GENERAL DISCUSSION

EXCERPT FROM A PREVIOUS E-MAIL:

First of all, I believe that before someone engages in the actual programming of a decompression model, they need to do quite a bit of research into the fundamentals behind that model. Usually this means the dissolved gas (Haldane) model as implemented by Buhlmann and/or others. Unfortunately, the relevant information is not conveniently compiled or located in one handy reference. The books by Buhlmann have been the closest thing to an all-in-one reference, but the information is incomplete, especially if you want to program the model.

Buhlmann's work has to be taken in the historical context from which it is derived. Buhlmann did not "invent" most of the concepts that he presents in his books. He took the work done by others in the field before him and refined the model (slightly). The major elements of the dissolved gas model were developed by John S. Haldane, Robert D. Workman (U.S. Navy), and Heinz R. Schreiner (American researcher). Buhlmann relied heavily on the work of Robert Workman and communicated frequently with Schreiner as a colleague in the late 60's and early 70's. Bill Hamilton was a co-worker of Schreiner's in the early years.

Workman, Schreiner, and Buhlmann are deceased now.
Bill Hamilton is still very active in the field and is probably one of the best ongoing sources for information on decompression topics. The key elements of the present day dissolved gas model, however, were laid down in a few research papers many years ago. These papers contain the core fundamentals about the model and its assumptions. This is information that every decompression "programmer" needs to read and to know. The references are listed as follows:

1. Boycott, A.E., Damant, G.C.C., & Haldane, J.S. "The Prevention of Compressed Air Illness," Journal of Hygiene, Volume 8, (1908), pp. 342-443. [This is the classic paper by Haldane and associates which started the field of decompression science. Haldane offers many insights (far ahead of his time) and a few misguided assumptions. There is a lot of talk about decompressing goats! It is well worth the time to read this paper. Much of it is still applicable today. An old 1908 copy of the Journal of Hygiene, Volume 8, can be found in the library of most major universities, especially those involved in medical science].

2. Workman, Robert D. "Calculation of Decompression Schedules for Nitrogen-Oxygen and Helium-Oxygen Dives," Research Report 6-65, U.S. Navy Experimental Diving Unit, Washington, D.C. (26 May 1965). [This paper is available through the National Technical Information Service (NTIS) or a photocopy can be ordered from the Undersea & Hyperbaric Medical Society (UHMS)].

3. Schreiner, H.R., and Kelley, P.L. "A Pragmatic View of Decompression," Underwater Physiology: Proceedings of the Fourth Symposium on Underwater Physiology, edited by C.J. Lambertsen. Academic Press, New York, (1971) pp. 205-219. [This paper is available by finding the book in a major

university library or a photocopy can be ordered from the UHMS].

The reason you should read the above papers is to understand the historical
context and development of the dissolved gas model, which is necessary to understand Buhlmann's implementation of the model. Key points are as follows:

* Haldane established the concept of various "tissue" compartments within the body in which the gas loading behaves according to the law of exponential decay found throughout nature. Haldane also established the concept of "ascent limiting criteria," in his case it was through supersaturation ratios.

* Workman used the research data of the U.S. Navy to establish the the concept of "M-values" for the ascent limiting criteria. These are expressed as a linear relationship between tolerated supersaturation in the "tissue" compartments and ambient pressure. Workman's M-values are based on the partial pressure of the inert gas in question, not on the total pressure of the breathing gas. Workman explained the concept that fast half-time compartments tolerate a greater supersaturation than slow half-time compartments. Workman also developed a detailed calculation procedure which is the foundation of those used today. A colleague of Workman's, William R. Braithwaite, later modified Workman's procedure to include the calculation of "tolerated ambient pressure" as a means to determine a "trial first stop."

* Schreiner explained the decompression model in terms of actual physiological elements such as gas transport in the blood to the tissues, solubility of gases in body fluids, fat fractions and composition of "tissue" compartments, and alveolar partial pressures of gases. He established the very important concept that the total inert gas partial pressure in a compartment is the sum of the partial pressures of all inert gases in that compartment, even if they have different half-times. Another major contribution that Schreiner made was to solve the differential equation for gas exchange when the ambient pressure changes at a constant rate. This is the general solution to the differential equation, of which the familiar instantaneous equation is only a subset. The general solution makes it possible to directly calculate the inert gas partial pressure of a compartment, as a function of time, for any linear (constant depth) or stepwise ascent or descent (at a constant rate). There are many other insights into decompression physiology given in this paper including a basis for the half-time constant, k.

As you can see from the above, many of the key elements used in the "Buhlmann Algorithm" were really developed by others and carried forward by Buhlmann. Of course, Buhlmann made a number of contributions to the science and practice of decompression calculations as well. His greatest contribution was to publish his book, in four editions from 1983 to 1995, as a nearly complete reference on making decompression calculations. Because this was the only "one-stop shopping" book widely available, it became the basis for most of the world's decompression computers and do-it-yourself programs. Key Buhlmann concepts, many explained in the 4th Edition (1995) of his book, are as follows:

* The variation in half-times between two gases is inversely proportional to the square-roots of their molecular weights.

This is a well-known relationship from chemistry called Graham's Law. It is particularly applicable when gases pass through a finely-pored membrane, a process called effusion which is a subset of diffusion.

* The overpressure or supersaturation tolerance in a compartment is based upon the excess volume of gas tolerated by the body in that compartment. The tolerated partial pressures between two different gases in the same compartment will vary according to their solubilities in the transport medium that delivered those gases to that compartment (blood plasma, in this case).

The two key concepts above can be used to derive complete sets of half-times and M-values for other gases such as argon and neon (although due to their higher solubilities when compared to nitrogen and helium, respectively, they offer no substantial benefit for decompression under most scenarios).

* The overall M-value for a compartment with multiple gases, each gas having different M-values, will vary in according to the proportion of each gas present in the compartment.

An explanation about Buhlmann "M-values" needs to be given. First of all, they are traditional M-values just as Workman defined them. Buhlmann simply modified the linear equation to suit his application. He started out with the traditional equation for an M-value in the form $y = mx + b$ and he solved it for x. This gives $x = (y - b)/m$. To get rid of m, the slope, in the denominator he just took the reciprocal and called it "Coefficient b." Traditional M-values are given as $P = m(Pamb) + Mo$ [$y = mx + b$ form], where P = tolerated inert gas partial pressure, m = slope, Pamb = ambient pressure, and Mo = intercept at sea level. Buhlmann expressed the same thing in absolute pressure coordinates.
Buhlmann's Coefficient a is the intercept at Pamb = 0 and Buhlmann's Coefficient b is the reciprocal of the slope. It is easy to convert back and forth between Buhlmann M-values and traditional Workman-style M-values. This is something that I think a lot of people don't understand.

In the 1995 Edition of his book, Tauchmedizin or "Diving Medicine," Buhlmann gives a lot of insight into diving physiology and talks about much of his experience in the field over the years.
He presents many of the results from his experimental research.
In so doing, he also points out some of the shortcomings in the model.
For example, he gives compartment partial pressures calculated at the end of dive series and expresses them in percent of the theoretical values. One thing is clear from his data. This is that, in every test series where incidences of DCS are shown, the affected divers are at a certain percentage less than the theoretical M-values in terms of compartment gas loading upon surfacing. This is usually in the range from 90% to 97%. The situation gets worse for repetitive dives which Buhlmann acknowledges and he cautions that reduction factors must be applied for repetitive diving calculations.
One interpretation of Buhlmann's data is that his M-values do not represent a reliable line between NO SYMPTOMS and MASSIVE SYMPTOMS, but rather they represent a line between a LIMITED NUMBER OF SYMPTOMS and a MASSIVE NUMBER OF SYMPTOMS. This is consistent with most other decompression model experience which acknowledges that an M-value line is a solid line drawn through "a fuzzy gray area."

This kind of information should encourage decompression modelers who use the Buhlmann M-values to incorporate an M-value reduction mechanism that is consistent across the entire ambient pressure range. One such mechanism is by reduction of the M-value Gradient. This is simply the difference between the M-value and ambient pressure.

Another good decompression reference, which I haven't mentioned previously, is Dr. Bruce Wienke. He has published several books which are available from Best Publishing Company. He is the author of the Reduced Gradient Bubble Model (RGBM). His excellent discussion about gradients is primarily focused on bubble models, but it is just as applicable to the dissolved gas model.

Well, I hope I have given you some encouragement to do some further reading into the fundamentals of decompression modeling. Decompression modeling is more than just programming a set of equations out of a book. There are a lot of complex considerations involved. I don't have all the answers, but collectively, we in the diving community can arrive at a lot of
answers by exchanging information.

GAS LOADING CALCULATIONS AND SCHREINER EQUATION

EXCERPT FROM NO. 1 FROM A PREVIOUS E-MAIL:

The fundamental relationship of the dissolved gas model is described by a differential equation:

$$dP/dt = k(Pi - P)$$

which states that the instantaneous rate of change of inert gas pressure (dP/dt) in a hypothetical tissue compartment is proportional to a time constant (k) multiplied by the gradient between the inspired inert gas pressure (Pi) and the present (or initial) compartment inert gas pressure (P). This kind of relationship in common in the natural sciences; Newton's
Law of Cooling, for example. The main principle is that a GRADIENT is the driving force behind the rate of change in the amount of something.

In order to solve this differential equation for P, compartment inert gas pressure, as a function of time, we must use integral calculus. Before we do this, however, there are two conditions we must consider. The first is when the inspired inert gas pressure, Pi, remains constant such as during a constant depth dive profile. The second is when Pi changes with respect to time such as during ascents and descents. In order to simplify the integration in the second case, we will stipulate that the inspired inert gas pressure changes at a constant rate such as with a constant rate of ascent or descent, i.e. 10 msw/min.

Not being able to write out integrals over the e-mail, we will go right to the solutions! In the first case (constant depth), the solution is:

$$P = Po + (Pi - Po)(1 - e^{-kt})$$

This is the "Haldane" equation or the "instantaneous" equation.

This same equation can also be written as:

P = Po + (Pi - Po)(1 - e^(-ln2t/half-time)) or
P = Po + (Pi - Po)(1 - e^(-0.693t/half-time)) or
P = Po + (Pi - Po)(1 - 2^(-t/half-time))

(the latter form appears in Buhlmann's Tauchmedizin book). In the above equations:

P = compartment inert gas pressure (final)
Po = initial compartment inert gas pressure
Pi = inspired compartment inert gas pressure
t = time (of exposure or interval)
k = time constant (in this case, half©time constant)
e = base of natural logarithms
ln2 = natural logarithm of 2

In the second case, ascent or descent at a constant rate, the solution is:

P = Pio + c(t - 1/k) - [Pio - Po - (c/k)]e^-kt

This is the general solution or "Schreiner" equation. It can also be written as:

P = Pio + R(t - 1/k) - [Pio - Po - (R/k)]e^-kt

In the above equations:

Pio = initial inspired (alveolar) inert gas pressure
(Pio = initial ambient pressure minus water vapor pressure)
Po = initial compartment inert gas pressure
c = rate of change in inspired gas pressure with change in ambient pressure
(this is simply rate of ascent/descent times the fraction of inert gas)
R = same as c
t = time (of exposure or interval)
k = half-time constant = ln2/half-time (same as instantaneous equation)

Note that when c (or R) = 0 in the above equation, it is reduced to the more
familiar instantaneous form, P = Po + (Pi - Po)(1 - e^-kt).

EXAMPLES OF APPLYING THE GAS LOADING EQUATIONS

EXCERPT FROM NO. 2 FROM A PREVIOUS E-MAIL:

One "answer" that I would like to give you right away is some advice about how your program calculates for ascent and descent portions of the dive profile. It appears that you, just like most others, are using the instantaneous gas loading equation and "forcing" the computer to calculate the ascent or descent profile in greater resolution by dividing the segment into smaller and smaller increments. This is NOT the correct way to calculate an ascent or descent profile!

The equation P = Po + (Pi - Po)(1 - e^-kt) is applicable only for CONSTANT DEPTH profiles. This is a point that Buhlmann unfortunately does not explain in his book. To use this familiar equation for ascent or descent profiles requires you to break up the interval into very small increments to produce any kind of

accuracy.

The CORRECT and direct calculation for gas loading during ascent and descent profiles is with the general solution given by Schreiner:

P = Pio + R(t - 1/k) - [Pio - Po - (R/k)]e^-kt

Where:

Pio = initial inspired (alveolar) inert gas pressure
(Pio = initial ambient pressure minus water vapor pressure)
Po = initial compartment inert gas pressure
R = rate of change in inspired gas pressure with change in ambient pressure
(this is simply rate of ascent/descent times the fraction of inert gas)
t = time
k = half-time constant = ln2/half-time (same as familiar equation)

Note that when R = 0 in the above equation, it is reduced to the more familiar form, P = Po + (Pi - Po)(1 - e^-kt).

