actinote, anorthite, Kfeld})     'IID
        assList.Add(6, {actinote, CPX, anorthite, Kfeld})     'IID-2
        If NaRatio < 0.9 And MgFeRatio > 0.4 And MgMnRatio > 0.67 Then
            assList.Add(7, {GRTss, FM, biotite, anorthite})     'IVB
            assList.Add(8, {GRTpyalm, cordierite, biotite, anorthite})     'IVC
            assList.Add(9, {biotite, cordierite, Kfeld, anorthite})     'IVF
        Else
            assList.Add(7, {GRTpyalm, cordierite, Kfeld, anorthite})     'IVC
            assList.Add(8, {GRTss, FM, Kfeld, anorthite})     'IVB
        End If
    End If

    Return assList
End Function

Public Function Silicates6AMP600() As Dictionary(Of Integer, Array)

    Dim assList As New Dictionary(Of Integer, Array)

    Dim BTorNOT As Boolean = True
    If MgMnRatio < 0.05 And FeMnRatio < 0.7 Then
        BTorNOT = False
    ElseIf NaRatio > 0.9 Then
        BTorNOT = False
    End If

    Dim STAUorNOT As Boolean = False
```

```
If MgFeRatio < 0.15 And FeMnRatio > 0.74 Then
    STAUorNOT = True
End If

assList.Add(1, {NaK, Kfeld, FM, wollastonite})     '  impossible
assList.Add(2, {wollastonite, grossular, CPX, Kfeld})     'IIIA
assList.Add(3, {FSP17, grossular, CPX, Kfeld})     'IIIB
assList.Add(4, {FSP17, grossular, CPX, clinozoisite})     'IIIC

If TetraLocation = enumTetraedre.TetraI Then
    assList.Add(5, {FSP17, anorthite, CPX, clinozoisite})     'I-I
    assList.Add(6, {amesite, anorthite, WhiteMica, disthene})     'IA
    assList.Add(7, {actinote, anorthite, FM, Kfeld})     'IEF
    assList.Add(8, {actinote, CPX, anorthite, Kfeld})     'IG
    If BTorNOT = True Then
        assList.Add(9, {amesite, biotite, FM, anorthite})     'IBH
        assList.Add(10, {biotite, amesite, WhiteMica, anorthite})     'IC
        assList.Add(11, {biotite, Kfeld, WhiteMica, anorthite})     'ID
    Else
        assList.Add(9, {Kfeld, amesite, FM, anorthite})     'IBH-bis
        assList.Add(10, {Kfeld, amesite, WhiteMica, anorthite})     'ID-bis
    End If

ElseIf TetraLocation = enumTetraedre.TetraII Then
    assList.Add(5, {CPX, FSP17, GRThalf, FM})     'IIF
    assList.Add(6, {CPX, FSP17, GRThalf, clinozoisite})     'IIG
    assList.Add(7, {anorthite, FSP17, GRThalf, clinozoisite})     'IIH
    If BTorNOT = True Then
        assList.Add(8, {GRTss, anorthite, FM, biotite})     'IIC - Bt
        assList.Add(9, {GRTpyalm, biotite, WhiteMica, anorthite})     'IID - Bt
        assList.Add(10, {Kfeld, biotite, WhiteMica, anorthite})     'IIE - Bt
        assList.Add(11, {CPX, biotite, Kfeld, FSP17})     'IIE - INNER
        If STAUorNOT = True Then
            assList.Add(12, {staurolite, anorthite, WhiteMica, disthene})     'IIA-with STAU
            assList.Add(13, {GRTpyalm, anorthite, WhiteMica, staurolite})     'IIB-with STAU
        Else
            assList.Add(12, {GRTpyalm, anorthite, WhiteMica, disthene})     'IIAB-no STAU
```

```
            End If
        Else
            assList.Add(8, {Kfeld, GRTss, FM, anorthite})     'IIC - NO Bt
            assList.Add(9, {GRTpyalm, Kfeld, WhiteMica, anorthite})     'IID - NO Bt
            assList.Add(10, {CPXgrt, FM, Kfeld, FSP17})     'IIE - INNER
            If STAUorNOT = True Then
                assList.Add(11, {staurolite, anorthite, WhiteMica, disthene})     'IIA-with STAU
                assList.Add(12, {GRTpyalm, anorthite, WhiteMica, staurolite})     'IIB-with STAU
            Else
                assList.Add(11, {GRTpyalm, anorthite, WhiteMica, disthene})     'IIAB-no STAU
            End If
        End If

    ElseIf TetraLocation = enumTetraedre.TetraIII Then
        assList.Add(5, {CPXgrt, FSP17, GRThalf, FM})     'IIF
        assList.Add(6, {CPXgrt, FSP17, GRThalf, clinozoisite})     'IIG
        assList.Add(7, {anorthite, FSP17, GRThalf, clinozoisite})     'IIH
        assList.Add(8, {cordierite, anorthite, WhiteMica, disthene})     'IIAB-no STAU
        If BTorNOT = True Then
            assList.Add(9, {cordierite, biotite, WhiteMica, anorthite})     'IID - Bt
            assList.Add(10, {Kfeld, biotite, WhiteMica, anorthite})     'IIE - Bt
            assList.Add(11, {CPXgrt, biotite, Kfeld, FSP17})     'IIE - INNER
            assList.Add(12, {cordierite, anorthite, GRTpyalm, biotite})     'IIC - Bt
            assList.Add(13, {GRTpyalm, anorthite, FM, biotite})     'IIC - Bt-2
        Else
            assList.Add(9, {Kfeld, WhiteMica, cordierite, anorthite})     'IIE - NO Bt
            assList.Add(10, {Kfeld, GRTss, FM, anorthite})     'IIC - NO Bt
            assList.Add(11, {GRTpyalm, Kfeld, cordierite, anorthite})     'IID - NO Bt
            assList.Add(12, {CPXgrt, FM, Kfeld, FSP17})     'IIE - INNER
        End If
    End If

    Return assList
End Function

#End Region
```

```
#Region "LocationInAFM"

    'These classes define the Al-Fe-Mg-Mn tetraedres used to select silicates' tetraedres I, II or III.

    Public Sub LocateTetraedre()

        Dim assemblageToTestNew As New ArrayList
        Dim mineralsToTestNew As New ArrayList

        If FaciesEnum = FormMain.enumFacies.facies10AMP700 Then
            If MgMnRatio > 0.73 And MgFeRatio > 0.7 Then : TetraLocation = enumTetraedre.TetraI
            Else : TetraLocation = enumTetraedre.TetraII
            End If
        ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP500 Then
            If MgMnRatio < 0.9 And FeMnRatio < 0.8 Then : TetraLocation = enumTetraedre.TetraIII
            Else
                If MgFeRatio < 0.05 Then : TetraLocation = enumTetraedre.TetraII
                Else : TetraLocation = enumTetraedre.TetraI
                End If
            End If
        ElseIf FaciesEnum = FormMain.enumFacies.facies5SV400 Then
            If MgMnRatio < 0.9 And FeMnRatio < 0.9 Then : TetraLocation = enumTetraedre.TetraIII
            Else
                If MgFeRatio < 0.05 Then : TetraLocation = enumTetraedre.TetraII
                Else : TetraLocation = enumTetraedre.TetraI
                End If
            End If
        ElseIf FaciesEnum = FormMain.enumFacies.facies9SB450 Then
            If MgMnRatio < 0.9 And FeMnRatio < 0.75 Then : TetraLocation = enumTetraedre.TetraIII
            Else
                If MgFeRatio < 0.12 Then : TetraLocation = enumTetraedre.TetraII
                Else : TetraLocation = enumTetraedre.TetraI
                End If
            End If
        ElseIf FaciesEnum = FormMain.enumFacies.facies10SB350 Then
            If MgMnRatio < 0.9 And FeMnRatio < 0.9 Then : TetraLocation = enumTetraedre.TetraIII
            Else
```

```
        If MgFeRatio < 0.1 Then : TetraLocation = enumTetraedre.TetraII
        Else : TetraLocation = enumTetraedre.TetraI
        End If
    End If

Else
    ASSEMBsilicateAFM.Clear()
    assemblageMxAFM.Clear()

    If FaciesEnum = FormMain.enumFacies.facies3AMP675 Then
        Facies01selectCoordinates()
        Facies01selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies3GRA800 Then
        Facies02selectCoordinates()
        Facies02selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies5GRA900 Then
        Facies03selectCoordinates()
        Facies03selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies5AMP670 Then
        Facies04selectCoordinates()
        Facies04selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP700 Then
        Facies05selectCoordinates()
        Facies05selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies7AMP600 Then
        Facies06selectCoordinates()
        Facies06selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies10GRA950 Then
        Facies07selectCoordinates()
        Facies07selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies9GRA800 Then
        Facies08selectCoordinates()
        Facies08selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies4AMP600 Then
        Facies11selectCoordinates()
        Facies11selectName()
    ElseIf FaciesEnum = FormMain.enumFacies.facies4GRA750 Then
```

```
                Facies15selectCoordinates()
                Facies15selectName()
            ElseIf FaciesEnum = FormMain.enumFacies.facies6GRA750 Then
                Facies16selectCoordinates()
                Facies16selectName()
            ElseIf FaciesEnum = FormMain.enumFacies.facies6AMP600 Then
                Facies17selectCoordinates()
                Facies17selectName()
            End If

            mxToBeFormed(mole("Al").NBmole, mole("Fe2").NBmole, mole("Mg").NBmole, mole("Mn").NBmole, "TetraedreAFM")
        End If

    End Sub

    Private Sub Facies01selectCoordinates()
        ASSEMBsilicateAFM.Add(0, {{0, 0.25, 0.75, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {0.6667, 0, 0.2766, 0.0567}}) '   IA
        ASSEMBsilicateAFM.Add(1, {{0, 0.25, 0.75, 0}, {2 / 3, 0, 1 / 3, 0}, {2 / 3, 0.0666, 0.2664, 0}, {0.6667, 0, 0.2766, 0.0567}})
'       IB
        ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {2 / 3, 0.0666, 0.2664, 0}, {0.6667, 0, 0.2766, 0.0567}, {1, 0, 0, 0}}) '  IC
        ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.25, 0.75, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '     IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.25, 0.75, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.54, 0.06, 0}, {0.4, 0, 0, 0.6}}) '   IIC
        ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.54, 0.06, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) '  IID
    End Sub

    Private Sub Facies01selectName()
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
    End Sub

    Private Sub Facies02selectCoordinates()
```

```
        ASSEMBsilicateAFM.Add(0, {{0, 0, 1, 0}, {0, 0.3, 0.7, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 0.55, 0.45}}) 'IA
        ASSEMBsilicateAFM.Add(1, {{0.6667, 0, 0.2666, 0.0667}, {0, 0.3, 0.7, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1566, 0.1767, 0}})
'       IB
        ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1566, 0.1767, 0}, {0.6667, 0, 0.2666, 0.0667}, {1, 0, 0, 0}}) ' IC
        ASSEMBsilicateAFM.Add(3, {{0, 1, 0, 0}, {0, 0, 0.55, 0.45}, {0, 0, 0, 1}, {0.4, 0, 0, 0.6}}) ' IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.3, 0.7, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '       IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.3, 0.7, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {0.4, 0, 0, 0.6}}) '     IIC
        ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {0.4, 0, 0, 0.6}, {1, 0, 0, 0}}) '  IID
        ASSEMBsilicateAFM.Add(7, {{0, 0, 0.55, 0.45}, {0, 0.3, 0.7, 0}, {0.4, 0, 0, 0.6}, {0, 1, 0, 0}}) '    IIE
    End Sub

    Private Sub Facies02selectName()
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
    End Sub

    Private Sub Facies03selectCoordinates()
        ASSEMBsilicateAFM.Add(0, {{0, 0, 1, 0}, {0, 0.5, 0.5, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 0.4, 0.6}}) '  IA
        ASSEMBsilicateAFM.Add(1, {{0.6667, 0, 0.28, 0.0533}, {0, 0.5, 0.5, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1167, 0.2166, 0}}) '
       IB
        ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1167, 0.2166, 0}, {0.6667, 0, 0.28, 0.0533}, {1, 0, 0, 0}}) '   IC
        ASSEMBsilicateAFM.Add(3, {{0, 1, 0, 0}, {0, 0, 0.4, 0.6}, {0, 0, 0, 1}, {0.4, 0, 0, 0.6}}) '   IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.5, 0.5, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '       IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.5, 0.5, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.45, 0.15, 0}, {0.4, 0, 0, 0.6}}) '     IIC
        ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.45, 0.15, 0}, {0.4, 0, 0, 0.6}, {1, 0, 0, 0}}) '  IID
        ASSEMBsilicateAFM.Add(7, {{0, 0, 0.4, 0.6}, {0, 0.5, 0.5, 0}, {0.4, 0, 0, 0.6}, {0, 1, 0, 0}}) '       IIE
    End Sub

    Private Sub Facies03selectName()
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
```

```
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
    End Sub

    Private Sub Facies04selectCoordinates()
        ASSEMBsilicateAFM.Add(0, {{0, 0.4, 0.6, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {2 / 3, 0, 0.3, 0.1 / 3}}) ' IA
        ASSEMBsilicateAFM.Add(1, {{0, 0.4, 0.6, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1333, 0.2, 0}, {2 / 3, 0, 0.3, 0.1 / 3}}) '   IB
        ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1333, 0.2, 0}, {2 / 3, 0, 0.3, 0.1 / 3}, {1, 0, 0, 0}}) 'IC
        ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.4, 0.6, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '      IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.4, 0.6, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.516, 0.084, 0}, {0.4, 0, 0, 0.6}}) '   IIC
        ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.516, 0.084, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) 'IID
    End Sub

    Private Sub Facies04selectName()
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
    End Sub

    Private Sub Facies05selectCoordinates()
        ASSEMBsilicateAFM.Add(0, {{0, 0.45, 0.55, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {2 / 3, 0, 0.3, 0.1 / 3}})     '  IA
        ASSEMBsilicateAFM.Add(1, {{0, 0.45, 0.55, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.0667, 0.2666, 0}, {2 / 3, 0, 0.3, 0.1 / 3}})
'       IB
        ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.0667, 0.2666, 0}, {2 / 3, 0, 0.3, 0.1 / 3}, {1, 0, 0, 0}})     '
       IC
        ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}})     '   IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.45, 0.55, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}})     'IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.45, 0.55, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.39, 0.21, 0}, {0.4, 0, 0, 0.6}})     '      IIC
```

```
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.39, 0.21, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}})      '    IID
End Sub

Private Sub Facies05selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies06selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{0, 0.35, 0.65, 0}, {0.5, 0, 0.5, 0}, {0, 0, 1, 0}, {0.5, 0, 0.475, 0.025}})      'IA
    ASSEMBsilicateAFM.Add(1, {{0, 0.35, 0.65, 0}, {0.5, 0, 0.5, 0}, {0.5, 0.25, 0.25, 0}, {0.5, 0, 0.475, 0.025}})     'IB
    ASSEMBsilicateAFM.Add(2, {{0.5, 0, 0.5, 0}, {0.5, 0.25, 0.25, 0}, {0.5, 0, 0.475, 0.025}, {1, 0, 0, 0}})    'IC
    ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}})     '    IIA
    ASSEMBsilicateAFM.Add(4, {{0, 0.35, 0.65, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}})     'IIB
    ASSEMBsilicateAFM.Add(5, {{0, 0.35, 0.65, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {0.4, 0, 0, 0.6}})      '      IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}})     '     IID
End Sub

Private Sub Facies06selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies07selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.02, 0.3133, 0}, {0.6667, 0, 0.31667, 0.01666}, {0, 0, 1, 0}}) ' IA
    ASSEMBsilicateAFM.Add(1, {{0.6667, 0.02, 0.3133, 0}, {0.6667, 0, 0.31667, 0.01666}, {0, 0.1, 0.9, 0}, {0, 0, 1, 0}}) '     IB
```

```
    ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.0667, 0.2666, 0}, {1, 0, 0, 0}, {0.6667, 0, 0.31667, 0.01666}}) '
   IC
    ASSEMBsilicateAFM.Add(3, {{0.4, 0, 0, 0.6}, {0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}}) 'IIA
    ASSEMBsilicateAFM.Add(4, {{0, 1, 0, 0}, {0.4, 0, 0, 0.6}, {0.4, 0.6, 0, 0}, {0.4, 0.12, 0.48, 0}}) '  IIB
    ASSEMBsilicateAFM.Add(5, {{0.4, 0, 0, 0.6}, {0.4, 0.6, 0, 0}, {1, 0, 0, 0}, {0.4, 0.3, 0.3, 0}}) '    IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.108, 0.408, 0.084}, {0.4, 0, 0.45, 0.15}, {0.4, 0, 0, 0.6}, {0, 0, 1, 0}}) '      IID
    ASSEMBsilicateAFM.Add(7, {{0, 1, 0, 0}, {0, 0.1, 0.9, 0}, {0.4, 0.12, 0.48, 0}, {0.4, 0, 0, 0.6}}) '  IIE
    ASSEMBsilicateAFM.Add(8, {{0.4, 0, 0.45, 0.15}, {0.4, 0, 0, 0.6}, {1, 0, 0, 0}, {0.4, 0.3, 0.3, 0}}) 'IIF
End Sub

Private Sub Facies07selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies08selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0, 0.3166, 0.0167}, {0, 0, 1, 0}, {0, 0.25, 0.75, 0}}) '   IA
    ASSEMBsilicateAFM.Add(1, {{0.6667, 0, 0.3166, 0.0167}, {0, 0.25, 0.75, 0}, {2 / 3, 0, 1 / 3, 0}, {2 / 3, 0.1 / 3, 0.3, 0}}) '
   IB
    ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {2 / 3, 0.1 / 3, 0.3, 0}, {0.6667, 0, 0.3166, 0.0167}, {1, 0, 0, 0}}) '    IC
    ASSEMBsilicateAFM.Add(3, {{0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {0.4, 0, 0, 0.6}}) 'IIA
    ASSEMBsilicateAFM.Add(4, {{0, 0.25, 0.75, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '    IIB
    ASSEMBsilicateAFM.Add(5, {{0, 0.25, 0.75, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.3, 0.3, 0}, {0.4, 0, 0, 0.6}}) '     IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.3, 0.3, 0}, {0.4, 0, 0, 0.6}, {1, 0, 0, 0}}) '    IID


    'ASSEMBsilicateAFM.Add(7, {{0, 0, 1, 0}, {0, 0.25, 0.75, 0}, {0.4, 0, 0, 0.6}, {0, 1, 0, 0}}) 'IIE  PBL HERE
End Sub

Private Sub Facies08selectName()
```

```
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")

    'assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies11selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{0, 0.6, 0.4, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {0.6667, 0, 0.3166, 0.0167}}) '     IA
    ASSEMBsilicateAFM.Add(1, {{0, 0.6, 0.4, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1333, 0.2, 0}, {0.6667, 0, 0.3166, 0.0167}}) '
   IB
    ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1333, 0.2, 0}, {0.6667, 0, 0.3166, 0.0167}, {1, 0, 0, 0}}) '    IC
    ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
    ASSEMBsilicateAFM.Add(4, {{0, 0.6, 0.4, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '      IIB
    ASSEMBsilicateAFM.Add(5, {{0, 0.6, 0.4, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.528, 0.072, 0}, {0.4, 0, 0, 0.6}}) '   IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.528, 0.072, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) 'IID
End Sub

Private Sub Facies11selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies15selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{0, 0.65, 0.35, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {2 / 3, 0, 0.3, 0.1 / 3}}) '      IA
    ASSEMBsilicateAFM.Add(1, {{0, 0.65, 0.35, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1433, 0.19, 0}, {2 / 3, 0, 0.3, 0.1 / 3}}) '
   IB
    ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1433, 0.19, 0}, {2 / 3, 0, 0.3, 0.1 / 3}, {1, 0, 0, 0}}) '      IC
```

```
    ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
    ASSEMBsilicateAFM.Add(4, {{0, 0.65, 0.35, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '    IIB
    ASSEMBsilicateAFM.Add(5, {{0, 0.65, 0.35, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.492, 0.108, 0}, {0.4, 0, 0, 0.6}}) ' IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.492, 0.108, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) 'IID
End Sub

Private Sub Facies15selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub

Private Sub Facies16selectCoordinates()
    ASSEMBsilicateAFM.Add(0, {{0, 0.48, 0.52, 0}, {2 / 3, 0, 1 / 3, 0}, {0, 0, 1, 0}, {0.6667, 0, 0.32, 0.0133}}) '     IA
    ASSEMBsilicateAFM.Add(1, {{0, 0.48, 0.52, 0}, {2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1, 0.2333, 0}, {0.6667, 0, 0.32, 0.0133}}) '
   IB
    ASSEMBsilicateAFM.Add(2, {{2 / 3, 0, 1 / 3, 0}, {0.6667, 0.1, 0.2333, 0}, {0.6667, 0, 0.32, 0.0133}, {1, 0, 0, 0}}) '      IC
    ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
    ASSEMBsilicateAFM.Add(4, {{0, 0.48, 0.52, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '    IIB
    ASSEMBsilicateAFM.Add(5, {{0, 0.48, 0.52, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.48, 0.12, 0}, {0.4, 0, 0, 0.6}}) '   IIC
    ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.48, 0.12, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) '  IID
End Sub

Private Sub Facies16selectName()
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre01")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
    assemblageMxAFM.Add("tetraedre02")
End Sub
```

```
    Private Sub Facies17selectCoordinates()
        ASSEMBsilicateAFM.Add(0, {{0, 0.67, 0.33, 0}, {0.5, 0, 0.5, 0}, {0, 0, 1, 0}, {0.5, 0, 0.475, 0.025}}) '      IA
        ASSEMBsilicateAFM.Add(1, {{0, 0.67, 0.33, 0}, {0.5, 0, 0.5, 0}, {0.5, 0.2, 0.3, 0}, {0.5, 0, 0.475, 0.025}}) '      IB
        ASSEMBsilicateAFM.Add(2, {{0.5, 0, 0.5, 0}, {0.5, 0.2, 0.3, 0}, {0.5, 0, 0.475, 0.025}, {1, 0, 0, 0}}) '      IC
        ASSEMBsilicateAFM.Add(3, {{0, 0, 0, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0.4, 0, 0, 0.6}}) 'IIA
        ASSEMBsilicateAFM.Add(4, {{0, 0.67, 0.33, 0}, {0, 1, 0, 0}, {0.4, 0.6, 0, 0}, {0.4, 0, 0, 0.6}}) '     IIB
        ASSEMBsilicateAFM.Add(5, {{0, 0.67, 0.33, 0}, {0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {0.4, 0, 0, 0.6}}) '   IIC
        ASSEMBsilicateAFM.Add(6, {{0.4, 0.6, 0, 0}, {0.4, 0.51, 0.09, 0}, {1, 0, 0, 0}, {0.4, 0, 0, 0.6}}) '  IID
    End Sub

    Private Sub Facies17selectName()
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre01")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
        assemblageMxAFM.Add("tetraedre02")
    End Sub
#End Region


End Class
```

4. ValueMx.vb

```
Public Class ValueMx

    Friend NBmole As Double       'nb Moles without Si, H, O or C for the Mx
    Friend typeValue As enumTypeValue
    Friend weightPct As Double
    Friend weightPct100 As Double
    Friend remark As String
    Friend density As Double

    Friend Enum enumTypeValue
        element = 1
        derivated = 2
        silicate = 3
        oxyde = 4
        carbonate = 5
        sulfure = 6
        otherMX = 7
        text = 8
    End Enum


    Friend Sub New(ByVal NBmole As Double, ByVal type As enumTypeValue, Optional DensityData As Double = 0)
        If Double.IsNaN(NBmole) Then
            Me.NBmole = 0
        Else
            Me.NBmole = NBmole
        End If