The Schreiner equation is used to compute the partial pressure gas loading for each gas separately during ascent/descent. The sum of these is then the total compartment gas loading. The following examples are subroutines in FORTRAN (should be easy to follow!) from some of my programs which show the correct application of equations.
This subroutine is for ascent/descent segments at a constant rate (i.e. descend to 100 fsw at 50 fsw/min.). Note that an ascent rate must be expressed as a negative number (i.e. -50 fsw/min).

```
      SUBROUTINE ASCDEC (SDEPTH, FDEPTH, RATE)
C
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O
      REAL FDEPTH, SDEPTH, PIHEO, PIN2O, RATE, RTIME, SGTIME, TEMPRT
      REAL HERATE, N2RATE, SPAMB, FPAMB, PAMB
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
      SGTIME = (FDEPTH - SDEPTH)/RATE
      TEMPRT = RTIME
      RTIME = TEMPRT + SGTIME
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
      FPAMB = FDEPTH + 33.0
      SPAMB = SDEPTH + 33.0
      PAMB = FPAMB
      PIHEO = (SPAMB - PH2O)*FHE(MIXNUM)
      PIN2O = (SPAMB - PH2O)*FN2(MIXNUM)
      HERATE = RATE*FHE(MIXNUM)
      N2RATE = RATE*FN2(MIXNUM)
      DO 430 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PIHEO + HERATE*(SGTIME - 1.0/KHE(I)) -
     *         (PIHEO - PTHEO(I) - HERATE/KHE(I))*EXP (-KHE(I)*SGTIME)
      PN2(I) = PIN2O + N2RATE*(SGTIME - 1.0/KN2(I)) -
```

```
     *         (PIN2O - PTN2O(I) - N2RATE/KN2(I))*EXP (-KN2(I)*SGTIME)
430   CONTINUE
      RETURN
      END
```

This next subroutine is for constant depth segments (i.e. at 100 fsw for 15 minutes).

```
      SUBROUTINE CDEPTH (DEPTH, SRTIME)
C
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O, SRTIME
      REAL DEPTH, PAMB, PIHE, PIN2, RTIME, SGTIME, TEMPRT
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
      SGTIME = SRTIME - RTIME
      TEMPRT = SRTIME
      RTIME = TEMPRT
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
      PAMB = DEPTH + 33.0
      PIHE = (PAMB - PH2O)*FHE(MIXNUM)
      PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
      DO 520 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PHEO(I) + (PIHE - PHEO(I))*
     *         (1.0 - EXP (-KHE(I)*SGTIME))
      PN2(I) = PN2O(I) + (PIN2 - PN2O(I))*
     *         (1.0 - EXP (-KN2(I)*SGTIME))
520   CONTINUE
      RETURN
      END
```

With the two above subroutines (subprograms), a complete dive profile can be directly calculated in any combination of ascent/descent segments and constant depth segments (same things as you call waypoints).

EXCERPT FROM NO. 3 FROM A PREVIOUS E-MAIL:

. . . you mention that you use 3 second time slices to calculate with precision for the gas uptake/elimination during ascent and descent segments of the dive profile. This implies that the computer must calculate many iterations to arrive at a total for the interval. This process is unnecessary if you use the Schreiner equation which will directly make an exact calculation for the interval. If you are a student of calculus, you will recognize that the familiar instantaneous equation and the Schreiner equation are definite integrations of the same original differential equation. The difference is that with the instantaneous equation [P = Po + (Pi - Po)(1 - e^-kt)], the inspired partial pressure of the gas, Pi, is assumed to be constant (i.e. as if you are staying at the same constant depth for the interval). The Schreiner equation,
P = Po + R(t - 1/k) - [Pio - Po - (R/k)]e^-kt , was integrated with the inspired partial pressure of the gas changing at a constant rate (such as during ascent or descent). This means that it gives you an exact number for the interval and there is no need

to do "slices" or iterations. Let me give you an example:

Say you are descending from 0 to 120 fsw at 60 fsw/min and you're breathing trimix with 15% O2, 45% He, and 40% N2.

The time, t, for this interval will be
(Final Depth - Initial Depth)/Rate = (120 - 0)/60 = 2 min.

The initial ambient pressure, Pambo, (in absolute coordinates) for this interval will be Depth + 33 = 0 + 33 = 33 fsw.

The inital inspired partial pressure for helium, PiHeo, will be the initial ambient pressure, Pambo, minus the water vapor pressure in the alveoli times the fraction of inert gas -
PiHeo = (Pambo - PH2O)*FHe. The water vapor pressure is always considered to be constant in the alveoli. The value given by Buhlmann and converted to fsw is 2.042 fsw. Thus,
PiHeo = (33 - 2.042)*0.45 = 13.93 fsw.

The inital inspired partial pressure for nitrogen, PiN2o, will be the initial ambient pressure, Pambo, minus the water vapor pressure in the alveoli times the fraction of inert gas -
PiN2o = (Pambo - PH2O)*FN2.
PiN2o = (33 - 2.042)*0.40 = 12.38 fsw.

The rate of change of inspired helium pressure with time, RHe, will be the descent rate, 60 fsw/min, times the fraction of inert gas -
RHe = 60*0.45 = 27 fsw/min.

The rate of change of inspired nitrogen pressure with time, RN2, will be the descent rate, 60 fsw/min, times the fraction of inert gas -
RN2 = 60*0.40 = 24 fsw/min.

Let's say this is the first dive of the day and you have not been diving recently. The inital partial pressure of helium, PHeo, in your compartments is zero. Thus PHeo = 0.0.

The initial partial pressure of nitrogen, PN2o, in your compartments is "saturation" at sea level due to breathing atmospheric air,
PN2o = 33 fsw (absolute) minus water vapor pressure times the fraction of inert gas,
PN2o = (33 - 2.042)*0.79 = 24.46 fsw.

For purposes of this example, let's just look at Buhlmann Compartment No. 1 for helium and nitrogen:

Helium half-time = 1.51 min, Nitrogen half-time = 4.0 min.
The time constant for helium, kHe, is the natural logarithm of 2 divided by the half-time -
kHe = ln2/half-time = 0.693/1.51 = 0.459

The time constant for nitrogen, kN2, is the natural logarithm of 2 divided by the half-time -
kN2 = ln2/half-time = 0.693/4.0 = 0.173

Therefore, to determine what the exact partial pressure for helium and nitrogen in Compartment No. 1 will be upon arrival at a depth of 120 fsw, use the Schreiner equation:

PHe = PiHeo + RHe(t - 1/kHe) - [PiHeo - PHeo - (RHe/kHe)]e^-kHe*t

PHe = 13.93 + 27(2 - 1/0.459) - [13.93 - 0 - (27/0.459)]e^-0.459*2
PHe = 27.03 fsw (absolute)

PN2 = PiN2o + RN2(t - 1/KN2) - [PiN2o - PN2o - (RN2/kN2)]e^-kN2*t
PN2 = 12.38 + 24(2 - 1/0.173) - [12.38 - 24.46 - (24/0.173)e^-0.173*2
PN2 = 28.36 fsw (absolute)

The total compartment inert gas pressure, PIG, will be the sum of PHe and PN2 -
PIG = 27.03 + 28.36 = 55.39 fsw (absolute)

Of course you will need to run the same calculations for each of the 16 compartments.

Now compare the method of this example with that of using the instantaneous equation, P = Po + (Pi - Po)(1 - e^-kt). If you were to divide the interval from 0 to 120 fsw up into slices of 3 seconds each, it would require 40 iterations of the calculation to arrive at the same result. (2 min = 120 seconds divided by 3 second slices = 40 iterations). This means that for 16 compartments and 2 gases, the computer will have to do 40 * 16 * 2 = 1280 calculations. The result that you get will be an approximation to the actual gas loading (i.e. like the trapezoid rule of numerical integration from calculus). Using the Schreiner equation it will only require 32 calculations and produce the EXACT result!

If you do not believe me, then try writing a couple of little programs to test each method and see what you come up with.

WATER VAPOR PRESSURE, RESPIRATORY QUOTIENT, AND ALVEOLAR GAS ADJUSTMENT

COMPOSITION OF ALVEOLAR GAS

Alveolar gas does not have the same concentrations of gases as the breathing gasmix. There are several reasons for the differences. First the alveolar gas is only partially replaced by breathing gas with each breath. Second, oxygen is constantly being absorbed from the alveolar gas. Third, carbon dioxide is constantly diffusing from the pulmonary blood into the alveoli. And fourth, dry breathing gas that enters the respiratory passages is humidified even before it reaches the alveoli.

HUMIDIFICATION OF THE GAS AS IT ENTERS THE RESPIRATORY PASSAGES

Breathing gas normally contains almost no carbon dioxide and little water vapor. However, as soon as the breathing gas enters the respiratory passages, it is exposed to the fluids covering respiratory surfaces. Even before the gas enters the alveoli, it becomes totally humidified. The partial pressure of water vapor at normal body temperature of 37 deg C is 47 mm Hg, which, therefore is the partial pressure of water in the alveolar gas. Since the total pressure in the alveoli cannot rise to more than ambient pressure [without pulmonary barotrauma], this water vapor simply expands the volume of the gas and thereby DILUTES all other gases in the inspired gasmix.

The body maintains a constant water vapor pressure of 47 mm Hg

in the alveoli at normal body temperature. Dry breathing gases are a contributing factor to dehydration when diving since the body must supply water to maintain this vapor pressure.

RESPIRATORY QUOTIENT (ALVEOLAR)

The ratio of carbon dioxide production to oxygen consumption is called the respiratory quotient. Depending on diet or physical exertion, this value ranges from 0.7 to 1.0. From U.S. Navy Diving Manual: 0.9 is a good rule-of-thumb value for making calculations.

FROM BENNETT & ELLIOTT (4TH EDITION):

Alveolar ventilation: the total 'dry gas' pressure in the alveoli, including water vapor pressure, will be essentially equal to barometric or ambient pressure. The water vapor pressure must be subtracted from this. It depends on the alveolar temperature and is 47 mm Hg (6.25 kPa) at 37 deg C.

FROM BUHLMANN (1995 DIVING MEDICINE):

For calculation of the partial pressures of a breathing gas, the water vapor pressure must be subtracted from the total ambient pressure:

Pi (inert gas) = (Pamb - 0.627 bar) x Fraction (inert gas)

Without consideration of the water vapor pressure, a higher partial pressure is calculated for O2, N2, He, etc. In experimental decompression research, without this fact of respiratory physiology, the straightforward calculation of the partial pressures of breathing gases results in higher-than-actual values for the tolerated inert gas overpressure.

FROM HAMILTON DCAP MANUAL (1997):

Alveolar gas: the operational assumptions of DCAP are based on inspired gas values, but the Tonawanda II (dissolved gas model) uses alveolar levels for the computations. This is a major difference between DCAP and some other decompression computational programs.

Rationale for alveolar gas adjustment: inspired gas is diluted by CO2 and water vapor in the lungs. These are not particularly important in the pressures used in diving, but they can become significant in altitude calculations where they make up a greater fraction of the total pressure. At a respiratory quotient assumed to be 0.8, the Alveolar Ventilation Equation yields an alveolar inert gas pressure (see Schreiner and Kelley, 1971):

P alveolar (inert gas) = (Pamb - 37 mm Hg) x Fraction (inert gas)

37 mm Hg = 0.0493 bar = 0.493 msw
37 mm Hg = 0.0487 atm = 1.607 fsw

The appropriate value is subtracted from ambient pressure during DCAP's calculation of alveolar gas.

SUMMARY FOR DECOMPRESSION PROGRAM CALCULATIONS:

As can be seen above, there is no question that water vapor pressure

must be subtracted from ambient pressure when calculating the partial pressure of an inspired (alveolar) breathing gas. The only question is what value to subtract. This question is put into perspective by use of the Alveolar Ventilation Equation:

P alveolar = [((PCO2 x (1 - Rq)) / Rq) + Pamb - PH2O] x Fraction (inert gas)

where Rq = respiratory quotient, PCO2 = carbon dioxide pressure, Pamb = ambient pressure, and PH2O = water vapor pressure.

If we use standard values of PCO2 = 40 mm Hg and PH20 = 47 mm Hg, then for the following respiratory quotients we get:

Rq = 0.8	0.9	1.0
Pamb - 37 mm Hg	Pamb - 42.55 mm Hg	Pamb - 47 mm Hg
Pamb - 0.0493 bar	Pamb - 0.0567 bar	Pamb - 0.0627 bar
Pamb - 0.493 msw	Pamb - 0.567 msw	Pamb - 0.627 msw
Pamb - 1.607 fsw	Pamb - 1.848 fsw	Pamb - 2.041 fsw
(Schreiner value)	(U.S. Navy value)	(Buhlmann value)

So, the decompression modeler must choose between these values with the Buhlmann value being the least conservative, the Schreiner value being the most conservative, and the U.S. Navy value in the middle!

PRESSURE UNITS

The units used to describe depth in diving are units of PRESSURE and not units of LENGTH! This is a common source of confusion and misunderstanding and the conversion between these units is often done incorrectly.

Two systems have evolved in diving regarding pressure units - the system of American usage and the system of European usage. Neither system is particularly "correct" and neither system conforms entirely to SI standards of usage.

In both systems, the pressure units for depth are defined ARBITRARILY and INDEPENDENTLY:

The American usage unit of depth (pressure) is feet of seawater (fsw) and is
DEFINED as,

1 fsw = 1/33 standard atmosphere = 3.0705 kPa = 3.0705 x 10^3 Pascals (N/m^2)

This unit conforms to a specific gravity for sea water of 1.020.

The European usage unit of depth (pressure) is the meter of seawater (msw) and is DEFINED as,

1 msw = 1/10 bar = 10 kPa = 10^4 Pascals (N/m^2)

This conforms to a specific gravity for sea water of 1.027.

Note that the units fsw and msw LOOK LIKE units of LENGTH but they are NOT!

The conversion between these PRESSURE UNITS is 3.2568 fsw/msw and 0.30705 msw/fsw.

The conversion between the linear units of LENGTH is 3.2808 feet/meter and 0.3048 meters/foot. These conversion factors are often INCORRECTLY applied
to convert between fsw and msw.

Sea water ranges in specific gravity between about 1.020 and 1.030. Thus, both the American and European units were chosed to stay within these limits. The density of the water (or the depth for that matter) does not matter if the PRESSURE of the diver is measured and used to determine the proper decompression. Thus a dive in fresh water involves no adjustment as long as the diver's PRESSURE is measured with the same gauge calibration as is used for the decompression calculations.

Some important notes about USAGE CONVENTIONS need to be mentioned:

1. In the American system, sea level is considered to be at STANDARD ATMOSPHERIC PRESSURE = 101.325 kPa = 1.01325 bar = 760 mm Hg.

2. Some users of the American system, including the U.S. Navy and the Journal of Undersea and Hyperbaric Medicine, deviate from the standard definition of fsw and define 1 fsw = 1/33.08 standard atmosphere. However, most users of the American system define 1 fsw = 1/33 standard atmosphere or 33 fsw = 1 atm.

3. In the European system, sea level is considered to be 1.0 bar. The standard atmosphere, however, is 1.01325 bar. This means that the European system differs from the American system by 13.25 mb or 1.3%.

The daily fluctuations in barometric pressure will most likely outweigh the trivial differences between the American system and the European system.