        typeValue = type
        density = DensityData
    End Sub
End Class
```

ADDITIONAL MATERIAL

Article: CONSONORM_HG: a new method of norm calculation for mid- to high-grade metamorphic rocks, by Mathieu, L., Trépanier, S. & Daigneault, R. (Journal of Metamorphic Geology).

This document contains a copy of the ternary diagrams and tetrahedra of the CONSONORM_HG norm. The $CaCO_3$–$FeCO_3$–$MgCO_3$ ternary diagrams (McSwiggen, 1993; Franzolin *et al.*, 2011) and the TiO_2–FeO–Fe_2O_3 ternary diagrams (Deer *et al.*, 1962; Haggerty, 1991; Lindsey, 1991) are from the literature. The other diagrams were adapted from ternary diagrams created with the Perple_X software, using several solid-solution models (see Table S3) and the hp02ver database (see article for details).

LIST OF DOCUMENTS

Figure S1. Ternary diagrams used for carbonate calculation

Table S1. Sulphides of CONSONORM_HG

Figure S2. Ternary diagrams used for Fe–Ti-oxide calculation

Figure S3. $CaCO_3$–$FeCO_3$–$MgCO_3$–SiO_2 tetrahedra used to estimate if quartz and carbonates do, or do not, co-exist in an assemblage

Figure S4. AFMM tetrahedra used to decide which tetrahedron, among tetrahedron I, II or III, is to be used to calculate the silicate assemblage

Annexe A. ACKNFM tetrahedra used to calculate the silicate assemblage for the 17 P–T conditions modeled by CONSONORM_HG

Figure S5. Fe–Mg–Mn ternary diagrams used to determine the type of minerals represented by the FM apex of each ACKNFM tetrahedron

Table S2. Simplified formula of normative minerals

Annexe B. Reactions and adjustments used by CONSONORM_HG

Table S3. Solid solution models used to calculate the ternary diagrams using the Perple_X software. These ternary diagrams were then integrated in the design of the AFMM, ACKNFM and Fe–Mg–Mn tetrahedral and ternary diagrams

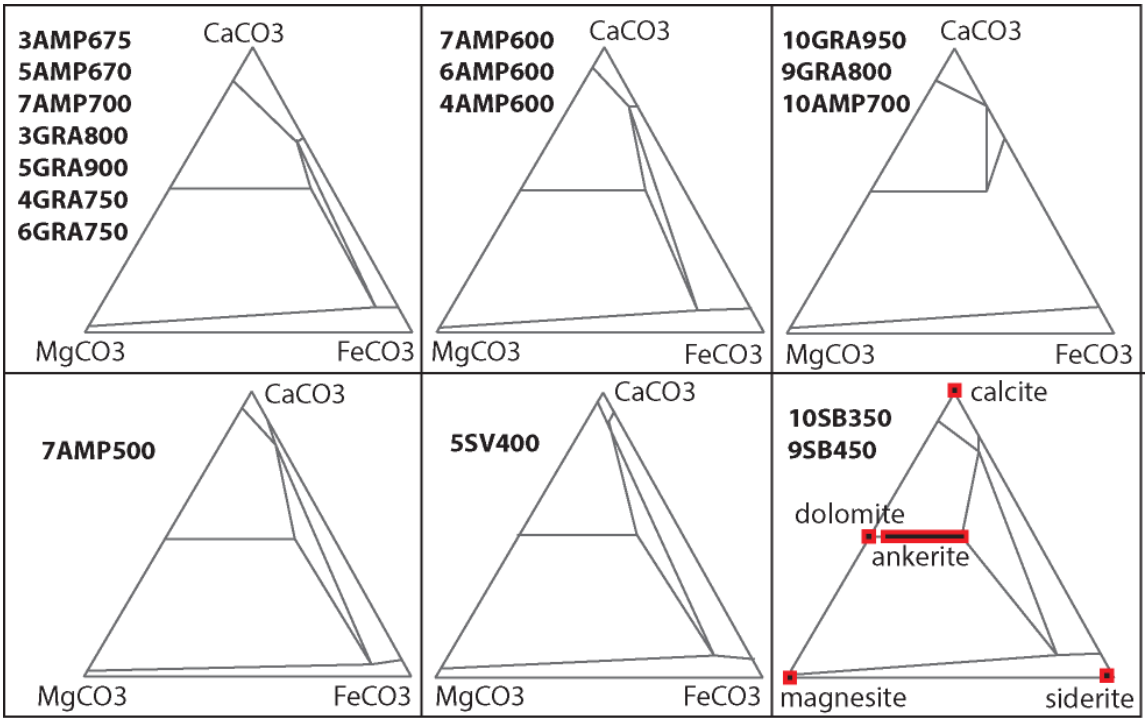


Fig. S1. Ternary diagrams used for carbonate calculation

Table S1. Sulphides of CONSONORM_HG

(1) Pyrrhotite: for models 10GRA950, 6GRA750, 5GRA900, 4GRA750 and 3GRA800. (2) Pyrite: for all the other models.

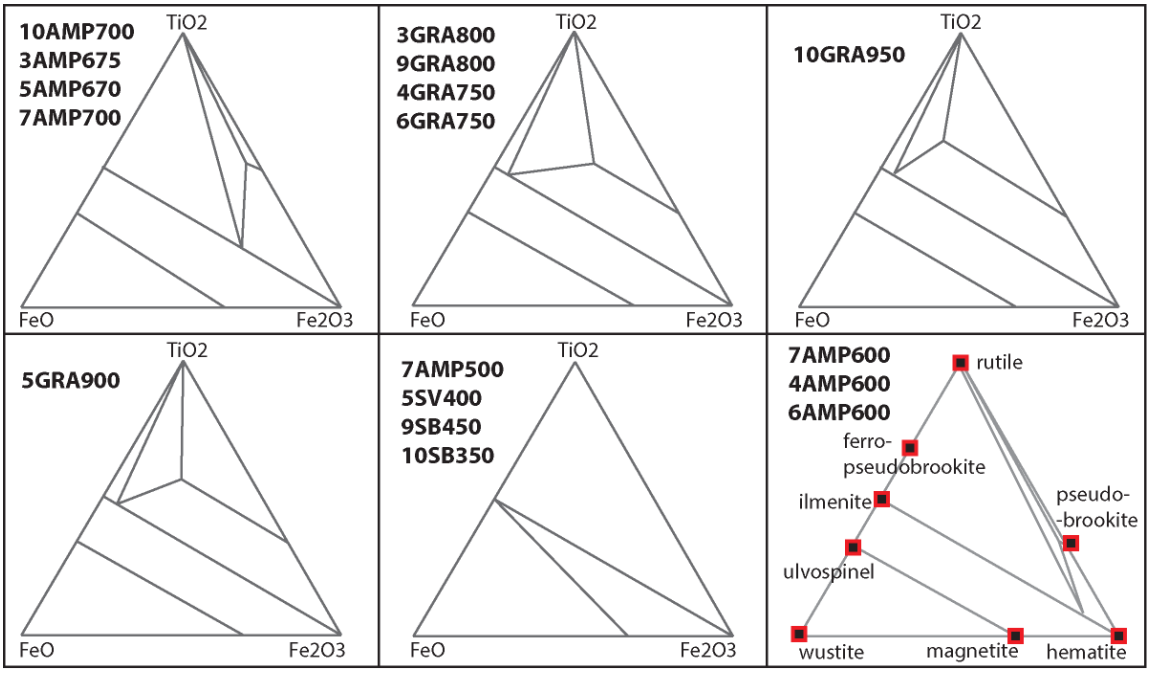


Fig. S2. Ternary diagrams used for Fe–Ti oxide calculation

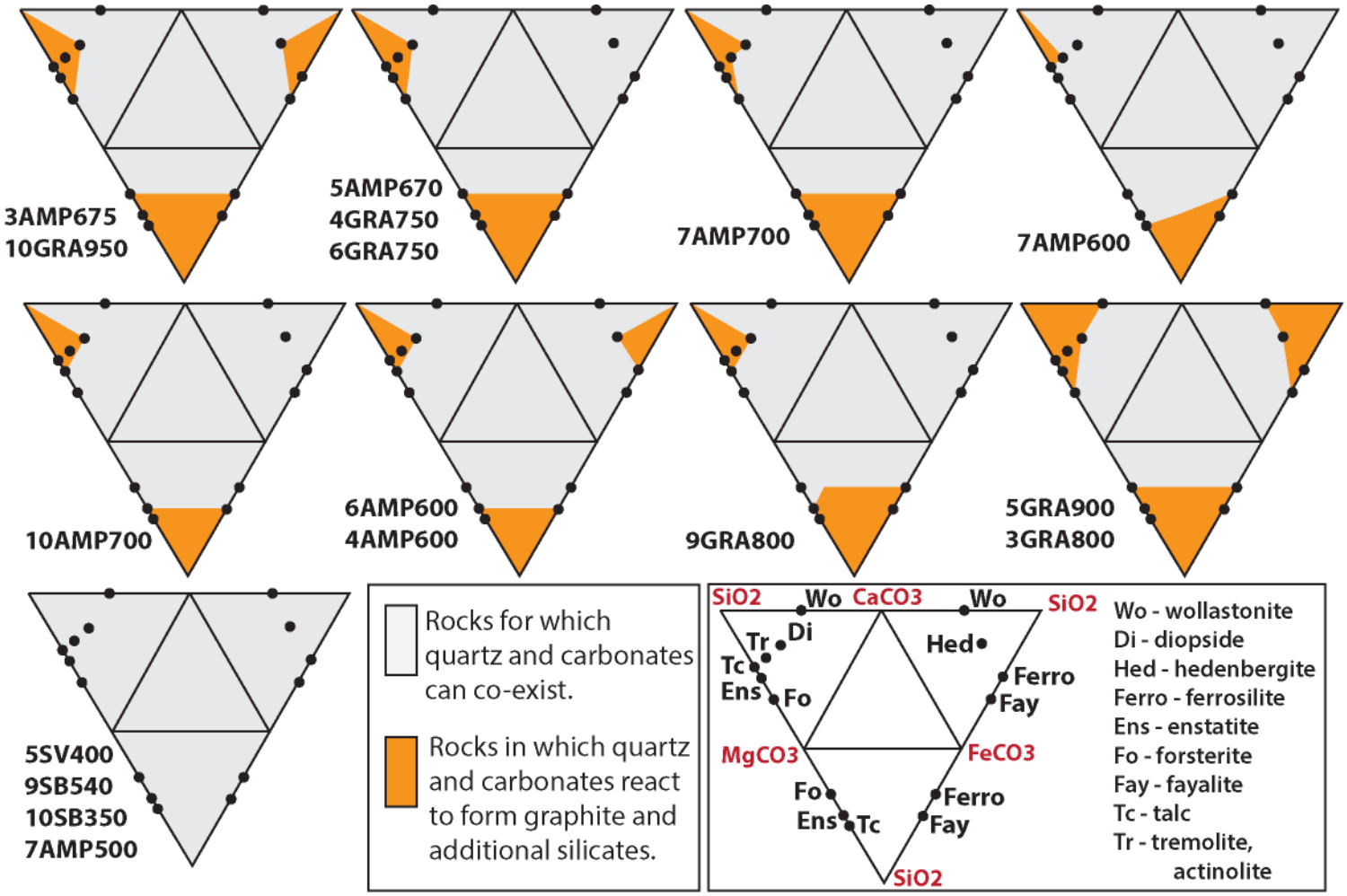


Fig. S3. $CaCO_3$–$FeCO_3$–$MgCO_3$–SiO_2 tetrahedra used to estimate if quartz and carbonates do, or do not, co-exist in an assemblage

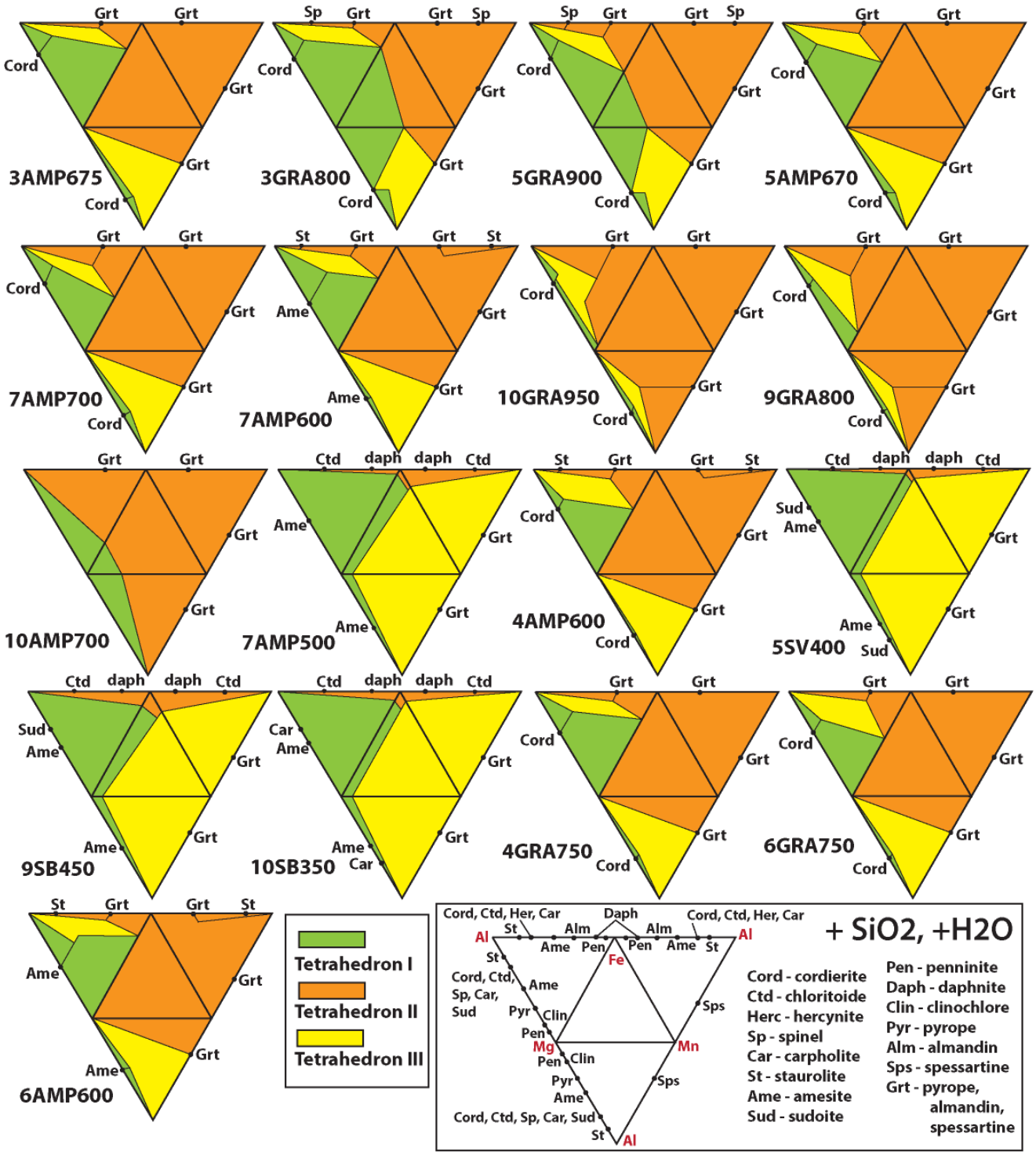


Fig. S4. AFMM tetrahedra used to decide which tetrahedron, among tetrahedron I, II or III, is to be used to calculate the silicate assemblage

A – ACKNFM tetrahedra used to calculate the silicate assemblage for the 17 P–T conditions modelled by CONSONORM_HG

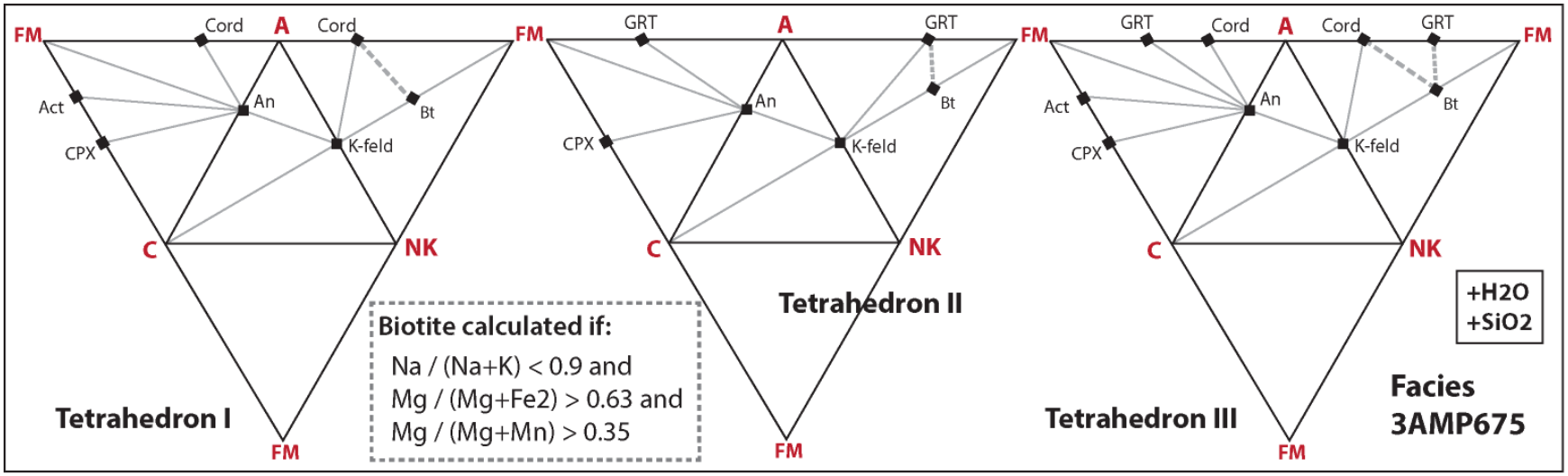
Burst views of the tetrahedra of the 17 models of CONSONORM_HG are display hereafter. The following abbreviations are used in these figures:

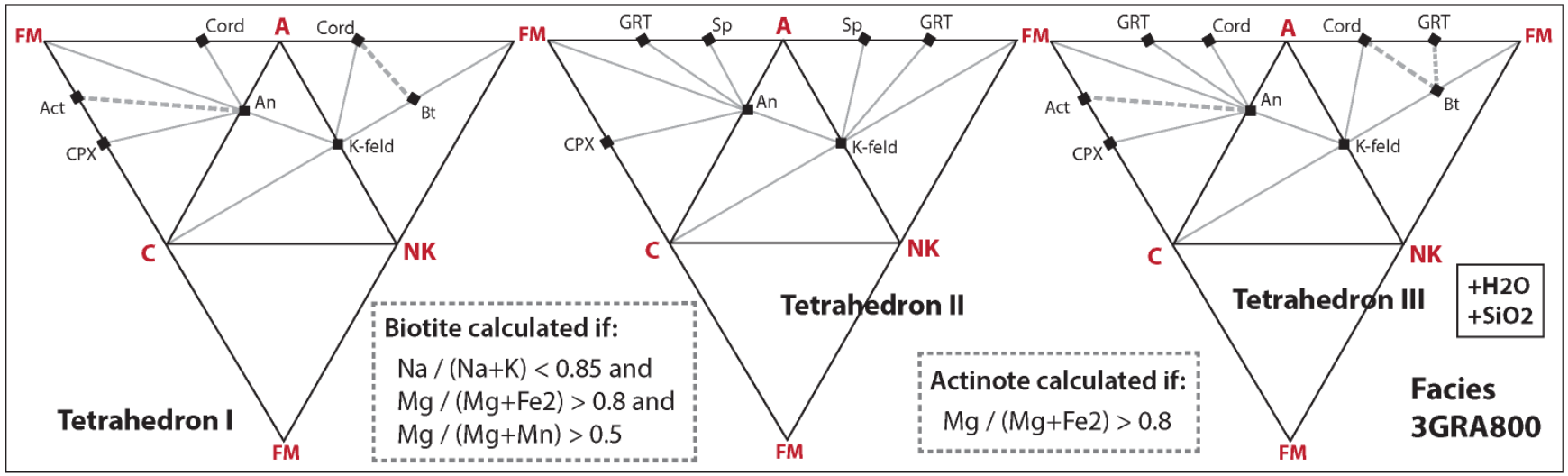
1) The tetrahedra apexes are designed as the A, C, FM and NK apexes:

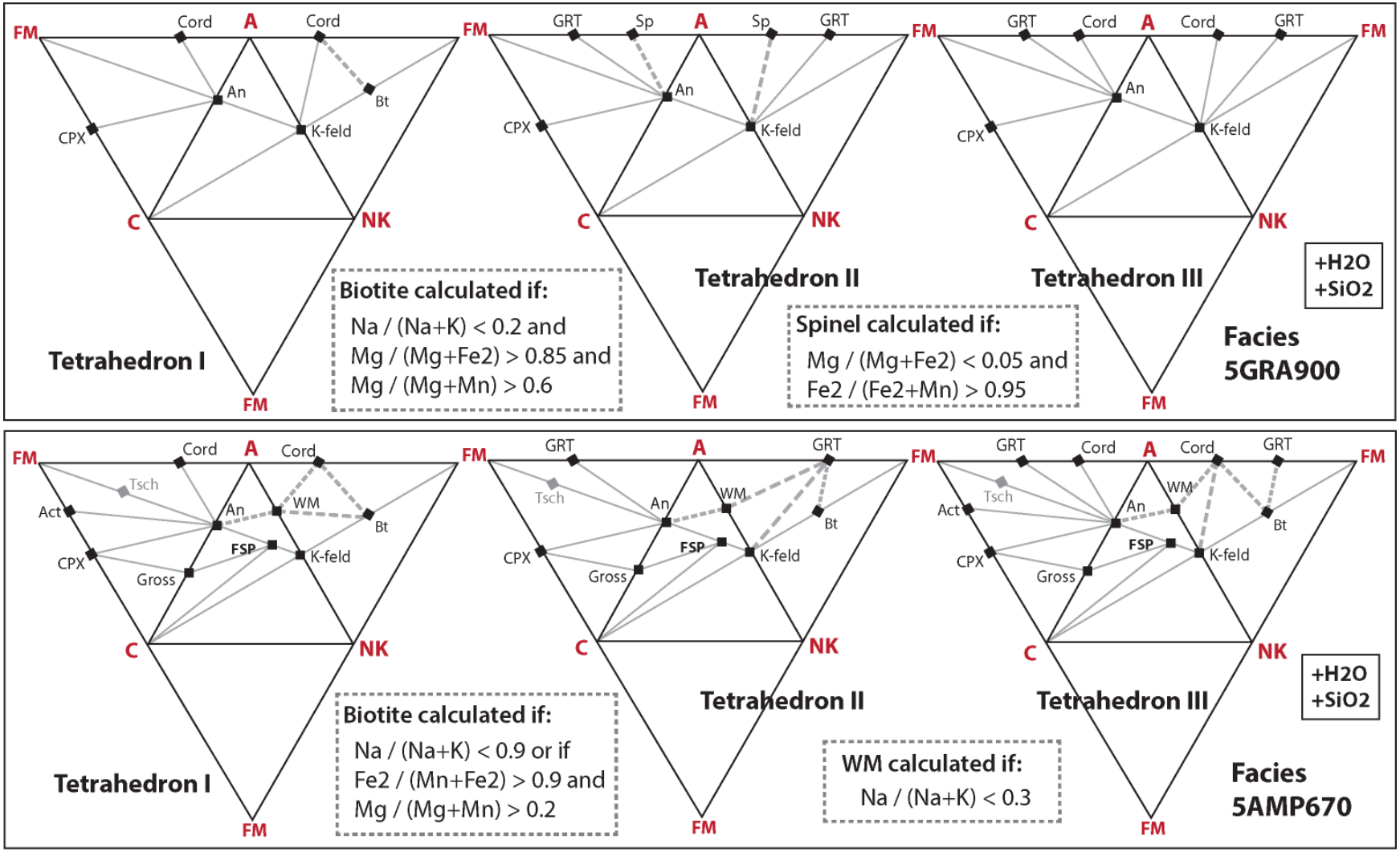
Apexes	Details
A	Al apex; represented by pyrophyllite (5SV400, 10SB350), kyanite (7AMP600, 10AMP700, 7AMP500, 9SB450, 6AMP600) or sillimanite (all the other facies)
C	Ca apex; represented by wollastonite
FM	Fe+Mg+Mn apex; represented by talc, olivine, anthophyllite, orthopyroxene, pyroxmangite and/or rhodonite depending on the facies and on the composition of the sample in Fe, Mg and Mn (see figure G of this annexe).
NK	Na–K apex; not represented by any mineral, because there exists no simple and common silicates of Na–K. This is a "forbidden apex", meaning that when a small tetrahedron containing this apex is selected during the CONSONORM_HG calculation, then 0.1 wt% of Na2O and K2O is removed from the sample, and the selection of a small tetrahedron is resumed, and this process is repeated until a small tetrahedron that does not contain the NK apex is selected. The removed amount of Na2O+K2O is usually small and is considered as being in excess by the norm.