It is important to note, however, that all calculations and measurements should be made in either one system or the other. Do not convert back and forth between the two systems. All calculations should be made against the same base and units should be kept consistent.

Another point worth mentioning is that neither the American unit, fsw, nor the European unit, bar, conform to SI standard usage. Technically, we should all be using the SI unit of pressure, the Pascal. However, the common practices of usage are so entrenched that this is unlikely to change.

Decompression programmers need to keep in mind the conventions of usage when setting up programs to operate in both fsw and msw. For the American system, fsw, sea level will be at 33 fsw = 1 atm. For the European system, msw, sea level will be at 10 msw = 1.0 bar.

Also, decompression programs should make all calculations in the pressure units used for depth. For example, gas loadings and M-values in the American system will all be in fsw. Gas loadings and M-values in the European system will all be in msw. This means that if the Buhlmann M-values are used, for example, the "a" Coefficients need to be converted from bar to fsw or msw as the base data. In the msw case, its easy, just multiply by 10.

PROPER CONVERSION OF FEET OF SEAWATER (FSW) TO METERS OF SEAWATER (MSW)

Prepared by Erik C. Baker
References: ANSI/IEEE Std. 268-1992, ASTM E380-91a, R.W. Bill Hamilton

DEFINITIONS

1. One MSW is DEFINED as 1/10 bar
2. One FSW is DEFINED as 1/33 atm

Note: FSW and MSW are units of PRESSURE and NOT length!

SI CONVERSIONS

SI unit of pressure is the pascal (Pa) = Newtons per square meter

1 bar = 100 kPa
1 atm = 1.013250E+05 Pa (Atmosphere, standard)
1 atm = 1.01325 bar

CONVERSION OF UNITS

$$1\ \text{MSW} = (0.1\ \text{bar}) * \frac{(1\ \text{atm})}{(1.01325\ \text{bar})} * \frac{(33\ \text{FSW})}{(1\ \text{atm})} = 3.25684678\ \text{FSW}$$

SPREADSHEET CALCULATIONS

FSW	MSW	FSW	MSW	FSW	MSW
5	1.5	205	62.9	405	124.4
10	3.1	210	64.5	410	125.9
15	4.6	215	66.0	415	127.4
20	6.1	220	67.6	420	129.0
25	7.7	225	69.1	425	130.5
30	9.2	230	70.6	430	132.0
35	10.7	235	72.2	435	133.6
40	12.3	240	73.7	440	135.1
45	13.8	245	75.2	445	136.6
50	15.4	250	76.8	450	138.2
55	16.9	255	78.3	455	139.7
60	18.4	260	79.8	460	141.2
65	20.0	265	81.4	465	142.8
70	21.5	270	82.9	470	144.3
75	23.0	275	84.4	475	145.8
80	24.6	280	86.0	480	147.4
85	26.1	285	87.5	485	148.9
90	27.6	290	89.0	490	150.5
95	29.2	295	90.6	495	152.0
100	30.7	300	92.1	500	153.5
105	32.2	305	93.6	505	155.1
110	33.8	310	95.2	510	156.6
115	35.3	315	96.7	515	158.1
120	36.8	320	98.3	520	159.7

125	38.4	325	99.8	525	161.2
130	39.9	330	101.3	530	162.7
135	41.5	335	102.9	535	164.3
140	43.0	340	104.4	540	165.8
145	44.5	345	105.9	545	167.3
150	46.1	350	107.5	550	168.9
155	47.6	355	109.0	555	170.4
160	49.1	360	110.5	560	171.9
165	50.7	365	112.1	565	173.5
170	52.2	370	113.6	570	175.0
175	53.7	375	115.1	575	176.6
180	55.3	380	116.7	580	178.1
185	56.8	385	118.2	585	179.6
190	58.3	390	119.7	590	181.2
195	59.9	395	121.3	595	182.7
200	61.4	400	122.8	600	184.2

FOLLOW-UP

As a follow-up to the previous material, I would like to make a few more points. The standard practice for decompression calculations is to calculate everything (applicable) in the pressure units of depth that you are using. This will be either fsw or msw. This e-mail will show examples of what I am talking about.

First, however, a quick point about half-times. Buhlmann published a decompression model (ZH-L16) with sixteen (16) half-times for sixteen (16) compartments. There are actually two sets of values given for Compartment No. 1. Buhlmann calls these 1 and 1b. The difference is that Compartment No. 1 has half-times of 4.0 min N2/1.51 min He and Compartment No. 1b has half-times of 5.0 min N2/1.88 min He. These are NOT two different compartments - you must choose one or the other for Compartment No. 1. The reason that Buhlmann did this is that most other decompression modelers consider 5 min as the fastest half-time for nitrogen and by choosing the Compartment 1b values you will be consistent with the other models. The 4.0 min N2/1.51 min He half-times are too fast for most applications. They would really only be useful where extreme bounce dives are involved. Most technical diving profiles rarely involve the fastest compartment anyway.

Below is an example of initializing these half-times in a FORTRAN program. Note that I have chosen the 1b values for Compartment No. 1:

```
      DATA HALFTH(1)/1.88/,HALFTH(2)/3.02/,HALFTH(3)/4.72/,
     *     HALFTH(4)/6.99/,HALFTH(5)/10.21/,HALFTH(6)/14.48/,
     *     HALFTH(7)/20.53/,HALFTH(8)/29.11/,HALFTH(9)/41.20/,
     *     HALFTH(10)/55.19/,HALFTH(11)/70.69/,HALFTH(12)/90.34/,
     *     HALFTH(13)/115.29/,HALFTH(14)/147.42/,HALFTH(15)/188.24/,
     *     HALFTH(16)/240.03/
      DATA HALFTN(1)/5.0/,HALFTN(2)/8.0/,HALFTN(3)/12.5/,
     *     HALFTN(4)/18.5/,HALFTN(5)/27.0/,HALFTN(6)/38.3/,
     *     HALFTN(7)/54.3/,HALFTN(8)/77.0/,HALFTN(9)/109.0/,
     *     HALFTN(10)/146.0/,HALFTN(11)/187.0/,HALFTN(12)/239.0/,
     *     HALFTN(13)/305.0/,HALFTN(14)/390.0/,HALFTN(15)/498.0/,
     *     HALFTN(16)/635.0/
```

Next, is an example of initializing the value for water vapor pressure. This example uses the Buhlmann value (respiratory quotient = 1.0). For now, I suggest that you use this value since it will be consistent with the rest of the Buhlmann methodology. Please note that this value is

expressed in msw, and NOT IN BAR, because it is standard practice to calculate everything in the pressure units of depth that we are using:

```
      PH2O = 0.627    << expressed in msw !
```

Next, is an example of initializing some other variables (arrays) in FORTRAN. This program example will be for a first dive (not repetitive) and for diving at sea level (no altitude stuff - we'll get to that later). Note that the initial partial pressure for He will be 0.0 in all compartments. The initial partial pressure for N2 in all compartments will be saturation due to breathing atmospheric air AT SEA LEVEL. In the European system, this will be calculated as follows:

Sea level pressure = 1.0 bar = 10 msw

PN2 (saturation) = (Pamb - PH20) * Fraction (inert gas)
= (10 msw - 0.627 msw) * 0.79 = 7.40467 msw

Again, note that we are expressing everything in msw because it is standard practice to calculate in the units of depth being used.

Programming example:

```
      DO 25 I = 1,16
           CMPTMT(I) = I
           KHE(I) = ALOG(2.0)/HALFTH(I)
           KN2(I) = ALOG(2.0)/HALFTN(I)
           PHE(I) = 0.00
           PN2(I) = 7.40467
25    CONTINUE
```

Next is an example of making calculations in the ascent/descent subroutine. For the interval of ascent or descent, the starting ambient pressure (SPAMB) will be the starting depth (SDEPTH) plus the barometric pressure at sea level and the final ambient pressure will be the final depth (FDEPTH) plus barometric pressure at sea level. Note that ambient pressure is always the ABSOLUTE ambient pressure. Again, the pressures are expressed in msw and NOT in bar:

```
      SPAMB = SDEPTH + 10.0   << msw !
      FPAMB = FDEPTH + 10.0   << msw !
```

Similar example for constant depth subroutine:

```
      PAMB = DEPTH + 10.0     << msw !
```

Also, in the constant depth subroutine is an example showing the inspired gas pressures being calculated in the same units as depth. For this example, assume you are at an absolute ambient pressure of 20 msw (2.0 bar) and your breathing trimix 17/33:

```
      PIHE = (PAMB - PH2O)*FHE(MIXNUM)
      PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
```

Thus,

```
      PIHE = (20 msw - 0.627 msw) * 0.33 = 6.393 msw
      PIN2 = (20 msw - 0.627 msw) * 0.50 = 9.686 msw
```

Next is an important point to be made. In technical diving,

the partial pressure of oxygen (PO2) is referenced in terms of atmospheres absolute (ATA or atm), especially when dealing with oxygen toxicity calculations. The scientific data and calculations for oxygen toxicity are all reported in units of atmospheres absolute. Therefore, in the European system, you MUST NOT calculate the partial pressure of oxygen (PO2) based on a sea level atmospheric pressure of 10 msw (1.0 bar) even though you do this for the gas loadings and other calculations. You must calculate the PO2 (in atm) based on the msw value for standard atmosphere (atm):

```
1 atm = 101.325 kPa = 1.01325 bar = 10.1325 msw
```

Thus, for the PO2 calculation (in atm) only:

```
PO2 = (PAMB/10.1325)*FO2
```

Example using absolute ambient pressure of 20 msw (2.0 bar) and trimix 17/33:

```
PO2 = (20 msw/10.1325 msw) * 0.17 = 0.335 atm
```

This value of PO2 (atm) can then be used in oxygen toxicity calculations and the result will be consistent with the units in which the scientific data has been reported.

Also note that we DO NOT subract water vapor pressure from the ambient pressure when calculating the PO2. This is in order to give a "worst case" value and to match PO2 calculations made by hand.

A final example to emphasize the standard procedure of performing decompression calculations in the units of depth being used. The Buhlmann M-value Coefficient a, which is reported in units of bar, must be converted to a "base" unit in the depth units being used. [The Buhlmann Coefficient b does not require any conversion because it is dimensionless (the slope) and is the same for any pressure unit being used]. In the following example, the Buhlmann Coefficient a is converted to msw (multiply the bar value by 10) and becomes the base unit in a FORTRAN block data subprogram which initializes these variables in arrays in the program. Also note that in this case, the values in Compartment No. 1 are the Buhlmann 1b values for the 5.0 min N2/1.88 min He half-times:

```
      BLOCK DATA COEFFS
      REAL AHE, BHE, AN2, BN2
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      COMMON /A/ AHE, BHE, AN2, BN2
      DATA AHE(1)/16.189/,AHE(2)/13.830/,AHE(3)/11.919/,AHE(4)/10.458/,
     *     AHE(5)/9.220/,AHE(6)/8.205/,AHE(7)/7.305/,AHE(8)/6.502/,
     *     AHE(9)/5.950/,AHE(10)/5.545/,AHE(11)/5.333/,
     *     AHE(12)/5.189/,AHE(13)/5.181/,AHE(14)/5.176/,
     *     AHE(15)/5.172/,AHE(16)/5.119/
      DATA BHE(1)/0.4770/,BHE(2)/0.5747/,BHE(3)/0.6527/,BHE(4)/0.7223/,
     *     BHE(5)/0.7582/,BHE(6)/0.7957/,BHE(7)/0.8279/,BHE(8)/0.8553/,
     *     BHE(9)/0.8757/,BHE(10)/0.8903/,BHE(11)/0.8997/,
     *     BHE(12)/0.9073/,BHE(13)/0.9122/,BHE(14)/0.9171/,
     *     BHE(15)/0.9217/,BHE(16)/0.9267/
      DATA AN2(1)/11.696/,AN2(2)/10.000/,AN2(3)/8.618/,AN2(4)/7.562/,
     *     AN2(5)/6.667/,AN2(6)/5.600/,AN2(7)/4.947/,AN2(8)/4.500/,
```

```
*     AN2(9)/4.187/,AN2(10)/3.798/,AN2(11)/3.497/,
*     AN2(12)/3.223/,AN2(13)/2.850/,AN2(14)/2.737/,
*     AN2(15)/2.523/,AN2(16)/2.327/
      DATA BN2(1)/0.5578/,BN2(2)/0.6514/,BN2(3)/0.7222/,BN2(4)/0.7825/,
*     BN2(5)/0.8126/,BN2(6)/0.8434/,BN2(7)/0.8693/,BN2(8)/0.8910/,
*     BN2(9)/0.9092/,BN2(10)/0.9222/,BN2(11)/0.9319/,
*     BN2(12)/0.9403/,BN2(13)/0.9477/,BN2(14)/0.9544/,
*     BN2(15)/0.9602/,BN2(16)/0.9653/
      END
```

EXAMPLE FORTRAN CODE AND EXPLANATION FOR DECOMPRESSION CALCULATION PROGRAM
Presented by Erik C. Baker

Note: The following FORTRAN code represents a complete and functional decompression calculation program. The purpose here is to show the basic procedures involved in decompression calculations. There are many other parameters usually involved in a typical desktop program such as oxygen toxicity calculations, gas consumption calculations, equivalent narcotic depth, etc. However, since those are separate topics from decompression calculations, they are not included in this program example.

VERY IMPORTANT NOTE: The decompression program must calculate the entire dive and decompression profile in the SAME system of pressure units. This will be either the American System of Pressure Units (feet of seawater, fsw) or the European System of Pressure Units (meters of seawater, msw) [see separate explanation about Pressure Units]. DO NOT MIX UNITS of different systems in the same program! DO NOT ATTEMPT TO CONVERT BETWEEN UNITS of different systems in the same program! The standard convention in decompression calculations is to calculate all pressures in the same units that are used for depth pressure (i.e. fsw or msw). DO NOT MIX UNITS like atm/ATA with fsw or bar with msw. Keep in mind that most of the pressures and partial pressures will be in terms of absolute pressure and NOT depth pressure. The depth pressures will be converted to absolute pressures to perform most of the calculation sequences.