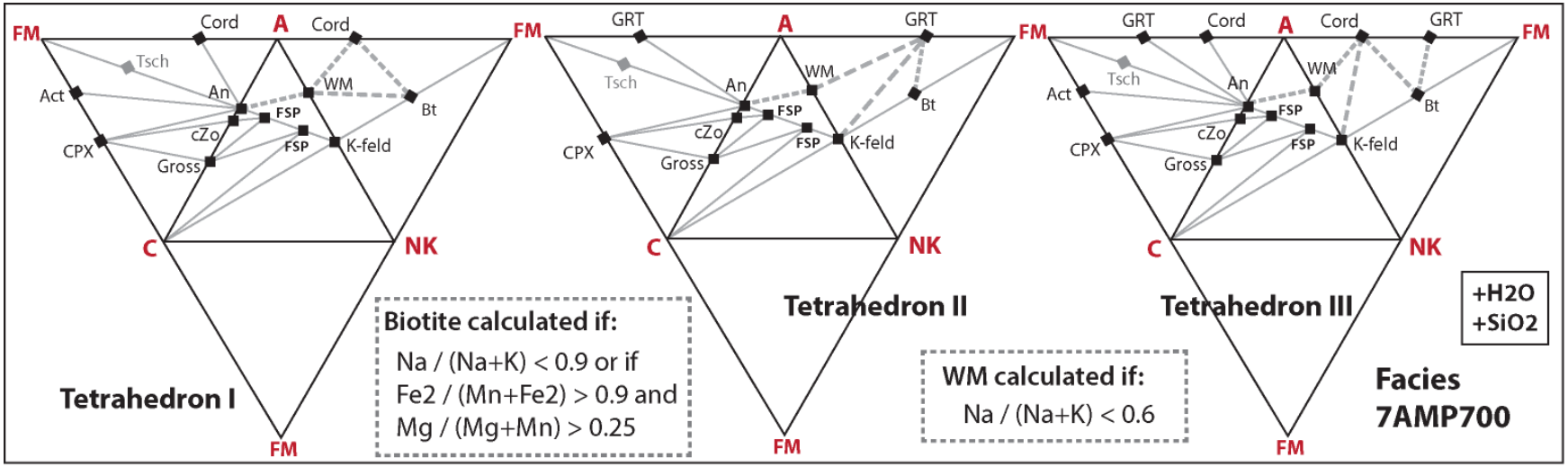
2) Mineral abbreviations used hereafter are the following:

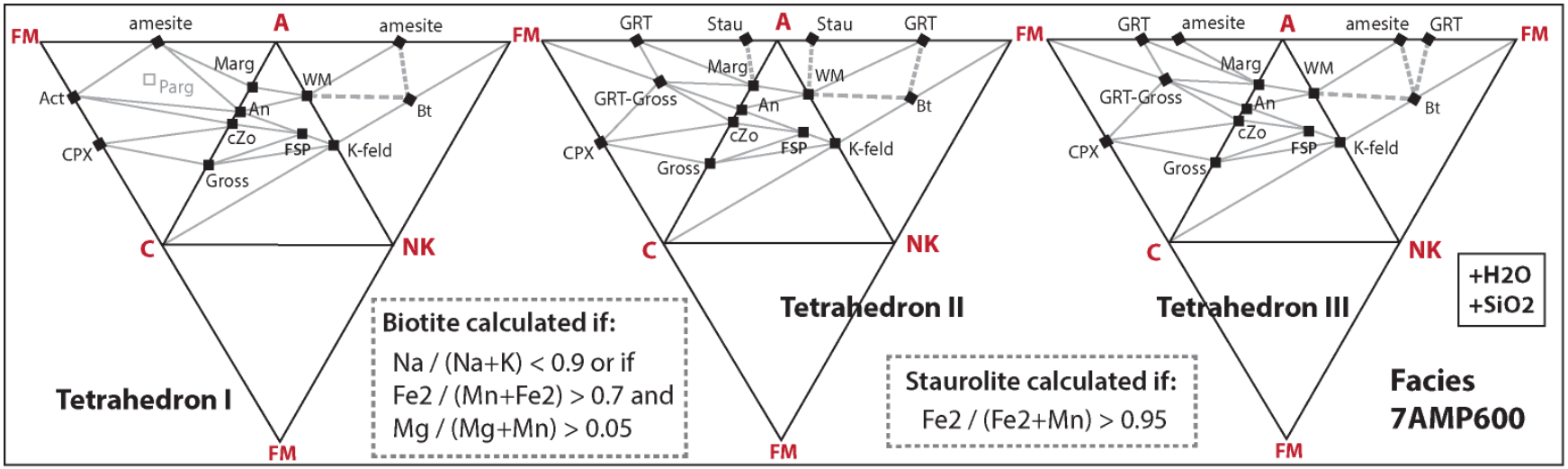
Abbreviation	Mineral's names	Abbreviation	Mineral's names
Act	actinolite, tremolite (amphibole)	**Gross**	grossular (garnet)
amesite	amesite (chlorite)	**Lw**	lawsonite
Bt	annite, phlogopite (biotite)	**Marg**	margarite (white mica)
An	anorthite (plagioclase)	**WM**	muscovite, paragonite (white mica)
Car	carpholite	**K-feld**	orthose, albite (alkali feldspar)
Ctd	chloritoide	**Parg**	pargasite (amphibole)
cZo	clinozoisite (epidote)	**GRT**	pyrope, almandine (garnet)
Cord	cordierite	**Sp**	spinel, hercynite (spinels)
daphnite	daphnite (chlorite)	**Stau**	staurolite
CPX	diopside, hedenbergite (clinopyroxene)	**Tsch**	tschermakite (amphibole)
FSP	mixture of anorthite, orthose and albite used to divide small tetrahedra (feldspar)		
GRT–Gross	mixture of pyrope, almandine and grossular used to divide small tetrahedra (garnet)		

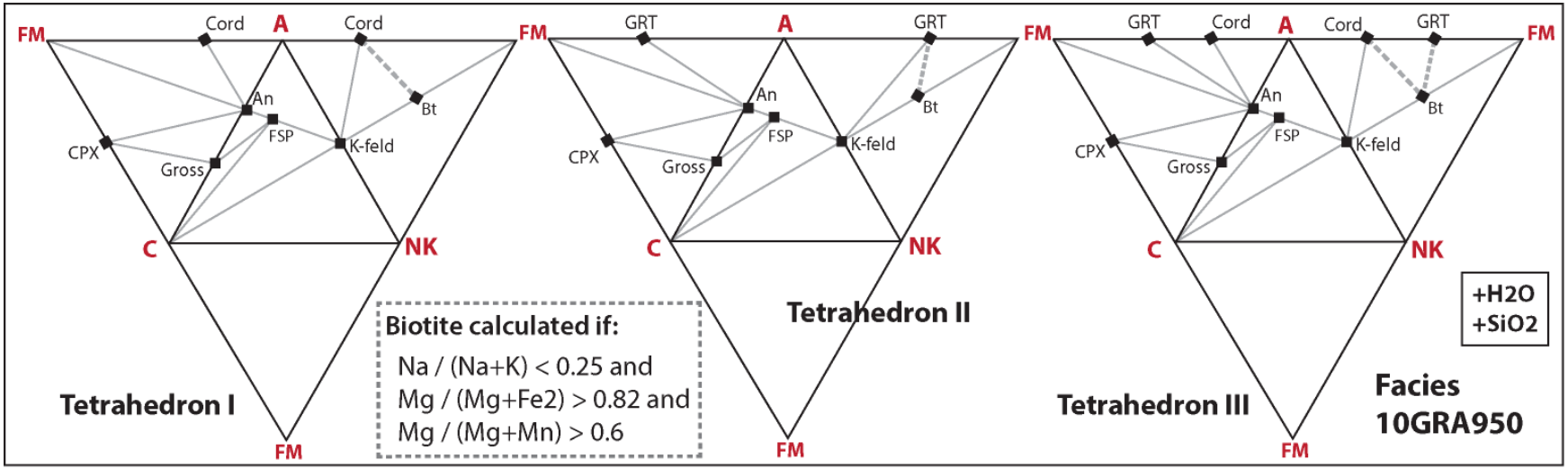

FM
Cord
A
Cord
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 and
Mg / (Mg+Fe2) > 0.63 and
Mg / (Mg+Mn) > 0.35
FM
GRT
A
GRT
FM
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron II
FM
GRT
Cord
A
Cord
GRT
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron III
+H2O
+SiO2
Facies
3AMP675

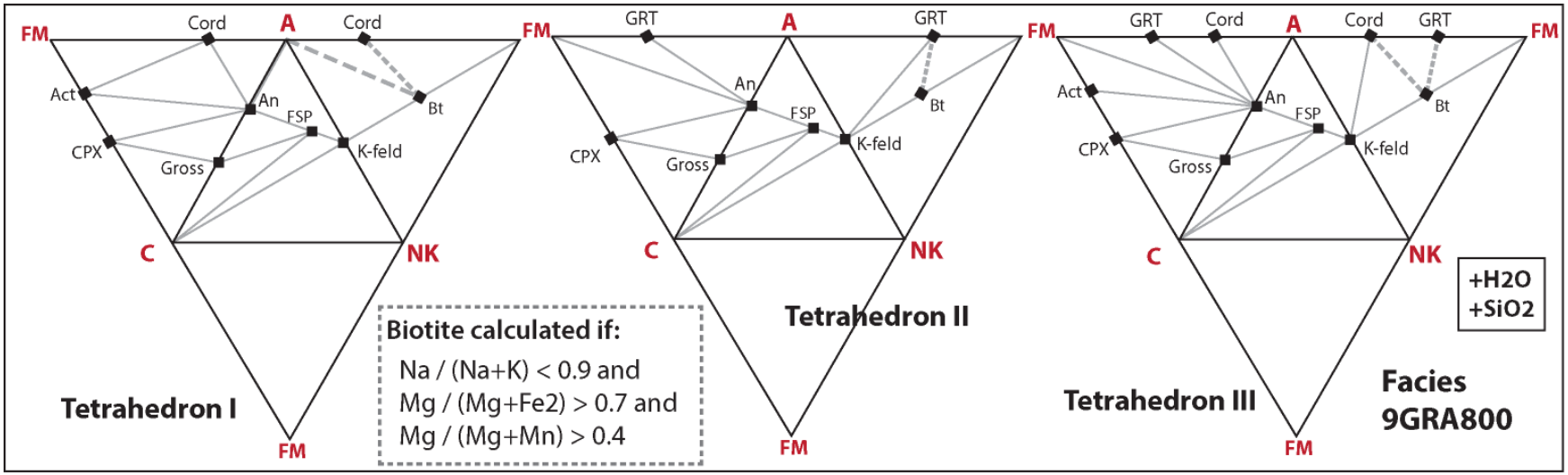

FM
Cord
A
Cord
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.85 and
Mg / (Mg+Fe2) > 0.8 and
Mg / (Mg+Mn) > 0.5
FM
GRT
Sp
A
Sp
GRT
FM
An
CPX
K-feld
C
NK
FM
Tetrahedron II
Actinote calculated if:
Mg / (Mg+Fe2) > 0.8
FM
GRT
Cord
A
Cord
GRT
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron III
+H2O
+SiO2
Facies
3GRA800

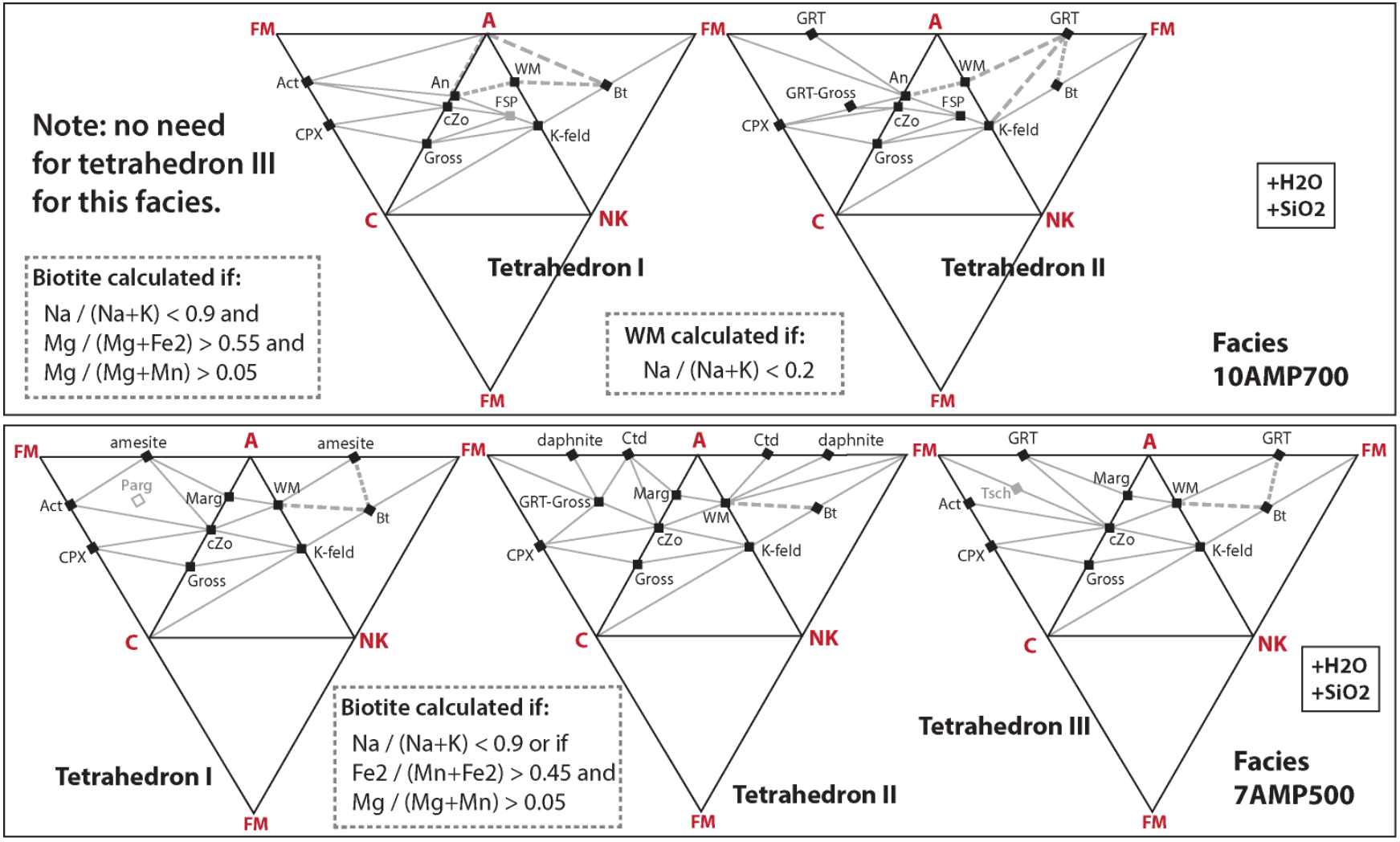

FM
Cord
A
Cord
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.2 and
Mg / (Mg+Fe2) > 0.85 and
Mg / (Mg+Mn) > 0.6
GRT
Sp
A
Sp
GRT
Tetrahedron II
Spinel calculated if:
Mg / (Mg+Fe2) < 0.05 and
Fe2 / (Fe2+Mn) > 0.95
Tetrahedron III
+H2O
+SiO2
Facies
5GRA900
Tsch
Act
WM
FSP
Gross
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.9 and
Mg / (Mg+Mn) > 0.2
WM calculated if:
Na / (Na+K) < 0.3
Facies
5AMP670

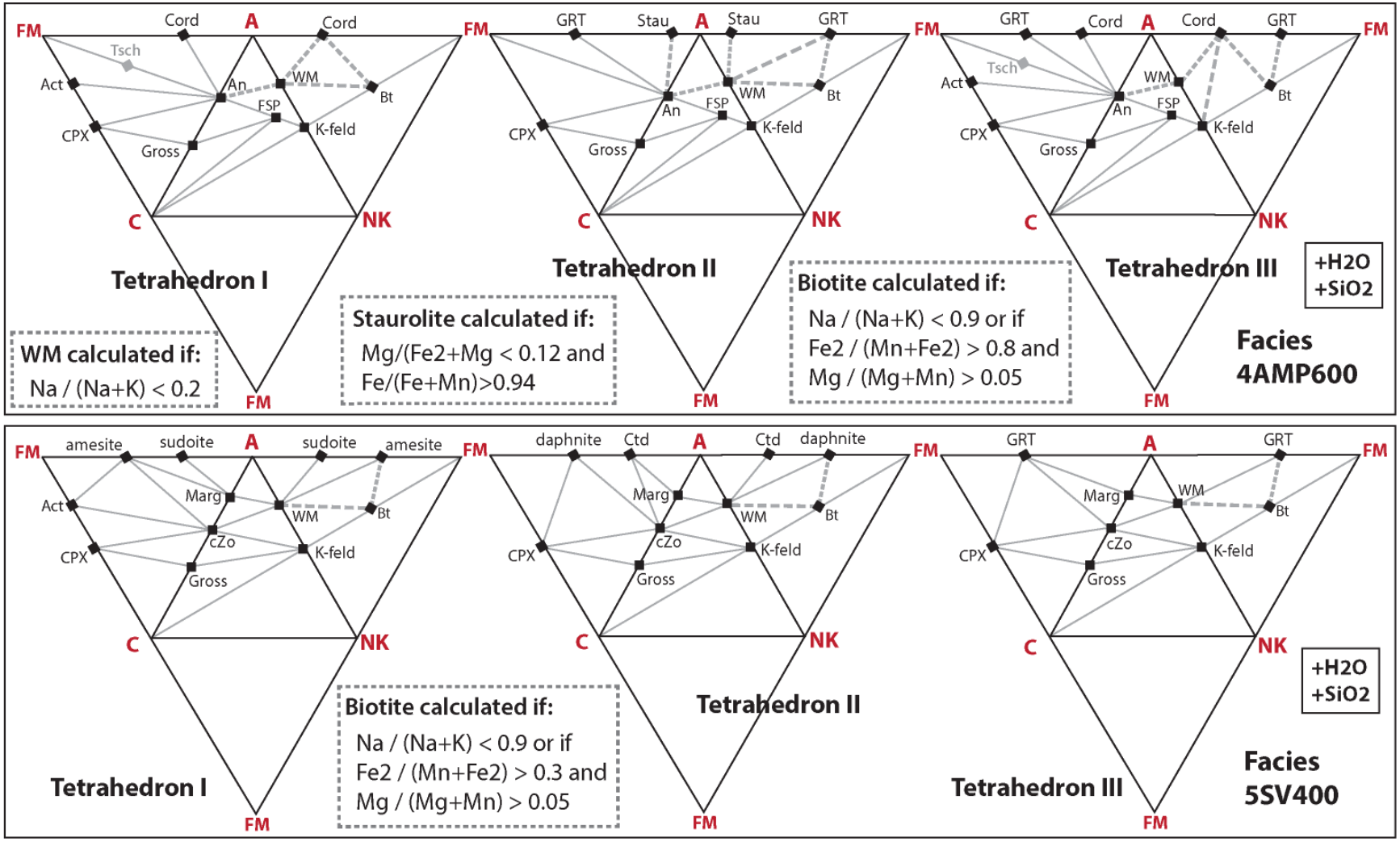

FM
Cord
A
Cord
Tsch
Act
An
WM
Bt
FSP
cZo
CPX
FSP
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.9 and
Mg / (Mg+Mn) > 0.25
GRT
Tetrahedron II
WM calculated if:
Na / (Na+K) < 0.6
Tetrahedron III
+H2O
+SiO2
Facies
7AMP700

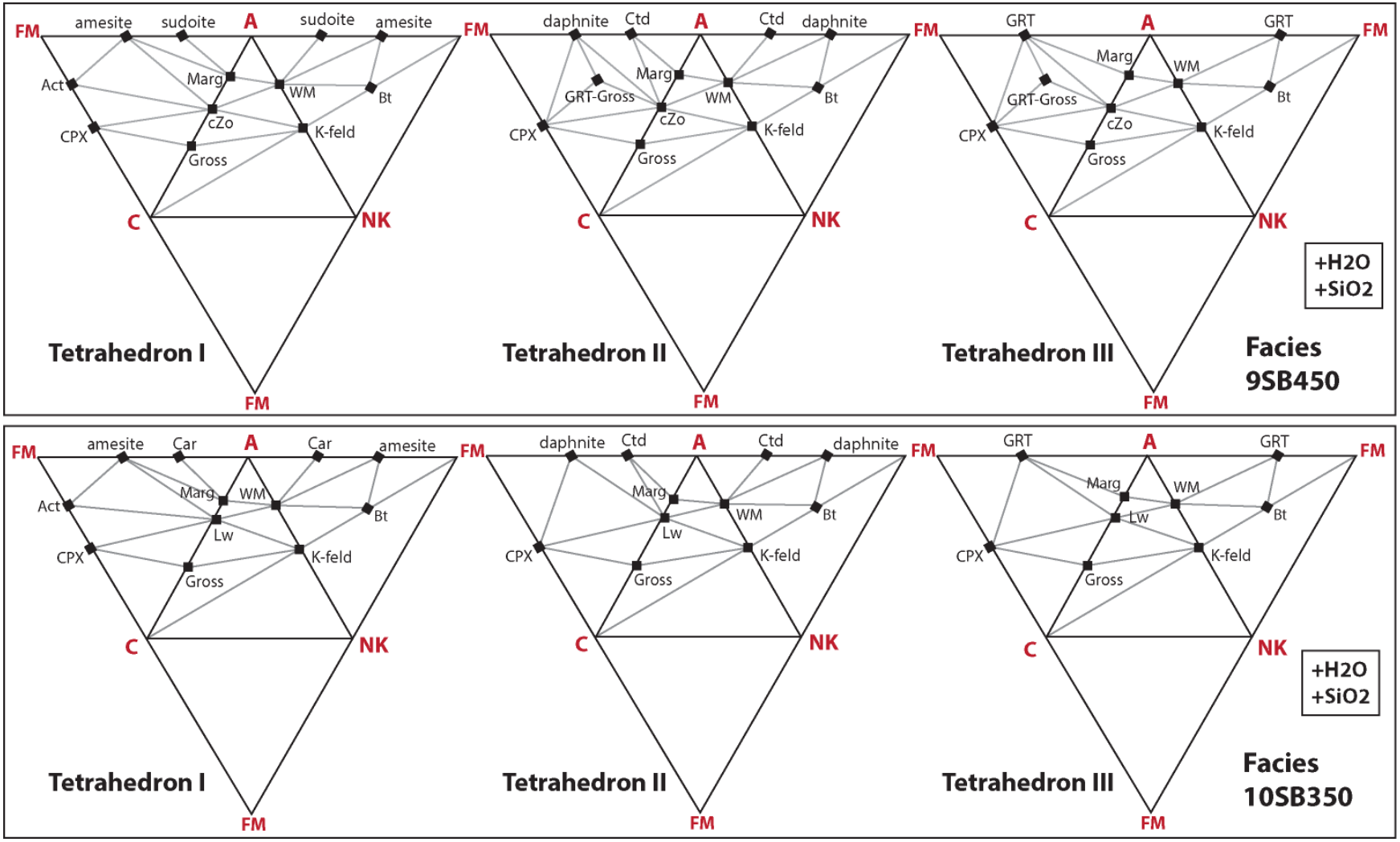

FM
amesite
A
amesite
Marg
Parg
WM
Act
An
Bt
cZo
CPX
FSP
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.7 and
Mg / (Mg+Mn) > 0.05
GRT
Stau
Tetrahedron II
GRT-Gross
Staurolite calculated if:
Fe2 / (Fe2+Mn) > 0.95
Tetrahedron III
+H2O
+SiO2
Facies
7AMP600

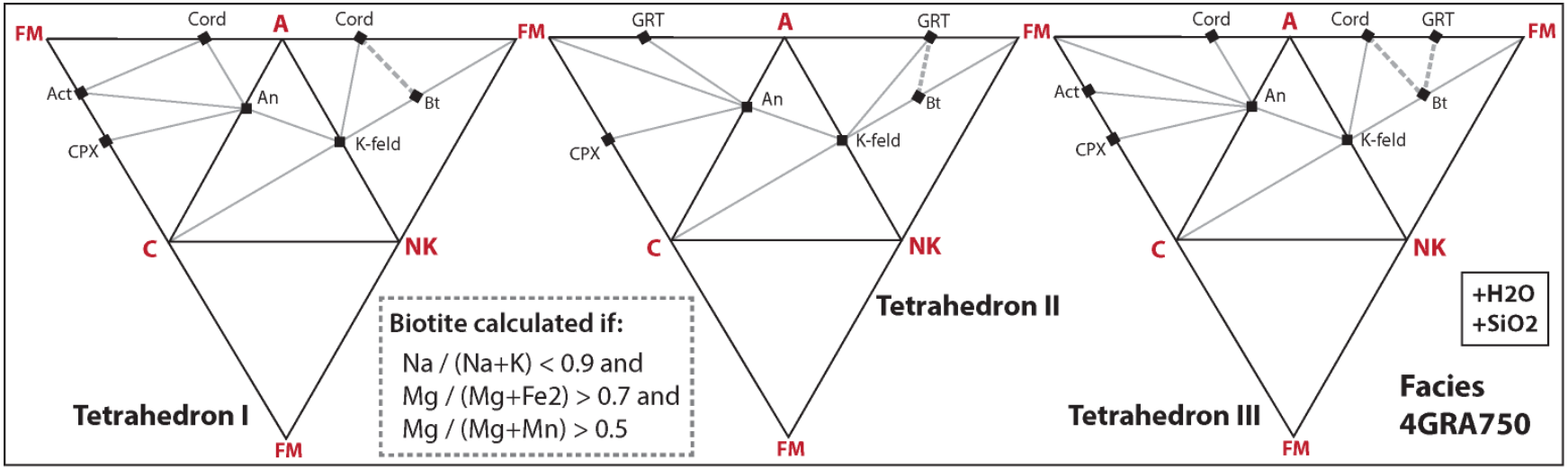

FM
Cord
A
Cord
An
Bt
FSP
CPX
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.25 and
Mg / (Mg+Fe2) > 0.82 and
Mg / (Mg+Mn) > 0.6
GRT
Tetrahedron II
Tetrahedron III
+H2O
+SiO2
Facies
10GRA950

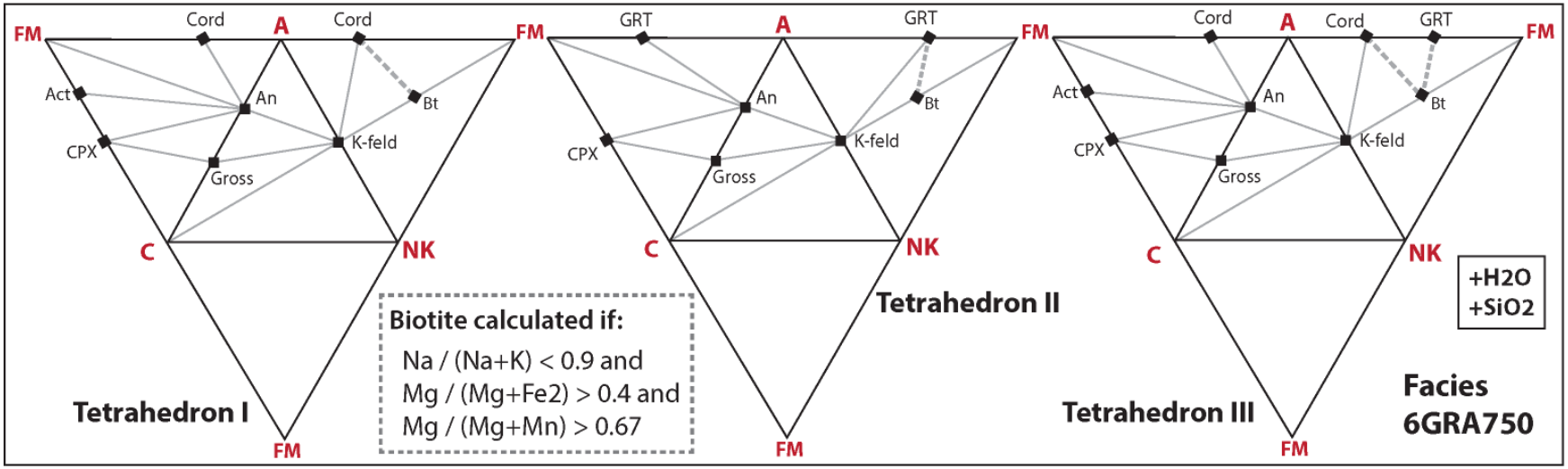

FM
Cord
A
Cord
Act
An
Bt
FSP
CPX
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 and
Mg / (Mg+Fe2) > 0.7 and
Mg / (Mg+Mn) > 0.4
GRT
Tetrahedron II
Tetrahedron III
+H2O
+SiO2
Facies
9GRA800

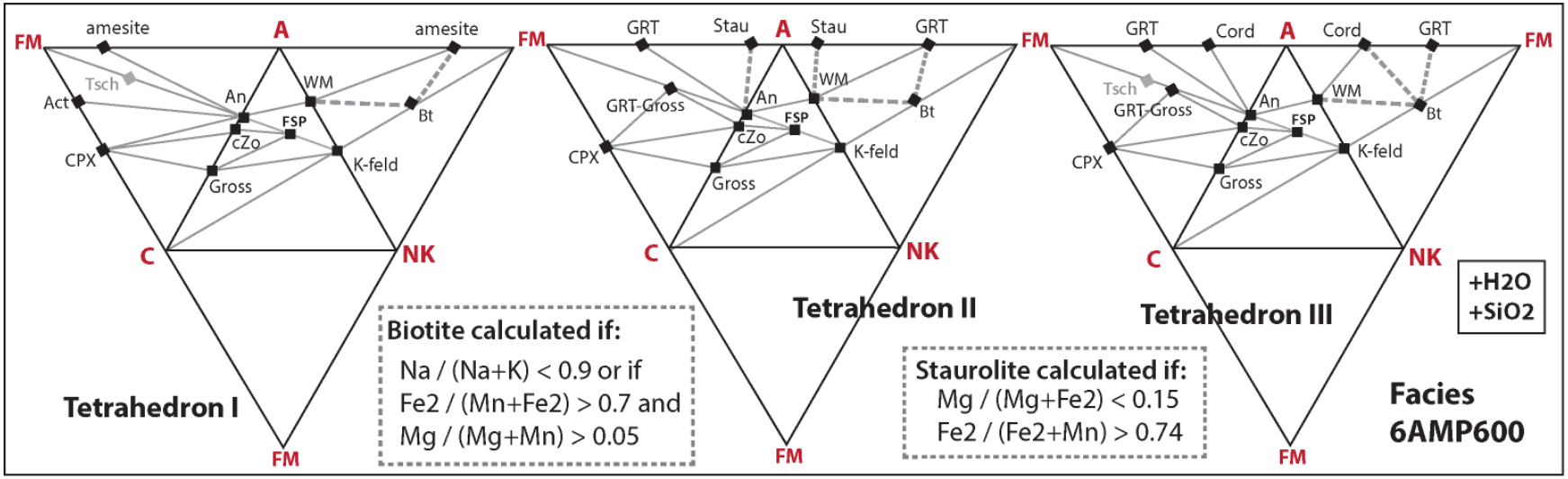

Note: no need for tetrahedron III for this facies.
FM
A
C
NK
Act
An
WM
Bt
CPX
FSP
cZo
K-feld
Gross
Tetrahedron I
GRT
GRT-Gross
Tetrahedron II
Biotite calculated if:
Na / (Na+K) < 0.9 and
Mg / (Mg+Fe2) > 0.55 and
Mg / (Mg+Mn) > 0.05
WM calculated if:
Na / (Na+K) < 0.2
+H2O
+SiO2
Facies
10AMP700
amesite
Parg
Marg
daphnite
Ctd
Tsch
Tetrahedron III
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.45 and
Mg / (Mg+Mn) > 0.05
Facies
7AMP500


FM
Cord
A
Cord
Tsch
Act
An
WM
Bt
FSP
CPX
K-feld
Gross
C
NK
Tetrahedron I
WM calculated if:
Na / (Na+K) < 0.2
GRT
Stau
A
Stau
GRT
Tetrahedron II
Staurolite calculated if:
Mg/(Fe2+Mg < 0.12 and
Fe/(Fe+Mn)>0.94
GRT
Cord
A
Cord
GRT
Tetrahedron III
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.8 and
Mg / (Mg+Mn) > 0.05
+H2O
+SiO2
Facies
4AMP600
amesite
sudoite
A
sudoite
amesite
Marg
cZo
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.3 and
Mg / (Mg+Mn) > 0.05
daphnite
Ctd
A
Ctd
daphnite
Tetrahedron II
GRT
A
GRT
Tetrahedron III
+H2O
+SiO2
Facies
5SV400


FM
amesite
sudoite
A
sudoite
amesite
Act
Marg
WM
Bt
CPX
cZo
K-feld
Gross
C
NK
FM
Tetrahedron I
daphnite
Ctd
A
Ctd
daphnite
GRT-Gross
Tetrahedron II
GRT
Tetrahedron III
+H2O
+SiO2
Facies
9SB450
Car
Lw
Facies
10SB350


FM
Cord
A
Cord
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 and
Mg / (Mg+Fe2) > 0.7 and
Mg / (Mg+Mn) > 0.5
FM
GRT
A
GRT
FM
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron II
FM
Cord
A
Cord
GRT
FM
Act
An
Bt
CPX
K-feld
C
NK
FM
Tetrahedron III
+H2O
+SiO2
Facies
4GRA750


FM
Cord
A
Cord
FM
Act
An
Bt
CPX
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 and
Mg / (Mg+Fe2) > 0.4 and
Mg / (Mg+Mn) > 0.67
FM
GRT
A
GRT
FM
An
Bt
CPX
K-feld
Gross
C
NK
FM
Tetrahedron II
FM
Cord
A
Cord
GRT
FM
Act
An
Bt
CPX
K-feld
Gross
C
NK
FM
Tetrahedron III
+H2O
+SiO2
Facies
6GRA750


FM
amesite
A
amesite
Tsch
Act
An
WM
Bt
FSP
cZo
CPX
K-feld
Gross
C
NK
FM
Tetrahedron I
Biotite calculated if:
Na / (Na+K) < 0.9 or if
Fe2 / (Mn+Fe2) > 0.7 and
Mg / (Mg+Mn) > 0.05
FM
GRT
Stau
A
Stau
GRT
GRT-Gross
An
WM
Bt
FSP
cZo
CPX
K-feld
Gross
C
NK
FM
Tetrahedron II
Staurolite calculated if:
Mg / (Mg+Fe2) < 0.15
Fe2 / (Fe2+Mn) > 0.74
FM
GRT
Cord
A
Cord
GRT
Tsch
GRT-Gross
An
WM
Bt
FSP
cZo
CPX
K-feld
Gross
C
NK
FM
Tetrahedron III
+H2O
+SiO2
Facies
6AMP600

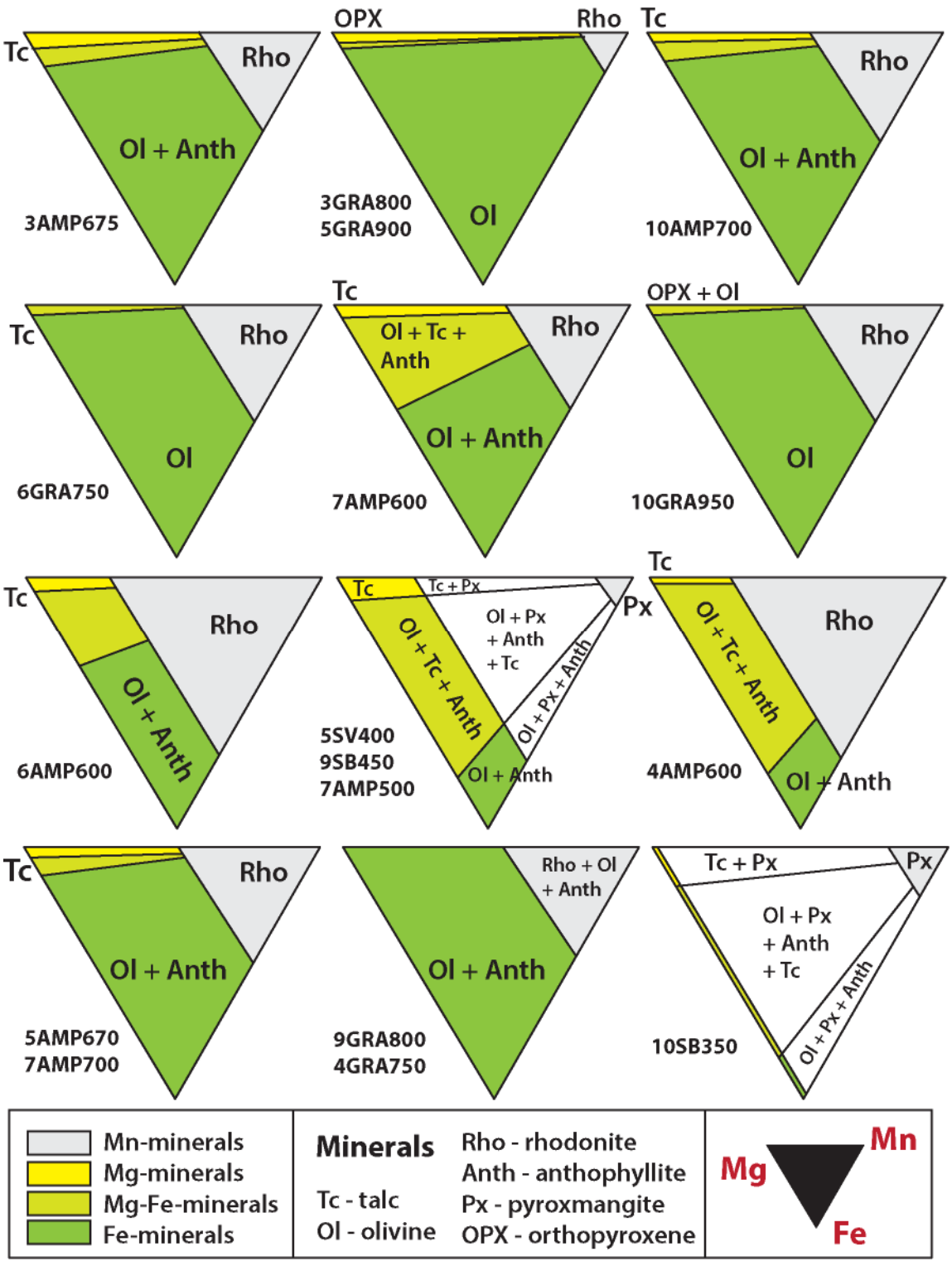


Fig. S5. Fe–Mg–Mn ternary diagrams used to determine the type of minerals represented by the FM apex of each ACKNFM tetrahedron

Table S2. Simplified formula of normative minerals

Mineral	Simplified formula										
	Fe2	Fe3	Mg	Mn	Na	K	Ca	Al	Si	O	H
Quartz									1	2	
Albite					1			1	3	8	
Anorthite							1	2	2	8	
Orthose						1		1	3	8	
Leucite						1		1	2	6	
Nepheline					1			1	1	4	
Kalsilite						1		1	1	4	
Lawsonite							1	2	2	10	4
AmesiteMg			4					4	2	18	8
AmesiteFe	4							4	2	18	8
DaphniteMg			5					2	3	18	8
DaphniteFe	5							2	3	18	8
Sudoite			2					4	3	18	8
PenniniteMg			11					2	7	18	8
PenniniteFe	11							2	7	18	8
Fayalite	2								1	4	
Forsterite			2						1	4	
Enstatite			2						2	6	
Ferrosilite	2								2	6	
Diopside			1				1		2	6	
Hedenbergite	1						1		2	6	
Jadeite					1			1	2	6	
Aegirine		1			1				2	6	
Annite	3					1		1	3	12	2
Phlogopite			3			1		1	3	12	2
Muscovite						1		3	3	12	2
Paragonite					1			3	3	12	2
Margarite							1	4	2	12	2
Grossular							3	2	3	12	
Spessartite				3				2	3	12	
Pyrope	3							2	3	12	
Almandine			3					2	3	12	
AnthophylliteMg			7						8	24	2
AnthophylliteFe	7								8	24	2
TremoliteMg			5				2		8	24	2

Mineral	Simplified formula										
	Fe2	Fe3	Mg	Mn	Na	Ca	Al	Ti	Si	O	H
TschermakiteMg			3			2	4		6	24	2
TschermakiteFe	3					2	4		6	24	2
PargasiteMg			4		1	2	3		6	24	2
PargasiteFe	4				1	2	3		6	24	2
GlaucophaneMg			3		2		2		8	24	2
GlaucophaneFe	3				2		2		8	24	2
Riebeckite	5				2				8	24	2
Gehlenite						2	2		1	7	
Kyanite							2		1	5	
Sillimanite							2		1	5	
Pyrophyllite							1		2	6	1
StauroliteMg			8				36		15	96	8
StauroliteFe	8						36		15	96	8
Talc			3						4	12	2
CordieriteMg			2				4		5	18	