The character variables in the main program and subroutines are:

COMAND A command to be passed to the MS-DOS operating system.
LINE1 A one line description of the dive.
WORD Either the word "ascent" or "descent" depending on the profile.

The integer variables in the main program and subroutines are:

MIXNUM The number of the gas mix (used to address array elements with fractions of the various gases in each mix).
NUMMIX The total number of gas mixes to be used on this dive.
PROFIL The profile code used to identify ascent/descent or constant depth segments of the dive. A profile code of 99 will invoke the decompression sequence.
SEGNUM The segment number. The complete dive is broken down into discrete segments of ascent/descent profiles and constant depth profiles.

TEMPSG Temporary variable used to update segment number.

The real number variables in the main program and subroutines are:

AHE Buhlmann Coefficient "a" (intercept) for helium (array variable with 16 elements).
AHEN2 Buhlmann Coefficient "a" (intercept) for both helium and nitrogen (array variable with 16 elements).
AN2 Buhlmann Coefficient "a" (intercept) for nitrogen (array variable with 16 elements).
BHE Buhlmann Coefficient "b" (slope) for helium (array variable with 16 elements).
BHEN2 Buhlmann Coefficient "b" (slope) for both helium and nitrogen (array variable with 16 elements).
BN2 Buhlmann Coefficient "b" (slope) for nitrogen (array variable with 16 elements).
CEILNG The deco ceiling (in depth pressure). This the shallowest depth for safe ascent without requiring a decompression stop.
CHANGE A depth pressure where a change in one of the decompression parameters will take place.
CKSUM Check sum used to verify correct gas fractions in gas mixes.
COUNT Counter variable used to compute segment time on deco stops.
DECORT Deco run time.
DEPTH Depth pressure.
FACTOR The Gradient Factor.
FCTRHI The Hi Gradient Factor.
FCTRLO The Lo Gradient Factor.
FCTRSL The slope parameter in a linear function which determines the change in Gradient Factor with change in deco stop depth.
FDEPTH Final depth pressure.
FHE Fraction of helium (array variable).
FN2 Fraction of nitrogen (array variable).
FO2 Fraction of oxygen (array variable).
FPAMB Final ambient pressure, absolute.
FSUM Sum of the fractions of the gases in a gas mix.
HALFTH Half-time for helium (array variable with 16 elements).
HALFTN Half-time for nitrogen (array variable with 16 elements).
HERATE Rate of change in inspired helium pressure.
KHE Time constant for helium (array variable with 16 elements).
KN2 Time constant for nitrogen (array variable with 16 elements).
MVALUE Buhlmann M-value in absolute pressure (array variable with 16 elements).
N2RATE Rate of change in inspired nitrogen pressure.
NXSTOP The depth pressure of the next deco stop.
O2DECO Oxygen deco factor.
PAMB Ambient pressure, absolute.
PAMBT Tolerated ambient pressure, absolute (array variable with 16 elements).
PERCMV Percent M-value (array variable with 16 elements).
PHE Partial pressure of helium, absolute (array variable with 16 elements).
PHEN2 Partial pressure of both helium and nitrogen, absolute (array variable with 16 elements).
PHEO Initial partial pressure of helium, absolute (array variable with 16 elements).
PH2O Water vapor pressure, absolute.

PIHE Inspired partial pressure of helium, absolute.
PIHEO Initial inspired partial pressure of helium, absolute.
PIN2 Inspired partial pressure of nitrogen, absolute.
PIN2O Initial inspired partial pressure of nitrogen, absolute.
PMVMAX Maximum Percent M-value across all 16 compartments.
PN2 Partial pressure of nitrogen, absolute (array variable with 16 elements).
PN2O Initial partial pressure of nitrogen, absolute (array variable with 16 elements).
RATE Rate of ascent (negative value) or descent (positive value).
ROUND Temporary variable used to round up run time to whole minute.
RTIME Run time.
SAFEAD Safe ascent depth pressure (array variable with 16 elements).
SDEPTH Starting depth pressure.
SGTIME Segment time.
SPAMB Starting ambient pressure, absolute.
SRTIME Run time at the end of a segment.
STEPSZ Step size between decompression stops (depth pressure increments).
STOPD Depth pressure of decompression stop.
STOPGF The Gradient Factor used to determine a particular deco stop.
STOPT Stop time.
TEMP1 Temporary variable.
TEMP2 Temporary variable.
TEMPRT Temporary variable used to update run time.
TEMPST Temporary variable used to update segment time.
TRIALD Trial depth pressure.

DESCRIPTION OF THE MAIN PROGRAM

As with any program, the first step is to identify the program and declare the variable types:

```
      PROGRAM DECOCALC
C     BUHLMANN 16 COMPARTMENTS, ZH-L16B M-VALUES (ZH-L16A He, ZH-L16B N2)
C
      CHARACTER COMAND*3, WORD*7, LINE1*70
      INTEGER NUMMIX, PROFIL, SEGNUM, MIXNUM
      REAL RTIME, PAMB, PH2O, FACTOR, CEILNG, STOPD, STEPSZ
      REAL FO2, FHE, FN2, FSUM, CKSUM, CHANGE, PMVMAX, SGTIME
      REAL PHE, PN2, HALFTH, HALFTN, KHE, KN2, FCTRHI, FCTRLO, FCTRSL
      REAL DEPTH, FDEPTH, SDEPTH, RATE, SRTIME, DECORT, STOPT
      REAL O2DECO, TEMP1, TEMP2, NXSTOP, STOPGF, TRIALD
```

Next, the decompression program will make extensive use of one-dimensional arrays (subscripted variables) to do the calculations. The arrays are dimensioned (i.e. number of array elements) below. Note that most of the arrays will be dimensioned to sixteen (16) elements to correlate with the 16 half-time compartments in the Buhlmann decompression model. The fractions of oxygen (FO2), helium (FHE), and nitrogen (FN2) are dimensioned to the number of different gas mixes permitted for use in the program (in this case 10).

```
      DIMENSION FO2(10), FHE(10), FN2(10), PHE(16), PN2(16)
      DIMENSION HALFTH(16), HALFTN(16), KHE(16), KN2(16)
```

For greatest computing efficiency, many of the variables will be common to both the main program and several subroutines (sub-programs). These common variables (and arrays) are declared below. Note that in FORTRAN, the common variables and arrays are organized into lettered COMMON BLOCKS.

```
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB, /F/ FACTOR, /J/ O2DECO
```

Next, the values for the sixteen (16) Buhlmann half-times (in minutes) are assigned to the subscripted elements in the arrays for the helium half-times (HALFTH) and the nitrogen half-times (HALFTN). In FORTRAN, this is accomplished with the DATA statement. Note that the first array elements are using Buhlmann's half-times for Compartment No. 1b (i.e. helium half-time = 1.88 min, nitrogen half-time = 5.0 min). The reason for this is that the 5 minute compartment for nitrogen is consistent with most other decompression models and, typically, the 4 minute half-time for nitrogen is too fast for most technical diving applications. Also note that these half-times will be the same regardless of whether the program is calculating in the American System of Pressure Units (feet of seawater, fsw) or the European System of Pressure Units (meters of seawater, msw). [see separate explanation about Pressure Units].

```
      DATA HALFTH(1)/1.88/,HALFTH(2)/3.02/,HALFTH(3)/4.72/,
     *     HALFTH(4)/6.99/,HALFTH(5)/10.21/,HALFTH(6)/14.48/,
     *     HALFTH(7)/20.53/,HALFTH(8)/29.11/,HALFTH(9)/41.20/,
     *     HALFTH(10)/55.19/,HALFTH(11)/70.69/,HALFTH(12)/90.34/,
     *     HALFTH(13)/115.29/,HALFTH(14)/147.42/,HALFTH(15)/188.24/,
     *     HALFTH(16)/240.03/
      DATA HALFTN(1)/5.0/,HALFTN(2)/8.0/,HALFTN(3)/12.5/,
     *     HALFTN(4)/18.5/,HALFTN(5)/27.0/,HALFTN(6)/38.3/,
     *     HALFTN(7)/54.3/,HALFTN(8)/77.0/,HALFTN(9)/109.0/,
     *     HALFTN(10)/146.0/,HALFTN(11)/187.0/,HALFTN(12)/239.0/,
     *     HALFTN(13)/305.0/,HALFTN(14)/390.0/,HALFTN(15)/498.0/,
     *     HALFTN(16)/635.0/
```

Other variables must be initialized as well. First, the water vapor pressure (PH2O), absolute, is assigned a value. This value will be expressed in either fsw or msw, depending on which system of pressure units (American or European) the program is calculating in. The value to be used will depend on therespiratory quotient (Rq) that was used in the Alveolar Ventilation Equation [see separate explanation about Water Vapor Pressure, Respiratory Quotient, and Alveolar Gas Adjustment].

```
      PH2O = 1.848 [fsw, Rq = 0.9] or
      PH2O = 0.567 [msw, Rq = 0.9]
```

The run time and segment number are initialized to zero.

```
      RTIME = 0.0
      SEGNUM = 0
```

Next is an important step. The time constants, k for helium (KHE) and k for nitrogen (KN2) will be computed for each compartment and the values are assigned into subscripted array elements. The time constant for each compartment is the natural logarithm of 2 divided by the half-time. After this, half-times will no longer be used in any of the gas loading calculations! All of the gas loading calculations will be based on the time constants, k, for each of the compartments for helium and nitrogen. This step is accomplished with a program loop (a DO loop in FORTRAN).

```
      DO 10 I = 1,16
           KHE(I) = ALOG(2.0)/HALFTH(I)
           KN2(I) = ALOG(2.0)/HALFTN(I)
10    CONTINUE
```

The initial values for the partial pressure of helium (PHE) and the partial pressure of nitrogen (PN2) must be assigned for each compartment. In this case, the program is set up for diving at sea level (not a dive at altitude) and it is for a first dive (not a repetitive dive). Therefore, the initial partial pressure for helium will be zero in all compartments (since the diver has been breathing air at the surface) and the initial partial pressure for nitrogen will be the "saturation" value due to breathing air at the surface for a long time. This "saturation" value is computed as follows: PHE = (Pamb - PH2O)*FHEair and PN2 = (Pamb - PH2O)*FN2air, where Pamb is the absolute ambient pressure, PH2O is the water vapor pressure, and Fair is the fraction of inert gas (helium or nitrogen) in atmospheric air. In the American System, the absolute ambient pressure at sea level is 33 fsw, in the European System it is 10 msw [see separate explanation about Pressure Units].
VERY IMPORTANT NOTE: In decompression calculations, all partial pressures are calculated in values of ABSOLUTE PRESSURE! They are NOT calculated in the values of depth pressure! Since the fraction of helium in atmospheric air is zero, the initial value for PHE will be zero. The fraction of nitrogen in atmospheric air is 79%. For initial PN2 in the American System, PN2 = (33 fsw - 1.848 fsw)*0.79 = 24.61 fsw [absolute]. For the European System, PN2 = (10 msw - 0.567 fsw)*0.79 = 7.452 msw [absolute]. Again, the values are assigned using a program loop:

```
      DO 11 I = 1,16
           PHE(I) = 0.00
           PN2(I) = 24.61 [fsw] or
           PN2(I) = 7.452 [msw]
```

```
11    CONTINUE
```

The next portion of code tells the program where to get the input data from.
Of course this will depend on what type of computer is being used (desktop versus in-water dive computer, etc.) and what operating system environment it
is running in (mainframe, MS-DOS, Windows, etc.). In this case, it is a very
simple feed-thru program running in a 32-bit extended MS-DOS or MS-Windows environment. The input data is located in an input file, DECOCALC.IN, in the
same directory. The output data is written to an output file, DECOCALC.OUT,
in the same directory. Some of the command lines are unique to the Microsoft
FORTRAN Development System.

```
      COMAND = 'CLS'
      CALL SYSTEMQQ (COMAND)
      PRINT *,' '
      PRINT *,'PROGRAM DECOCALC'
      PRINT *,' '
      OPEN (UNIT = 7, FILE = 'DECOCALC.IN', STATUS = 'UNKNOWN',
     *         ACCESS = 'SEQUENTIAL', FORM = 'FORMATTED')
      OPEN (UNIT = 8, FILE = 'DECOCALC.OUT', STATUS = 'UNKNOWN',
     *         ACCESS = 'SEQUENTIAL', FORM = 'FORMATTED')
```

The program can now read in data including any descriptive information about
the dive to be calculated. This description along with a heading is written
to the output file.

```
      READ (7,801) LINE1
      WRITE (8,802)
      WRITE (8,800)
      WRITE (8,803) LINE1
      WRITE (8,800)
```

Now the program must read in the data - first to calculate the dive profile,
and then the decompression profile. In this case, all of the input data is
pre-formatted within the input file. The first series of data is the number
of gas mixes (NUMMIX) being used on the dive and the fractions of oxygen (FO2), helium (FHE), and nitrogen (FN2) for each gas mix. The program reads
these into subscripted array elements and then checks to make sure that all of
the gas fractions add up to 100%. This is accomplished with a program loop.

```
      READ (7,*) NUMMIX
      DO 45 I = 1, NUMMIX
           READ (7,*) FO2(I), FHE(I), FN2(I)
           FSUM = FO2(I) + FHE(I) + FN2(I)
           CKSUM = FSUM
           IF (CKSUM .NE. 1.0) THEN
```

```
               CALL SYSTEMQQ (COMAND)
               PRINT *,' '
               PRINT *,'ERROR IN INPUT FILE (GASMIX DATA) - PROGRAM TERM
     *INATED'
               PRINT *,' '
               GOTO 350
            END IF
45    CONTINUE
```

The gas mix data is then written to the output file using a program loop.

```
      WRITE (8,810)
      DO 55 J = 1, NUMMIX
          WRITE (8,811) J, FO2(J), FHE(J), FN2(J)
55    CONTINUE
```

Some decompression modelers do not believe that high oxygen mixes (80%-100%) provide full benefit from an off-gassing standpoint. An oxygen deco factor (O2DECO) may be employed which acts to increase the fraction of nitrogen used in the gas loading calculations. This value is now read into the program and written to the output file. The heading for the dive profile portion of the dive is also written to the output file as well.

```
      READ (7,*) O2DECO
      WRITE (8,800)
      WRITE (8,812) O2DECO
      WRITE (8,800)
      WRITE (8,820)
      WRITE (8,800)
      WRITE (8,821)
      WRITE (8,822)
      WRITE (8,823)
      WRITE (8,824)
```