CordieriteFe	2						4		5	18	
ChloritoideMg			1				2		1	7	2
ChloritoideFe	1						2		1	7	2
Wollastonite						1			1	3	
Epidote	1					2	2		3	13	1
Clinozoisite						2	3		3	13	1
Spinel			1				2			4	
Hercynite	1						2			4	
Carpholite			1				2		2	10	4
Brucite			1							2	2
Lime						1				1	
Corindon							2			3	
Periclase			1							1	
Merwinite			1			3			2	8	
Akermanite			1			2			2	7	
Monticellite			1			2			1	4	
Kirschsteinite	1					1			1	4	
AntigoriteMg			3						2	9	4
AntigoriteFe	3								2	9	4
Pyroxmangite				1					1	3	
Rhodonite				1					1	3	

Mineral	Simplified formula												
	Fe2	Fe3	Mg	Na	Ca	Al	Ti	Si	O	H	C	S	Other
Tephroite								1	4				Mn = 2
Diaspore						1			2	1			
Ilmenite	1						1		3				
Rutile							1		2				
Ulvospinel	2						1		4				
Magnetite	1	2							4				
Hematite		2							3				
Pseudobrookite		2					1		5				
Wustite	1								1				
Graphite											1		
Siderite	1								3		1		
Magnesite			1						3		1		
Calcite					1				3		1		
Dolomite			1		1				6		2		
Ankerite	(0-1)		(0-1)		1				6		2		
Barite									4			1	Ba = 1
Pyrite	1											2	
Pyrrhotite	1											1	
Galena												1	Pb = 1
Sphalerite												1	Zn = 1
Millerite												1	Ni = 1
Molybdenite												2	Mo = 1
Chalcopyrite	1											2	Cu = 1
Arsenopyrite	1											1	As = 1
Chromite	1								4				Cr = 2
Zircon								1	4				Zr = 1
Apatite					5				12	1			P = 3, Cl+F = 1
Dravite			3	1		6		6	31	4			B = 3
Schorl		3		1		6		6	31	4			B = 3
FoititeMg			2			7		6	31	4			B = 3
Foitite		2				7		6	31	4			B = 3

B – Reactions and adjustments used by CONSONORM_HG

B.1. Procedure for tourmaline calculation (all elements are in molar proportions)

Tourmaline is calculated if B has been analysed, and successive tourmalines are calculated until B exhaustion using one of the following sequences:

If Na > Al and Mg > Fe2, calculate: dravite, schorl, foititeMg and foitite.
Else if Na > Al and Mg < Fe2, calculate: schorl, dravite, foititeMg and then foitite.
Else if Na < Al and Mg > Fe2, calculate: foititeMg, foitite, dravite and then schorl.
Else if Na < Al and Mg < Fe2, calculate: foitite, foititeMg, schorl and then dravite.

B.2. Spessartine

Garnet is calculated using Fe2+Mn+Mg at first. Then, if (Fe2+Mg)/(Fe2+Mg+Mn) > 0.9, then Mn is used to calculate spessartine. I all cases, Fe2 (or Fe2+Mn) is attributed to almandine, and Mg is used to calculate pyrope.

B.3. White mica and feldspar

If the assemblage contains white micas and alkaline feldspar, K is preferentially allocated to feldspar, while additional Na is allocated to white micas.

B.4. Pargasite

For the amphibolite grade models only, pargasite is formed if the assemblage contains one of the following set of minerals:

- Anorthite, albite, actinolite and a FM mineral (orthopyroxene, olivine, talc, etc.)
- Or anorthite, albite, actinolite and clinopyroxene
- Or anorthite, albite, clinopyroxene, and: grossular, clinozoisite or wollastonite
- Or anorthite, albite, a FM mineral and: amesite, cordierite or pyrope–almandin
- Or cordierite, actinolite and albite
- Or amesite, actinolite and albite

Procedure: Mg, Na, Ca and Al are taken from the minerals enumerated here and are used to calculate as much pargasite as possible.

B.5. Tschermakite

For the amphibolite grade models only, tschermakite is formed after pargasite, only if the assemblage still contains anorthite and a FM mineral (orthopyroxene, anthophyllite, olivine, talc, etc.). The amount of Mg, Ca and Al contained in anorthite and FM-minerals are used to calculate as many tschermakite as possible.

B.6. Glaucophane

For the blueschist grade models only, if “Na-biotite” has been calculated from the ACNKFM tetrahedron (this may happen because Na and K are joined on the same apex