Next, the program reads in the dive profile data. This is where a lot of variation will come into play depending on the type of computer, operating environment, etc. In a commercial desktop program such as Abyss or Voyager, the dive profile can be entered with a mouse as "waypoints" on a graphics screen. For an in-water dive computer, the data will come from pressure transducers and other sensors in the housing (or remote tank unit). In this program, the input file data is set up as a series of dive "segments" which are similar to "waypoints." Each segment will be either an ascent/descent or a constant depth profile. For example: Segment No. 1 - descent from surface to a depth of 100 fsw at a rate of 60 fsw/min using Gas Mix No. 1. Segment No. 2 - constant depth at 100 fsw for 20 minutes using Gas Mix No. 1. Segment No. 3 - ascent from 100 fsw to 60 fsw at a rate of -20 fsw/min using Gas Mix No. 2 . . . and so on. The entire dive profile is constructed of these

individual segments in any combination of ascent/descent or constant depth profiles. An ascent/descent segment is identified with a profile code (PROFIL) = 1 and a constant depth segment is identified with a profile code =
2. When it is time to decompress to the surface, the profile code = 99. The
main purpose of this part of the program is to track (compute) the gas loadings (i.e. PHE and PN2) in each of the compartments during the dive. To
do this for an ascent/descent segment, the following minimum data is required:
starting depth (SDEPTH), final depth (FDEPTH), rate of ascent/descent (RATE),
and the number of the gas mix (MIXNUM) [which is used to address the array elements containing the fractions of helium and nitrogen for that gas mix].
For a constant depth segment, the following minimum data is required: depth
(DEPTH), time at this depth which can be given as the segment time (SGTIME) or
as the run time at the end of the segment (SRTIME), and the number of the gas
mix (MIXNUM). The gas loading calculations for each segment are performed in
separate subroutines depending on whether it is an ascent/descent segment (subroutine ASCDEC) or a constant depth segment (subroutine CDEPTH) [see separate explanation about Gas Loading Calculations and Schreiner Equation].
The subroutines are invoked with the CALL statement in FORTRAN. The use of
subroutines makes the program run very fast and efficiently. The following
program sequence uses an IF-THEN-ELSE structured block in a GOTO loop (Line
100). The program escapes from this sequence when it encounters a profile code = 99 which then sends it to Line 200, the start of the decompression sequence. After each ascent/descent or constant depth segment, summary data
is written to the output file.

```
100   CONTINUE
      READ (7,*) PROFIL
      IF (PROFIL .EQ. 1) THEN
           READ (7,*) SDEPTH, FDEPTH, RATE, MIXNUM
           CALL ASCDEC (SDEPTH, FDEPTH, RATE)
           IF (FDEPTH .GT. SDEPTH) THEN
               WORD = 'Descent'
           ELSE IF (SDEPTH .GT. FDEPTH) THEN
               WORD = 'Ascent '
           ELSE
               WORD = 'ERROR'
           END IF
           WRITE (8,830) SEGNUM, SGTIME, RTIME, MIXNUM, WORD, SDEPTH,
     *     FDEPTH, RATE
      ELSE IF (PROFIL .EQ. 2) THEN
           READ (7,*) DEPTH, SRTIME, MIXNUM
           CALL CDEPTH (DEPTH, SRTIME)
           WRITE (8,831) SEGNUM, SGTIME, RTIME, MIXNUM, DEPTH
      ELSE IF (PROFIL .EQ. 99) THEN
           GOTO 200
```

```
      ELSE
           CALL SYSTEMQQ (COMAND)
           PRINT *,' '
           PRINT *,'ERROR IN INPUT FILE (PROFILE CODE) - PROGRAM TERMINA
     *TED'
           PRINT *,' '
           GOTO 350
      END IF
      GOTO 100
```

The subroutines for gas loading calculations are separate subprograms located
after the end of the main program. They are explained separately after this
explanation of the main program.

All segments of the dive profile have been completed and it is now time to begin the decompression sequence (profile code = 99). This is where things
can get very complicated in a decompression program. Several reasons and considerations for this are as follows:

1. This particular program assumes that the user (basically only myself) is
very knowledgeable about decompression and diving and won't exceed any
deco limits DURING the dive profile. Usually, this will not be a problem unless an ascent is made during the dive profile such as during
a MULTI-LEVEL dive. If this is the case, then the program will need to
compare the gas loadings against the M-values (i.e. verify the deco ceiling) before any ascent is made, especially after a switch in the gas
mix.

2. Every decompression program must have some methodology for conservatism
to account for individual variations in physiology and tolerances to decompression diving. Generally, people who are overweight and/or in poor physical conditioning will require a substantial conservatism factor. The popular methodologies for conservatism include increasing
the gas fractions used in the calculations, applying a depth safety factor which calculates for a deeper-than-actual dive depth [this is the
method that Buhlmann prescribes in his books], calculating for a longer-
than-actual bottom time, and adjusting the half-times to be asymmetrical
(slower) during off-gassing. In the course of my analyses and research
into decompression, I have found the above methodologies to be problematic in a number of areas including inconsistent conservatism results (on a Percent M-value basis) between short/shallow dives and long/deep dives. Furthermore, these methodologies do not take into account the current knowledge from bubble mechanics. This is why I developed the Gradient Factor method for conservatism [see my article on

"Deep Stops"]. Regardless of what conservatism method is used, it will make the decompression programming somewhat more complicated.

3. The behavior of the gas loadings is NOT intuitive! The uptake and elimination of inert gases (gas loadings) are described by exponential polynomial equations. Sometimes this produces very unexpected results (particularly when both helium and nitrogen are involved). You CANNOT make any assumptions about when a profile is "safe" or not. The program must incorporate a consistent means of "checking" the profile to ensure that the M-values (or M-values modified by conservatism factors) are never exceeded. Take for example an extreme bounce dive. Even if the bottom time is short, it is quite possible that most compartments will continue to ON-GAS DURING THE ASCENT from the bottom! This means that the program must be able to halt the ascent anytime it approaches a deco ceiling (i.e. leading compartment gas loading approaches an M-value).

4. There are a number of ways to program the decompression sequence to be "failsafe." A more complicated way (but at the same time much more flexible) is through the use of "projection subroutines." These are several separate subroutines which "project" what the gas loadings (and other parameters) will be at a future point in time (the common variables in the program are NOT updated in these subroutines). These can be very useful (and fast) for finding critical points in the decompression profile such as the crossing of the ambient pressure line, the deepest possible stop depth, and the first stop that will not exceed an M-value (or modified M-value). This is particularly applicable when two or more inert gases are in use such as helium and nitrogen. The problem with "projection subroutines" is that they are tedious and complicated to program and require a few hundred lines of additional code. For the purposes of this presentation, I will not explain this option further.

5. It is NOT possible to directly determine a "NO-STOP TIME" if more than one inert gas is involved. I realize that Buhlmann presents a formula for this in his book (which is the Haldane equation rearranged), but that is for one inert gas ONLY. When two or more inert gases (i.e. helium and nitrogen) are in the compartments, then two sets of exponential polynomial equations are involved and there are an infinite number of solutions for the "NO-STOP TIME" [i.e. it is IMPOSSIBLE to solve directly!]. This means that the power of the computer must be

employed to verify the decompression requirement all the way back to the surface. With a desktop program, a separate decompression profile must be calculated for each dive profile (including each different combination of deco mixes). Currently, there is no in-water decompression computer, commercially available, with the battery/processor power to do such calculations in real-time.

6. The most common method of keeping the decompression situation under control during ascent is to proceed in small increments or STEP SIZES. Usually these are the standard stop depth increments of 10 fsw or 3 msw. Also, the convention is to clear the M-values for the NEXT stop depth BEFORE YOU LEAVE the present stop depth. Mathematically, there are a couple ways to accomplish this. One method is to "shift" the M-values deeper by one stop depth (or STEP SIZE) such as is done in the DCAP program. The more common method is the "LOOK AHEAD" method which I will present here.

The Decompression Sequence:

The first step is to input the data pertaining to the decompression sequence. In some programs this will already be accomplished with initialization files, default menus, or similar means. In this "feed-thru" program, the data is read from the input file starting after Line 200. The pertinent data is the present or starting depth (SDEPTH), the number of the gas mix (MIXNUM), the rate to be used during the ascent (RATE), and the increment or step size (STEPSZ) to be used between deco stops. In most programs, the step size will be fixed to the standard 10 fsw or 3 msw. This is the safest policy unless the program is equipped with "projection subroutines" to deal with the strange happenings that can result from "big" step sizes. The next two parameters are the Hi Gradient Factor (FCTRHI) and the Lo Gradient Factor (FCTRLO) which are used in the Gradient Factor Method for conservatism [which is included as an option in this presentation]. Finally, the depth (CHANGE) at which a change in any of the decompression parameters will take place is read-in. Basically, a change in the gas mix, ascent rate, or step size can take place at any pre-determined deco stop.

```
200   CONTINUE
      READ (7,*) SDEPTH
      READ (7,*) MIXNUM, RATE, STEPSZ, FCTRHI, FCTRLO
      READ (7,*) CHANGE
```

The heading for the decompression profile portion of the dive is written to
the output file. The deco run time variable is initialized to zero.

```
      WRITE (8,800)
      WRITE (8,840)
      WRITE (8,800)
      WRITE (8,841)
      WRITE (8,842)
      WRITE (8,843)
      WRITE (8,844)
     DECORT = 0.0
```

Optional: the current Gradient Factor (FACTOR) is set to the Lo Gradient Factor to introduce the first "deep stop" [see my article on "Deep Stops"].

```
      FACTOR = FCTRLO
```

The next part of the decompression sequence is the most difficult since there
are several different ways to approach the problem. This is where the PHILOSOPHY of the decompression programmer comes into play. What I mean by
this is that the programmer must make certain decisions about what this program is being used for, what kind of divers will be using the program, and
whether the decompression programming sequence will be either complicated yet
flexible or straightforward but not-very-flexible. The crux of the problem is
determining what the current "deco ceiling" is BASED ON THE PRESENT GAS LOADING and then proceeding with the ascent WHILE MAKING SURE THAT AN M-VALUE
(OR MODIFIED M-VALUE) WILL NOT BE EXCEEDED during the ascent. Again, this gets back to the fact that the behavior of the gas loadings is determined by
exponential polynomial equations (NOT intuitive!). Under certain scenarios,
such as a deep bounce dive, there is a distinct possibility that the deco ceiling (safe ascent depth) calculated at the end of the bottom time will change (become deeper) during the ascent from the bottom. This is because the
compartments can ON-GAS DURING THE ASCENT in certain instances (bounce dive,
change in gas mixes, etc). So, the question is how to build in safeguards against this problem. The more complicated (yet flexible) approach is through
the use of "projection subroutines" which predict what the future conditions
will be. The more straightforward (yet inflexible) approach is to proceed with the ascent in small increments (step sizes) and check the deco ceiling
status at each step. We will proceed with the latter approach here in what is
known as the "LOOK AHEAD" method.

The Trial Depth Ascent Process:

The first step is to determine a TRIAL DEPTH for ascent. In keeping with the cautious approach of ascending in small increments, we will select the first trial depth (TRIALD) to be the next shallower standard stop depth above our present or starting bottom depth (SDEPTH). To compute this, we take SDEPTH and "truncate" it to the next shallower standard stop depth. In the American System, fsw, this will be based on a standard increment of 10 fsw:

```
      TEMP1 = (SDEPTH/10.0) - 0.5
      TRIALD = AINT(TEMP1) * 10.0
```

In the European System, msw, this will be based on a standard increment of 3 msw:

```
      TEMP1 = (SDEPTH/3.0) - 0.5
      TRIALD = AINT(TEMP1) * 3.0
```

The trial depth (TRIALD) should now be checked to make sure that it is not less than zero (the surface).

```
      TEMP2 = TRIALD
      IF (TEMP2 .LE. 0.0) THEN
           TRIALD = 0.0
      END IF
```

Next, we determine what the current deco ceiling is (CEILNG). To do this we use the subroutine SAFASC (for safe ascent). It is explained separately after this explanation of the main program.

```
230   CALL SAFASC (CEILNG)
```

This has returned the value of the current deco ceiling (CEILNG). The trial depth (TRIALD) must now be checked to make sure that it is not shallower than the current deco ceiling. If so, then the present depth becomes a deco stop depth, the trial depth becomes the next stop, and the program jumps ahead to the decompression stop sequence.

```
      IF (CEILNG .GT. TRIALD) THEN
           STOPD = SDEPTH
           NXSTOP = TRIALD
           GOTO 240
      END IF
```

If a deco ceiling will not be violated, the profile is then allowed to ascend to the trial depth (TRIALD). This is done by calling the ascent/descent subroutine ASCDEC.

```
      CALL ASCDEC (SDEPTH, TRIALD, RATE)
```

The gas loadings have now been updated as a result of the ascent.