of the tetrahedron), this “mineral” is reacted with a FM-mineral (orthopyroxene, olivine, talc, anthophyllite, etc.) to form as much glaucophane as possible.

B.7. Silicon deficit

At the end of the normative calculation, if the rock does not contain enough silicon for all the minerals calculated, the Si deficit is solved using “reactant” minerals to form “reacted” minerals (see table below). The reactions are made sequentially, in the order indicated by the table, until the Si deficit is solved. Note that if FeO has been produced during Si deficit calculations, it is later transferred to the Fe–Ti oxides.

"Reactant" minerals	"Reacted" minerals	Models
kyanite, sillimanite or pyrophyllite	corundum or diaspore	all
cordierite	orthopyroxene, corundum or diaspore	10GRA950, 9GRA800
cordierite	amesite, corundum or diaspore	3AMP675, 5AMP670, 7AMP700
cordierite	spinel–hercynite	all other
carpholite	amesite, corundum or diaspore	all
glaucophane	albite, brucite or periclase, FeO	all
orthose	leucite	all
albite	jadeite	7AMP500, 5SV400, 10SB350, 9SB450
albite	nepheline	all other
leucite	kalsilite	all
white mica	corundum or diaspore	all
biotite	kalsilite, brucite or periclase, FeO	all
jadeite	nepheline	all
lawsonite	grossular, corundum or diaspore	all
anorthite	grossular, margarite, corundum or diaspore	7AMP600, 10AMP700, 7AMP500, 9SB450, 10SB350, 5SV400
anorthite	grossular, corundum or diaspore	all other
margarite	clinozoisite, corundum or diaspore	all
talc	orthopyroxene	10AMP700
talc	antigorite	7AMP500, 4AMP600, 5SV400, 10SB350, 9SB450
talc	anthophyllite	all other
anthophyllite	orthopyroxene	all

"Reactant" minerals	"Reacted" minerals	Models
actinolite	orthopyroxene, clinopyroxene	5GRA900, 10GRA950
actinolite	clinopyroxene, antigorite	7AMP500, 5SV400, 10SB350, 9SB450
actinolite	clinopyroxene, olivine	all
clinopyroxene	brucite or periclase	all
tschermakite	grossular, penninite, corundum or periclase	all
epidote	grossular, penninite, corundum or periclase	all
staurolite	spinel–hercynite, corundum or periclase	4AMP600, 7AMP600, 6AMP600
staurolite	chloritoide, corundum or periclase	7AMP500
staurolite	pyrope–almandine, corundum or periclase	all other
clinozoisite	grossular, corundum or diaspore	all
antigorite	olivine	4AMP600, 7AMP500
antigorite	brucite or periclase, FeO	all other
akermanite	merwinite, monticellite	all
orthopyroxene	olivine	all
grossular	gehlenite, wollastonite	all
pyrope–almandine	spinel, olivine	all
spessartine	brucite or periclase, corundum or diaspore	all
chloritoide	amesite, corundum or diaspore	10SB350
chloritoide	spinel–hercynite	all other
daphnite	olivine, spinel–hercynite	all
sudoite	amesite, corundum or diaspore	all
amesite	brucite or periclase, corundum or diaspore	10SB350
amesite	brucite or periclase, spinel–hercynite	7AMP600, 7AMP500, 4AMP600, 6AMP600, 9SB450
amesite	olivine, spinel–hercynite	all other
penninite	olivine, spinel–hercynite	all
monticellite	merwinite, brucite or periclase, FeO	all
gehlenite	lime, corundrum or diaspore	all
merwinite	lime, corundrum or diaspore	all
olivine	brucite or periclase, FeO	all
wollastonite	lime	all
rhodonite or pyroxmangite	tephroite	all
tephroite	MnO	all

Table S3. Solid solution models used to calculate the ternary diagrams using the Perple_X software. These ternary diagrams were then integrated in the design of the AFMM, ACKNFM and Fe–Mg–Mn tetrahedral and ternary diagrams

Symbol	Solution	Reference
Anth	antophyllite	Perple_X
Bio(HP)	biotite	Powell and Holland, 1999
Chl(HP)	chlorite	Holland et al., 1998
Cpx(HP)	clinopyroxene	Holland and Powell, 1996
Ctd(HP)	chloritoid	White et al., 2000
Ep(HP)	epidote	Andersson et al., 2002
feldspar	feldspar	Fuhrman and Lindsley, 1988
Gt(HP)	garnet	Holland and Powell, 1998