Next, we call the subroutine MVCALC (for M-value calculation) to determine the maximum Percent M-value (PMVMAX) across all 16 compartments UPON ARRIVAL AT THE TRIAL DEPTH. This represents a worst-case condition in terms of proximity to an M-value. This procedure is done as a second safety check in the process. The maximum Percent M-value is written to the output file for each segment during the ascent so that the user (the diver) will easily spot any discrepancy. In addition, this is valuable information for divers who wish to tailor their profiles according to individual disposition and physiology (i.e. set personal deco limits on a Percent M-value basis). The Percent M-value parameter is also used to gauge the effect of various Gradient Factors when using the Gradient Factor Method for conservatism. The subroutine MVCALC is explained separately after this explanation of the main program.

```
      CALL MVCALC (PMVMAX)
```

We now write the pertinent data for the ascent segment to the output file and check to make sure that we have not arrived at the surface. If we have arrived at the surface (depth pressure = 0) then we jump to the end of the program.

```
      IF (TRIALD .EQ. 0.0) THEN
           WRITE (8,850) SEGNUM, SGTIME, RTIME, MIXNUM, TRIALD, RATE,
     *     PMVMAX, FACTOR
           GOTO 300
      ELSE
           WRITE (8,851) SEGNUM, SGTIME, RTIME, MIXNUM, TRIALD, RATE,
     *     PMVMAX
      END IF
```

Next we check to see if this trial depth is a depth where there is a change in the decompression parameters. If so, they are read-in.

```
      IF (CHANGE .EQ. TRIALD) THEN
           READ (7,*) MIXNUM, RATE, STEPSZ
           READ (7,*) CHANGE
      END IF
```

Now we must determine the next trial depth for ascent. The current trial depth becomes the new starting depth and the next trial depth will be the new starting depth minus one standard stop depth. We then go back to Line 230 to start the trial depth ascent process all over again.

```
      SDEPTH = TRIALD
     TRIALD = SDEPTH - 10.0  [fsw] or
```

```
      TRIALD = SDEPTH - 3.0   [msw]
      GOTO 230
```

The Decompression Stop Process:

The next part of the decompression sequence involves the process of completing
decompression stops. When the trial depth ascent process encounters a deco
ceiling, the program goes to Line 240. In this case, the first few lines
pertain to the use of the Gradient Factor Method for conservatism.

Optional: If the first stop is greater than zero (i.e. below the surface)
then the slope for the change in Gradient Factor with change in stop depth is
calculated [see my article on "Deep Stops"].

```
240   IF (STOPD .GT. 0.0) THEN
           FCTRSL = (FCTRHI - FCTRLO)/(0.0 - STOPD)
      END IF
```

Optional: the Gradient Factor used to determine the current deco stop is
assigned to STOPGF and the Gradient Factor for the next stop is
calculated.

```
250   STOPGF = FACTOR
      FACTOR = NXSTOP*FCTRSL + FCTRHI
```

Next, we determine how much time is required at the present deco stop in order
to safely ascend to the NEXT STOP without violating a deco ceiling. This is
the basis of the "LOOK AHEAD" method. To do this we use the subroutine DSTOP
(for deco stop). It is explained separately after this explanation of the
main program. Note that the gas loadings for the deco stop (constant depth
profile) will be updated by the subroutine DSTOP.

```
      CALL DSTOP (STOPD, NXSTOP)
```

Since the decompression tables produced by this program will be based on run
times (and NOT on stop times), a deco run time variable is employed so that
the stop times (as well as the run times) written to the output file will be
in whole minutes. Each stop time will include the ascent time from the last
stop to the present stop. The only exception to this is the stop time for the
first stop which is rounded up to the nearest whole minute from the segment
time.

```
      IF (DECORT .EQ. 0.0) THEN
           STOPT = ANINT(SGTIME + 0.5)
      ELSE
           STOPT = RTIME - DECORT
      END IF
```

Next, we write the pertinent data for the deco stop to the output file.

```
      WRITE (8,852) SEGNUM, SGTIME, RTIME, MIXNUM, INT(STOPD),
     *           INT(STOPT), INT(RTIME), STOPGF
```

Once the program has returned from the deco stop subroutine, we are now clear to ascend to the next stop. The current stop depth becomes the new starting depth and the current next stop depth becomes the new stop depth. The deco run time is set to the current run time.

```
      SDEPTH = STOPD
      STOPD = NXSTOP
      DECORT = RTIME
```

The profile then ascends to the new stop depth. This is done by calling the ascent/descent subroutine ASCDEC.

```
      CALL ASCDEC (SDEPTH, STOPD, RATE)
```

Next, the absolute ambient pressure in the common block is set for the new stop depth.

```
     PAMB = STOPD + 33.0  [fsw] or
      PAMB = STOPD + 10.0  [msw]
```

Now we call the subroutine MVCALC (for M-value calculation) to determine the maximum Percent M-value (PMVMAX) across all 16 compartments UPON ARRIVAL AT THE STOP DEPTH. Again, this procedure is done as a second safety check in the process and it allows divers to evaluate their profiles on a Percent M-value basis.

```
      CALL MVCALC (PMVMAX)
```

We now write the pertinent data for the ascent segment to the output file. If we have arrived at the surface (depth pressure = 0) then we include the surfacing Gradient Factor in the printout.

```
      IF (STOPD .EQ. 0.0) THEN
           WRITE (8,850) SEGNUM, SGTIME, RTIME, MIXNUM, STOPD, RATE,
     *     PMVMAX, FACTOR
      ELSE
           WRITE (8,851) SEGNUM, SGTIME, RTIME, MIXNUM, STOPD, RATE,
     *     PMVMAX
      END IF
```

Next, we check to make sure that we have not arrived at the surface. If we have arrived at the surface (depth pressure = 0) then we jump to the end of the program.

```
      IF (STOPD .EQ. 0.0) THEN
           GOTO 300
      END IF
```

Then we check to see if this stop depth is a depth where there is a change in
the decompression parameters. If so, they are read-in.

```
      IF (CHANGE .EQ. STOPD) THEN
           READ (7,*) MIXNUM, RATE, STEPSZ
           READ (7,*) CHANGE
      END IF
```

Finally, we check to make sure that present step size setting will not make
the next stop shallower than the surface and then we set the next stop depth
based on the step size. We then go back to Line 250 to start the
decompression stop process all over again.

```
      IF (STOPD - STEPSZ .LT. 0.0) THEN
           NXSTOP = 0.0
      ELSE
           NXSTOP = STOPD - STEPSZ
      END IF
      GOTO 250
```

The last several lines of the main program include writing a message to the
screen indicating that the program calculations are complete, closing the
input and output files, and numerous FORMAT statements for all read and write
statements.

```
300   CONTINUE
      WRITE (*,800)
      WRITE (*,860)
      WRITE (*,861)
      WRITE (*,800)
350   CONTINUE
      CLOSE (UNIT = 7, STATUS = 'KEEP')
      CLOSE (UNIT = 8, STATUS = 'KEEP')
800   FORMAT (' ')
[not all FORMAT statements are shown here - see full program code]
861   FORMAT ('0Output data is located in the file DECOCALC.OUT')
      END
```

[END OF MAIN PROGRAM DESCRIPTION]

DESCRIPTION OF SUBROUTINES FOR GAS LOADING CALCULATIONS

First is the subroutine for ascent/descent segments (Subroutine ASCDEC) which
uses the Schreiner equation for exponential gas loading [see separate
explanation about Gas Loading Calculations and Schreiner Equation].

```
      SUBROUTINE ASCDEC (SDEPTH, FDEPTH, RATE)
```

As in the main program, variables and arrays are declared.

```
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O
      REAL FDEPTH, SDEPTH, PIHEO, PIN2O, RATE, RTIME, SGTIME, TEMPRT
      REAL HERATE, N2RATE, SPAMB, PAMB, FPAMB
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
```

Variables which are common to this subroutine and the main program are accessed and updated in the lettered COMMON BLOCKS. In this case, we are most interested in the updates to COMMON BLOCK /C/ which contains the array variables PHE and PN2. This subroutine will update the gas loadings in each compartment as a result of the ascent or descent segment.

```
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
```

The common variables of segment time (SGTIME), run time (RTIME), and segment number (SEGNUM) are updated.

```
      SGTIME = (FDEPTH - SDEPTH)/RATE
      TEMPRT = RTIME
      RTIME = TEMPRT + SGTIME
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
```

Next, the depth pressures used as input data must be converted to absolute pressures for the calculations. This is done by adding the surface barometric (atmospheric) pressure to the depth pressure. For a dive at sea level, the surface barometric pressure is 33 fsw (American System) or 10 msw (European System). The final ambient pressure (FPAMB) of this ascent or descent segment is the final depth pressure (FDEPTH) plus the surface barometric pressure. The starting ambient pressure (SPAMB) of this ascent or descent segment is the starting depth pressure (SDEPTH) plus the surface barometric pressure.

```
      FPAMB = FDEPTH + 33.0  [fsw] or
      FPAMB = FDEPTH + 10.0  [msw]

      SPAMB = SDEPTH + 33.0  [fsw] or
      SPAMB = SDEPTH + 10.0  [msw]
```

The absolute ambient pressure in the common block is updated to the final ambient pressure.

```
     PAMB = FPAMB
```

The initial inspired partial pressure of helium (PIHEO) is the starting ambient pressure (absolute) minus the water vapor pressure (absolute) times the fraction of helium in the present gas mix. The initial inspired partial

pressure of nitrogen (PIN2O) is the starting ambient pressure (absolute) minus
the water vapor pressure (absolute) times the fraction of nitrogen in the present gas mix.

```
      PIHEO = (SPAMB - PH2O)*FHE(MIXNUM)
      PIN2O = (SPAMB - PH2O)*FN2(MIXNUM)
```

The rate of change in inspired helium pressure (HERATE) is simply the ascent
or descent rate times the fraction of helium in the present gas mix. The rate
of change in inspired nitrogen pressure (N2RATE) is simply the ascent or descent rate times the fraction of nitrogen in the present gas mix. VERY IMPORTANT: Keep in mind that all rates (RATE, HERATE, N2RATE) must be positive (+) for descent segments and negative (-) for ascent segments.

```
      HERATE = RATE*FHE(MIXNUM)
      N2RATE = RATE*FN2(MIXNUM)
```

Finally, a program loop is used to update the gas loadings across all 16 compartments in the array variables. The initial partial pressure of helium
(PHEO) is the old (present) partial pressure of helium (PHE) which is about to
get updated. The initial partial pressure of nitrogen (PN2O) is the old (present) partial pressure of nitrogen (PN2) which is about to get updated.
The new (updated) partial pressures for helium and nitrogen (PHE and PN2), i.e. the gas loadings, are computed using the Schreiner equation:

```
      DO 430 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PIHEO + HERATE*(SGTIME - 1.0/KHE(I)) -
     *         (PIHEO - PHEO(I) - HERATE/KHE(I))*EXP (-KHE(I)*SGTIME)
      PN2(I) = PIN2O + N2RATE*(SGTIME - 1.0/KN2(I)) -
     *         (PIN2O - PN2O(I) - N2RATE/KN2(I))*EXP (-KN2(I)*SGTIME)
430   CONTINUE
      RETURN
      END   [return to main program]
```

Second is the subroutine for constant depth segments (Subroutine CDEPTH) which
uses the Haldane equation for exponential gas loading [see separate explanation about Gas Loading Calculations and Schreiner Equation].

```
      SUBROUTINE CDEPTH (DEPTH, SRTIME)
```

As in the main program, variables and arrays are declared.

```
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O, SRTIME
      REAL DEPTH, FPAMB, PAMB, PIHE, PIN2, RTIME, SGTIME, TEMPRT
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
```

Variables which are common to this subroutine and the main program are accessed and updated in the lettered COMMON BLOCKS. In this case, we are most

interested in the updates to COMMON BLOCK /C/ which contains the array variables PHE and PN2. This subroutine will update the gas loadings in each
compartment as a result of the constant depth segment.

```
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
```

The common variables of segment time (SGTIME), run time (RTIME), and segment
number (SEGNUM) are updated.

```
      SGTIME = SRTIME - RTIME
      TEMPRT = SRTIME
      RTIME = TEMPRT
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
```

Next, the depth pressure used as input data must be converted to absolute pressure for the calculations. This is done by adding the surface barometric
(atmospheric) pressure to the depth pressure. For a dive at sea level, the
surface barometric pressure is 33 fsw (American System) or 10 msw (European
System). The ambient pressure (PAMB) for this constant depth segment is the
depth pressure (DEPTH) plus the surface barometric pressure.

```
      PAMB = DEPTH + 33.0  [fsw] or
      PAMB = DEPTH + 10.0  [msw]
```

The inspired partial pressure of helium (PIHE) is the ambient pressure (absolute) minus the water vapor pressure (absolute) times the fraction of helium in the present gas mix. The inspired partial pressure of nitrogen (PIN2) is the ambient pressure (absolute) minus the water vapor pressure (absolute) times the fraction of nitrogen in the present gas mix.

```
      PIHE = (PAMB - PH2O)*FHE(MIXNUM)
      PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
```

Finally, a program loop is used to update the gas loadings across all 16 compartments in the array variables. The initial partial pressure of helium
(PHEO) is the old (present) partial pressure of helium (PHE) which is about to
get updated. The initial partial pressure of nitrogen (PN2O) is the old (present) partial pressure of nitrogen (PN2) which is about to get updated.
The new (updated)partial pressures for helium and nitrogen (PHE and PN2), i.e.
the gas loadings, are computed using the Haldane equation:

```
      DO 520 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PHEO(I) + (PIHE - PHEO(I))*
     *         (1.0 - EXP (-KHE(I)*SGTIME))
      PN2(I) = PN2O(I) + (PIN2 - PN2O(I))*
```

```
     *         (1.0 - EXP (-KN2(I)*SGTIME))
520   CONTINUE
      RETURN
      END   [return to main program]
```

DESCRIPTION OF SUBROUTINE FOR SAFE ASCENT CALCULATION (DECO CEILING)

This subroutine is used to determine the deco ceiling (CEILNG) at any point during the dive or decompression profile.

```
      SUBROUTINE SAFASC (CEILNG)
```

As in the main program, variables and arrays are declared.

```
      REAL AHE, BHE, AN2, BN2, AHEN2, BHEN2
      REAL PHE, PN2, PHEN2, PAMBT, SAFEAD, FACTOR, CEILNG
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      DIMENSION AHEN2(16), BHEN2(16), PHE(16), PN2(16)
      DIMENSION PHEN2(16), PAMBT(16), SAFEAD(16)
```

Variables which are common to this subroutine, the main program, and a BLOCK DATA subprogram are identified in the lettered COMMON BLOCKS. In this case, we will NOT be updating any variables in the COMMON BLOCKS, only using their values to perform the calculations. The COMMON BLOCK /C/ contains the PRESENT values for the gas loadings, PHE and PN2. The COMMON BLOCK /E/ contains the values for the Buhlmann Coefficients "a" and "b" for helium (AHE and BHE) and nitrogen (AN2 and BN2) which are used in linear equations to calculate the tolerated ambient pressures. The Buhlmann Coefficients in COMMON BLOCK /E/ are initialized in a BLOCK DATA subprogram (which appears as a separate subprogram after the end of the main program). The BLOCK DATA subprogram for the Buhlmann Coefficients is explained separately in this presentation. The COMMON BLOCK /F/ contains the PRESENT value for the Gradient Factor (FACTOR) which is used to calculate with conservatism under the Gradient Factor Method (optional).

```
      COMMON /C/ PHE, PN2, /E/ AHE, BHE, AN2, BN2, /F/ FACTOR
```

Next, the variable for the deco ceiling (CEILNG) is initialized to zero.

```
      CEILNG = 0.0
```

Finally, a program loop is used to do a series of calculations which produces the deco ceiling.

THE ESSENCE OF THIS PROCESS IS TO COMPARE THE PRESENT GAS LOADINGS FOR EACH

COMPARTMENT AGAINST THE M-VALUES FOR EACH COMPARTMENT TO DETERMINE THE SAFE
ASCENT DEPTH FOR EACH COMPARTMENT. THIS IS ACCOMPLISHED DIRECTLY BY REARRANGING THE M-VALUE LINEAR EQUATION AND SOLVING FOR TOLERATED AMBIENT PRESSURE, WHICH IS THEN CONVERTED TO A SAFE ASCENT DEPTH. THE MAXIMUM SAFE
ASCENT DEPTH ACROSS ALL COMPARTMENTS IS THEN THE DECO CEILING FOR THE PROFILE.

First, this requires the summation of the PRESENT values for the partial pressures of helium (PHE) and nitrogen (PN2) to give the total partial pressure of inert gas (PHEN2) present in each compartment. Next, in accordance with the Buhlmann algorithm, "intermediate" values (AHEN2 and BHEN2) for Coefficient "a" (intercept) and Coefficient "b" (slope) are calculated to account for both helium and nitrogen in the compartments. This
results in an adjustment between the separate M-values (and thus tolerated ambient pressures) for helium and nitrogen based on the proportions of these
inert gases present in each compartment. These same calculations apply if only one inert gas is present in the compartments (i.e. you don't need another
subroutine since the "intermediate" Coefficients will default to the values
for either helium or nitrogen). The tolerated ambient pressure, absolute, (PAMBT) is then calculated for each compartment based on the PRESENT total inert gas loading (PHEN2) and the PRESENT "intermediate" Coefficients (AHEN2
and BHEN2). Note: the M-value and the tolerated ambient pressure are related
by different forms of the same linear equation where the M-value is expressed
in the y = mx + b form and the tolerated ambient pressure is expressed in the
x = (y - b)/m form. The safe ascent depth (SAFEAD) for each compartment is
calculated by converting the tolerated ambient pressure, absolute, (PAMBT) back to a depth pressure. This is the tolerated ambient pressure, absolute,
minus the surface barometric pressure, absolute. Keep in mind which system of
pressure units you are working in, American (fsw)or European (msw). The final
step is to find the maximum safe ascent depth or deco ceiling (CEILNG) ACROSS
ALL COMPARTMENTS. This is the final number which determines the actual safe
deco ceiling such that an M-value is not exceeded in ANY compartment BASED ON
THE PRESENT GAS LOADING. The following code sequence calculates with the straight (unmodified) M-values only (NO conservatism):

```
      DO 600 I = 1,16
      PHEN2(I) = PHE(I) + PN2(I)
      AHEN2(I) = (PHE(I)*AHE(I) + PN2(I)*AN2(I))/PHEN2(I)
      BHEN2(I) = (PHE(I)*BHE(I) + PN2(I)*BN2(I))/PHEN2(I)
      PAMBT(I) = (PHEN2(I) - AHEN2(I))*BHEN2(I)
      SAFEAD(I) = PAMBT(I) - 33.0  [fsw] or
      SAFEAD(I) = PAMBT(I) - 10.0  [msw]
      CEILNG = MAX(CEILNG, SAFEAD(I))
```

```
600   CONTINUE
      RETURN
      END   [return to main program]
```

Optional: the following is the same sequence using the Gradient Factor Method
for conservatism [see my article on "Deep Stops"]:

```
      DO 600 I = 1,16
      PHEN2(I) = PHE(I) + PN2(I)
      AHEN2(I) = (PHE(I)*AHE(I) + PN2(I)*AN2(I))/PHEN2(I)
      BHEN2(I) = (PHE(I)*BHE(I) + PN2(I)*BN2(I))/PHEN2(I)
      PAMBT(I) = (PHEN2(I) - AHEN2(I)*FACTOR)/(FACTOR/BHEN2(I) -
     *            FACTOR + 1.0)
      SAFEAD(I) = PAMBT(I) - 33.0  [fsw] or
      SAFEAD(I) = PAMBT(I) - 10.0  [msw]
      CEILNG = MAX(CEILNG, SAFEAD(I))
600   CONTINUE
      RETURN
      END   [return to main program]
```

DESCRIPTION OF SUBROUTINE FOR DECOMPRESSION STOPS

```
      SUBROUTINE DSTOP (STOPD, STEPSZ)
```

As in the main program, variables and arrays are declared.

```
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O
      REAL STOPD, PAMB, PIHE, PIN2, RTIME, SGTIME, TEMPRT
      REAL CEILNG, NXSTOP, ROUND, TEMPST, COUNT, O2DECO
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
```

Variables which are common to this subroutine, the safe ascent subroutine, and
the main program are accessed and/or updated in the lettered COMMON BLOCKS.
In this case, we are most interested in the updates to COMMON BLOCK /C/ which
contains the array variables PHE and PN2. This subroutine will update the gas
loadings in each compartment as a result of the decompression stop.

```
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB, /J/ O2DECO
```

The next sequence in this subroutine is an area where the PHILOSOPHY of the
decompression programmer comes into play. The programmer must decide whether
the decompression tables will be based on run times or stop times. My
experience with technical diving suggests that it is better (easier for the
diver) to base everything on run times. This means that all deco stops are
rounded up to the nearest whole minute in run time and all ascents between
stops are contained in the "stop time." The advantage for divers is that they

only need to "track" the run time to know when to leave a stop. There is no in-water mathematics involved such as adding a stop time to the run time on a dive watch to know when a stop is finished. Also, in-water bottom timers and stop watches are usually geared for run time as well. Accordingly, this subroutine will start out by rounding up to the next whole minute in run time.
The difference will be the initial segment time (stop time) for this deco stop. After one pass through the gas loading calculations, additional stop time as required will be added in even one minute increments (so that the run times will always be in whole numbers). The common variables of segment time (SGTIME), run time (RTIME), and segment number (SEGNUM) are updated.

```
TEMPRT = RTIME
ROUND = ANINT(TEMPRT + 0.5)
SGTIME = ROUND - RTIME
RTIME = ROUND
TEMPST = SGTIME
TEMPSG = SEGNUM
SEGNUM = TEMPSG + 1
```

Next, the stop depth pressure must be converted to absolute pressure for the calculations. This is done by adding the surface barometric (atmospheric) pressure to the stop depth pressure. For a dive at sea level, the surface barometric pressure is 33 fsw (American System) or 10 msw (European System).
The ambient pressure (PAMB) for this constant depth segment is the depth pressure (DEPTH) plus the surface barometric pressure.

```
PAMB = STOPD + 33.0  [fsw] or
PAMB = STOPD + 10.0  [msw]
```

The inspired partial pressure of helium (PIHE) is the ambient pressure (absolute) minus the water vapor pressure (absolute) times the fraction of helium in the present gas mix. The inspired partial pressure of nitrogen (PIN2) is the ambient pressure (absolute) minus the water vapor pressure (absolute) times the fraction of nitrogen in the present gas mix.

```
PIHE = (PAMB - PH2O)*FHE(MIXNUM)
PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
```

Optional: some decompression modelers do not believe that high oxygen mixes (80%-100%) provide full benefit from an off-gassing standpoint. An oxygen deco factor (O2DECO) may be employed which acts to increase the fraction of nitrogen used in the gas loading calculations.

```
IF ((FN2(MIXNUM) .GE. 0.0) .AND. (FN2(MIXNUM) .LE. 0.2)) THEN
     PIN2 = (PAMB - PH2O)*(1.0 - O2DECO + O2DECO*FN2(MIXNUM))
END IF
```

In the final sequence, a program loop is used to update the gas loadings across all 16 compartments in the array variables. The initial partial

pressure of helium (PHEO) is the old (present) partial pressure of helium (PHE) which is about to get updated. The initial partial pressure of nitrogen (PN2O) is the old (present) partial pressure of nitrogen (PN2) which is about to get updated. The new (updated) partial pressures for helium and nitrogen (PHE and PN2), i.e. the gas loadings, are computed using the Haldane equation (for a constant depth profile). After the inital pass through this loop, the subroutine SAFASC is called to determine the updated deco ceiling. If the current deco ceiling is deeper than the next stop, then an additional minute is added to the stop time and the process repeats. This GOTO loop will continue until the gas loadings are reduced enough for the deco ceiling to clear the next stop. This procedure is the "LOOK AHEAD" method for decompression calculations. The important fundamental is that the stop time at this deco stop will accumulate until the gas loadings at this stop are reduced enough to clear the M-values (or modified M-values) FOR THE NEXT STOP.

```
700   DO 720 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PHEO(I) + (PIHE - PHEO(I))*
     *         (1.0 - EXP (-KHE(I)*SGTIME))
      PN2(I) = PN2O(I) + (PIN2 - PN2O(I))*
     *         (1.0 - EXP (-KN2(I)*SGTIME))
720   CONTINUE
      CALL SAFASC (CEILNG)
      IF (CEILNG .GT. NXSTOP) THEN
           SGTIME = 1.0
           COUNT = TEMPST
           TEMPST =  COUNT + 1.0
           TEMPRT = RTIME
           RTIME = TEMPRT + 1.0
           GOTO 700
      END IF
      SGTIME = TEMPST
      RETURN
      END  [return to the main program]
```

DESCRIPTION OF SUBROUTINE FOR M-VALUE CALCULATIONS

```
      SUBROUTINE MVCALC (PMVMAX)
```

As in the main program, variables and arrays are declared.

```
      REAL AHE, BHE, AN2, BN2, AHEN2, BHEN2, PAMB
      REAL PHE, PN2, PHEN2, MVALUE, PERCMV, PMVMAX
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      DIMENSION AHEN2(16), BHEN2(16), PHE(16), PN2(16)
      DIMENSION PHEN2(16), MVALUE(16), PERCMV(16)
```

Variables which are common to this subroutine, the main program, and a BLOCK

DATA subprogram are identified in the lettered COMMON BLOCKS. In this case, we will NOT be updating any variables in the COMMON BLOCKS, only using their values to perform the calculations. The COMMON BLOCK /C/ contains the PRESENT values for the gas loadings, PHE and PN2. The COMMON BLOCK /E/ contains the values for the Buhlmann Coefficients "a" and "b" for helium (AHE and BHE) and nitrogen (AN2 and BN2) which are used in linear equations to calculate the M-values. The Buhlmann Coefficients in COMMON BLOCK /E/ are initialized in a BLOCK DATA subprogram (which appears as a separate subprogram after the end of the main program). The BLOCK DATA subprogram for the Buhlmann Coefficients is explained separately in this presentation.

```
      COMMON /C/ PHE, PN2, /D/ PAMB, /E/ AHE, BHE, AN2, BN2
```

The maximum Percent M-value variable (PMVMAX) is initialized to zero.

```
      PMVMAX = 0.0
```

Finally, a program loop is used to do a series of calculations which produces the maximum Percent M-value across all sixteen (16) compartments. First, this requires the summation of the PRESENT values for the partial pressures of helium (PHE) and nitrogen (PN2) to give the total partial pressure of inert gas (PHEN2) present in each compartment. Next, in accordance with the Buhlmann algorithm, "intermediate" values (AHEN2 and BHEN2) for Coefficient "a" (intercept) and Coefficient "b" (slope) are calculated to account for both helium and nitrogen in the compartments. This results in an adjustment between the separate M-values for helium and nitrogen based on the proportions of these inert gases present in each compartment. These same calculations apply if only one inert gas is present in the compartments (i.e. you don't need another subroutine since the "intermediate" Coefficients will default to the values for either helium or nitrogen). The M-value (in absolute pressure) is then calculated for each compartment based on the PRESENT ambient pressure (absolute) and the PRESENT "intermediate" Coefficients (AHEN2 and BHEN2). The Percent M-value (PERCMV) for each compartment is calculated by dividing the PRESENT total partial pressure of inert gas (PHEN2)in each compartment by the M-value for each compartment. The final step is to find the maximum Percent M-value (PMVMAX) ACROSS ALL COMPARTMENTS. This is the actual number which describes the proximity of a profile to the established limits (the Buhlmann

M-values in this case). Note that these calculations are referenced to the
straight (unmodified) M-values (i.e. NO conservatism factors are applicable)
because we are comparing the gas loadings against these standard values.

```
      DO 800 I = 1,16
      PHEN2(I) = PHE(I) + PN2(I)
      AHEN2(I) = (PHE(I)*AHE(I) + PN2(I)*AN2(I))/PHEN2(I)
      BHEN2(I) = (PHE(I)*BHE(I) + PN2(I)*BN2(I))/PHEN2(I)
      MVALUE(I) = PAMB/BHEN2(I) + AHEN2(I)
      PERCMV(I) = PHEN2(I)/MVALUE(I)
      PMVMAX = MAX(PMVMAX, PERCMV(I))
800   CONTINUE
      RETURN
      END
```