hCrd	cordierite	Perple_X
MaPa	margarite	Perple_X
Mn-Opx	orthopyroxene	Perple_X
Opx(HP)	orthopyroxene	Holland and Powell, 1996
MuPa	white mica	Chatterjee and Froese, 1975
O(HP)	olivine	Holland and Powell, 1998
Omph(HP)	clinopyroxene	Green et al., 2007
Sp(HP)	spinel	Ganguly and Saxena, 1987
St(HP)	staurolite	Andersson et al., 2002
T	talc	Perple_X
TrTsPg(HP)	clinoamphibole	Wei and Powell, 2003; White et al., 2003
GlTrTsPg	clinoamphibole (models 10SB350 and 9SB450)	Wei and Powell, 2003; White et al., 2003

REFERENCES

Andersson, J.O., Helander, T., Höglund, L., Shi, P.F. & Sundman, B., 2002. Thermo-Calc and DICTRA, Computational tools for materials science. *Calphad*, **26**, 273-312.

Chatterjee, N.D. & Froese, E., 1975. A thermodynamic study of the pseudo-binary join muscovite-paragonite in the system KAlSi3O8-NaAlSi3O8-Al2O3-SiO2-H2O. *American Mineralogist*, **60**, 985-93.

Deer, W.A., Howie, R.A. & Zussman, J., 1962. Rock-forming minerals. (several books)

Franzolin, E., Schmidt, M.W. & Poli, S., 2011. Ternary Ca–Fe–Mg carbonates: subsolidus phase relations at 3.5 GPa and a thermodynamic solid solution model including order/disorder. *Contributions to Mineralogy and Petrology*, **161**(2), 213-227.

Ganguly, J., & Saxena, S.K., 1987. Mixtures and mineral reactions. Springer, Berlin Heidelberg, New York

Green, E., Holland, T. & Powell, R., 2007. An order-disorder model for omphacitic pyroxenes in the system jadeite-diopside-hedenbergite-acmite, with applications to eclogitic rocks. *American Mineralogist*, **92**, 1181-9.

Fuhrman, M.L. & Lindsley, D.H., 1988. Ternary-Feldspar Modeling and Thermometry. *American Mineralogist*, **73**, 201-15.

Haggerty, S.E., 1991. Oxide textures; a mini-atlas. *Reviews in Mineralogy and Geochemistry*, **25**, 129-219.

Holland, T. & Powell, R., 1996. Thermodynamics of order-disorder in minerals. 2. Symmetric formalism applied to solid solutions. *American Mineralogist*, **81**, 1425-37.

Holland, T.J.B. & Powell, R., 1998. An internally consistent thermodynamic data set for phases of petrological interest. *Journal of Metamorphic Geology*, **16**, 309-43.

Holland, T., Baker, J. & Powell, R., 1998. Mixing properties and activity-composition relationships of chlorites in the system MgO-FeO-Al2O3-SiO2-H2O. *European Journal of Mineralogy*, **10**, 395-406.

Lindsey, W.L., 1991. Iron oxide solubilization by organic matter and its effect on iron availability. Iron Nutrition and Interactions in Plants. *Developments in Plant and Soil Sciences*, **43**, 29-36.

McSwiggen, P.L., 1993. Alternative solution model for the ternary carbonate system CaCO3 - MgCO3 - FeCO3. *Physics and Chemistry of Minerals*, **20**(1), 42-55.

Perple_X 2014. website, http://www.perplex.ethz.ch/PerpleX_solution_model_glossary.html

Powell, R. & Holland, T., 1999. Relating formulations of the thermodynamics of mineral solid solutions: Activity modeling of pyroxenes, amphiboles, and micas. *American Mineralogist*, **84**, 1-14.

Wei, C.J. & Powell, R., 2003. Phase relations in high-pressure metapelites in the system KFMASH (K2O-FeO-MgO-Al2O3-SiO2-H2O) with application to natural rocks. *Contributions to Mineralogy and Petrology*, **145**, 301-15.

White, R.W., Powell, R., Holland, T.J.B. & Worley, B.A., 2000. The effect of TiO2 and Fe2O3 on metapelitic assemblages at greenschist and amphibolite facies conditions: mineral equilibria calculations in the system K2O-FeO-MgO-Al2O3-SiO2-H2O-TiO2-Fe2O3. *Journal of Metamorphic Geology*, **18**, 497-511.

White, R.W., Powell, R. & Phillips, G.N., 2003. A mineral equilibria study of the hydrothermal alteration in mafic greenschist facies rocks at Kalgoorlie, Western Australia. *Journal of Metamorphic Geology*, **21**, 455-68.