DESCRIPTION OF BLOCK DATA SUBPROGRAMS FOR M-VALUE COEFFICIENTS

A BLOCK DATA subprogram is used in FORTRAN to initialize the Buhlmann
Coefficients "a" and "b" for helium (AHE and BHE) and nitrogen (AN2 and
BN2).
The generic term for these are "M-value coefficients" and they are used in the
linear equations to calculate M-values (and tolerated ambient pressures). The
Buhlmann Coefficient "a" is the intercept at zero ambient pressure (absolute)
and it is expressed in the system of pressure units that you are calculating
in [fsw or msw]. Note: Buhlmann published the ZH-L16 "a" Coefficients in the
pressure units of BAR. For practical purposes in a decompression program,
these must be converted to the system of pressure units that you are working
in (American, fsw, or European, msw). To convert the "a" Coefficients from
bar to fsw, multiply by 32.5684678. To convert the "a" Coefficients from bar
to msw, multiply by 10. Coefficient "b" is the reciprocal of the slope and it
is dimensionless (i.e. it is the same for all applications). The following
BLOCK DATA subprogram initializes the Buhlmann Coefficients for Compartments
1b thru 16 for the American System of Pressure Units, feet of seawater (fsw).
These are the ZH-L16A Coefficients for helium (AHE and BHE) and the ZH-L16B
Coefficients for nitrogen (AN2 and BN2). The "a" Coefficients (AHE and AN2)
are expressed in feet of seawater, fsw, absolute:

```
      BLOCK DATA COEFFS
      REAL AHE, BHE, AN2, BN2
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      COMMON /E/ AHE, BHE, AN2, BN2
      DATA AHE(1)/52.73/,AHE(2)/45.04/,AHE(3)/38.82/,AHE(4)/34.06/,
     *     AHE(5)/30.03/,AHE(6)/26.72/,AHE(7)/23.79/,AHE(8)/21.18/,
```

```
     *     AHE(9)/19.38/,AHE(10)/18.06/,AHE(11)/17.37/,AHE(12)/16.90/,
     *     AHE(13)/16.87/,AHE(14)/16.86/,AHE(15)/16.84/,AHE(16)/16.67/
      DATA BHE(1)/0.4770/,BHE(2)/0.5747/,BHE(3)/0.6527/,BHE(4)/0.7223/,
     *     BHE(5)/0.7582/,BHE(6)/0.7957/,BHE(7)/0.8279/,BHE(8)/0.8553/,
     *     BHE(9)/0.8757/,BHE(10)/0.8903/,BHE(11)/0.8997/,
     *     BHE(12)/0.9073/,BHE(13)/0.9122/,BHE(14)/0.9171/,
     *     BHE(15)/0.9217/,BHE(16)/0.9267/
      DATA AN2(1)/38.09/,AN2(2)/32.57/,AN2(3)/28.07/,AN2(4)/24.63/,
     *     AN2(5)/21.71/,AN2(6)/18.24/,AN2(7)/16.11/,AN2(8)/14.66/,
     *     AN2(9)/13.64/,AN2(10)/12.37/,AN2(11)/11.39/,AN2(12)/10.50/,
     *     AN2(13)/9.28/,AN2(14)/8.91/,AN2(15)/8.22/,AN2(16)/7.58/
      DATA BN2(1)/0.5578/,BN2(2)/0.6514/,BN2(3)/0.7222/,BN2(4)/0.7825/,
     *     BN2(5)/0.8126/,BN2(6)/0.8434/,BN2(7)/0.8693/,BN2(8)/0.8910/,
     *     BN2(9)/0.9092/,BN2(10)/0.9222/,BN2(11)/0.9319/,
     *     BN2(12)/0.9403/,BN2(13)/0.9477/,BN2(14)/0.9544/,
     *     BN2(15)/0.9602/,BN2(16)/0.9653/
      END
```

The following BLOCK DATA subprogram initializes the Buhlmann Coefficients for
Compartments 1b thru 16 for the European System of Pressure Units, meters of
seawater (msw). These are the ZH-L16A Coefficients for helium (AHE and BHE)
and the ZH-L16B Coefficients for nitrogen (AN2 and BN2). The "a" Coefficients
(AHE and AN2) are expressed in meters of seawater, msw, absolute:

```
      BLOCK DATA COEFFS
      REAL AHE, BHE, AN2, BN2
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      COMMON /E/ AHE, BHE, AN2, BN2
      DATA AHE(1)/16.189/,AHE(2)/13.830/,AHE(3)/11.919/,AHE(4)/10.458/,
     *     AHE(5)/9.220/,AHE(6)/8.205/,AHE(7)/7.305/,AHE(8)/6.502/,
     *     AHE(9)/5.950/,AHE(10)/5.545/,AHE(11)/5.333/,
     *     AHE(12)/5.189/,AHE(13)/5.181/,AHE(14)/5.176/,
     *     AHE(15)/5.172/,AHE(16)/5.119/
      DATA BHE(1)/0.4770/,BHE(2)/0.5747/,BHE(3)/0.6527/,BHE(4)/0.7223/,
     *     BHE(5)/0.7582/,BHE(6)/0.7957/,BHE(7)/0.8279/,BHE(8)/0.8553/,
     *     BHE(9)/0.8757/,BHE(10)/0.8903/,BHE(11)/0.8997/,
     *     BHE(12)/0.9073/,BHE(13)/0.9122/,BHE(14)/0.9171/,
     *     BHE(15)/0.9217/,BHE(16)/0.9267/
      DATA AN2(1)/11.696/,AN2(2)/10.000/,AN2(3)/8.618/,AN2(4)/7.562/,
     *     AN2(5)/6.667/,AN2(6)/5.600/,AN2(7)/4.947/,AN2(8)/4.500/,
     *     AN2(9)/4.187/,AN2(10)/3.798/,AN2(11)/3.497/,
     *     AN2(12)/3.223/,AN2(13)/2.850/,AN2(14)/2.737/,
     *     AN2(15)/2.523/,AN2(16)/2.327/
      DATA BN2(1)/0.5578/,BN2(2)/0.6514/,BN2(3)/0.7222/,BN2(4)/0.7825/,
     *     BN2(5)/0.8126/,BN2(6)/0.8434/,BN2(7)/0.8693/,BN2(8)/0.8910/,
     *     BN2(9)/0.9092/,BN2(10)/0.9222/,BN2(11)/0.9319/,
     *     BN2(12)/0.9403/,BN2(13)/0.9477/,BN2(14)/0.9544/,
     *     BN2(15)/0.9602/,BN2(16)/0.9653/
      END
```

Example of complete, functional FORTRAN Decompression Program (msw units)
ready for compiling:

```
      PROGRAM DECOCALC
C     BUHLMANN 16 COMPARTMENTS, ZH-L16B M-VALUES, MSW UNITS
```

```
C
      CHARACTER COMAND*3, WORD*7, LINE1*70
      INTEGER NUMMIX, PROFIL, SEGNUM, MIXNUM
      REAL RTIME, PAMB, PH2O, FACTOR, CEILNG, STOPD, STEPSZ
      REAL FO2, FHE, FN2, FSUM, CKSUM, CHANGE, PMVMAX, SGTIME
      REAL PHE, PN2, HALFTH, HALFTN, KHE, KN2, FCTRHI, FCTRLO, FCTRSL
      REAL DEPTH, FDEPTH, SDEPTH, RATE, SRTIME, DECORT, STOPT
      REAL O2DECO, TEMP1, TEMP2, NXSTOP, STOPGF, TRIALD
      DIMENSION FO2(10), FHE(10), FN2(10), PHE(16), PN2(16)
      DIMENSION HALFTH(16), HALFTN(16), KHE(16), KN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB, /F/ FACTOR, /J/ O2DECO
      DATA HALFTH(1)/1.88/,HALFTH(2)/3.02/,HALFTH(3)/4.72/,
     *     HALFTH(4)/6.99/,HALFTH(5)/10.21/,HALFTH(6)/14.48/,
     *     HALFTH(7)/20.53/,HALFTH(8)/29.11/,HALFTH(9)/41.20/,
     *     HALFTH(10)/55.19/,HALFTH(11)/70.69/,HALFTH(12)/90.34/,
     *     HALFTH(13)/115.29/,HALFTH(14)/147.42/,HALFTH(15)/188.24/,
     *     HALFTH(16)/240.03/
      DATA HALFTN(1)/5.0/,HALFTN(2)/8.0/,HALFTN(3)/12.5/,
     *     HALFTN(4)/18.5/,HALFTN(5)/27.0/,HALFTN(6)/38.3/,
     *     HALFTN(7)/54.3/,HALFTN(8)/77.0/,HALFTN(9)/109.0/,
     *     HALFTN(10)/146.0/,HALFTN(11)/187.0/,HALFTN(12)/239.0/,
     *     HALFTN(13)/305.0/,HALFTN(14)/390.0/,HALFTN(15)/498.0/,
     *     HALFTN(16)/635.0/
      PH2O = 0.567
      RTIME = 0.0
      SEGNUM = 0
      COMAND = 'CLS'
      DO 10 I = 1,16
           KHE(I) = ALOG(2.0)/HALFTH(I)
           KN2(I) = ALOG(2.0)/HALFTN(I)
           PHE(I) = 0.00
           PN2(I) = 7.452
10    CONTINUE
      CALL SYSTEMQQ (COMAND)
      PRINT *,' '
      PRINT *,'PROGRAM DECOCALC'
      PRINT *,' '
      OPEN (UNIT = 7, FILE = 'DECOCALC.IN', STATUS = 'UNKNOWN',
     *         ACCESS = 'SEQUENTIAL', FORM = 'FORMATTED')
      OPEN (UNIT = 8, FILE = 'DECOCALC.OUT', STATUS = 'UNKNOWN',
     *         ACCESS = 'SEQUENTIAL', FORM = 'FORMATTED')
      READ (7,801) LINE1
      WRITE (8,802)
      WRITE (8,800)
      WRITE (8,803) LINE1
      WRITE (8,800)
      READ (7,*) NUMMIX
      DO 45 I = 1, NUMMIX
           READ (7,*) FO2(I), FHE(I), FN2(I)
           FSUM = FO2(I) + FHE(I) + FN2(I)
           CKSUM = FSUM
           IF (CKSUM .NE. 1.0) THEN
               CALL SYSTEMQQ (COMAND)
               PRINT *,' '
               PRINT *,'ERROR IN INPUT FILE (GASMIX DATA) - PROGRAM TERM
     *INATED'
               PRINT *,' '
               GOTO 350
           END IF
45    CONTINUE
      WRITE (8,810)
      DO 55 J = 1, NUMMIX
          WRITE (8,811) J, FO2(J), FHE(J), FN2(J)
55    CONTINUE
      READ (7,*) O2DECO
      WRITE (8,800)
      WRITE (8,812) O2DECO
      WRITE (8,800)
      WRITE (8,820)
      WRITE (8,800)
      WRITE (8,821)
      WRITE (8,822)
      WRITE (8,823)
      WRITE (8,824)
```

```
100   CONTINUE
      READ (7,*) PROFIL
      IF (PROFIL .EQ. 1) THEN
           READ (7,*) SDEPTH, FDEPTH, RATE, MIXNUM
           CALL ASCDEC (SDEPTH, FDEPTH, RATE)
           IF (FDEPTH .GT. SDEPTH) THEN
               WORD = 'Descent'
           ELSE IF (SDEPTH .GT. FDEPTH) THEN
               WORD = 'Ascent '
           ELSE
               WORD = 'ERROR'
           END IF
           WRITE (8,830) SEGNUM, SGTIME, RTIME, MIXNUM, WORD, SDEPTH,
     *     FDEPTH, RATE
      ELSE IF (PROFIL .EQ. 2) THEN
           READ (7,*) DEPTH, SRTIME, MIXNUM
           CALL CDEPTH (DEPTH, SRTIME)
           WRITE (8,831) SEGNUM, SGTIME, RTIME, MIXNUM, DEPTH
      ELSE IF (PROFIL .EQ. 99) THEN
           GOTO 200
      ELSE
           CALL SYSTEMQQ (COMAND)
           PRINT *,' '
           PRINT *,'ERROR IN INPUT FILE (PROFILE CODE) - PROGRAM TERMINA
     *TED'
           PRINT *,' '
           GOTO 350
      END IF
      GOTO 100
200   CONTINUE
      READ (7,*) SDEPTH
      READ (7,*) MIXNUM, RATE, STEPSZ, FCTRHI, FCTRLO
      READ (7,*) CHANGE
      WRITE (8,800)
      WRITE (8,840)
      WRITE (8,800)
      WRITE (8,841)
      WRITE (8,842)
      WRITE (8,843)
      WRITE (8,844)
      DECORT = 0.0
      FACTOR = FCTRLO
      TEMP1 = (SDEPTH/3.0) - 0.5
      TRIALD = AINT(TEMP1) * 3.0
      TEMP2 = TRIALD
      IF (TEMP2 .LE. 0.0) THEN
           TRIALD = 0.0
      END IF
230   CALL SAFASC (CEILNG)
      IF (CEILNG .GT. TRIALD) THEN
           STOPD = SDEPTH
           NXSTOP = TRIALD
           GOTO 240
      END IF
      CALL ASCDEC (SDEPTH, TRIALD, RATE)
      CALL MVCALC (PMVMAX)
      IF (TRIALD .EQ. 0.0) THEN
           WRITE (8,850) SEGNUM, SGTIME, RTIME, MIXNUM, TRIALD, RATE,
     *     PMVMAX, FACTOR
           GOTO 300
      ELSE
           WRITE (8,851) SEGNUM, SGTIME, RTIME, MIXNUM, TRIALD, RATE,
     *     PMVMAX
      END IF
      IF (CHANGE .EQ. TRIALD) THEN
           READ (7,*) MIXNUM, RATE, STEPSZ
           READ (7,*) CHANGE
      END IF
      SDEPTH = TRIALD
      TRIALD = SDEPTH - 3.0
      GOTO 230
240   IF (STOPD .GT. 0.0) THEN
           FCTRSL = (FCTRHI - FCTRLO)/(0.0 - STOPD)
      END IF
250   STOPGF = FACTOR
```

```
      FACTOR = NXSTOP*FCTRSL + FCTRHI
      CALL DSTOP (STOPD, NXSTOP)
      IF (DECORT .EQ. 0.0) THEN
           STOPT = ANINT(SGTIME + 0.5)
      ELSE
           STOPT = RTIME - DECORT
      END IF
      WRITE (8,852) SEGNUM, SGTIME, RTIME, MIXNUM, INT(STOPD),
     *           INT(STOPT), INT(RTIME), STOPGF
      SDEPTH = STOPD
      STOPD = NXSTOP
      DECORT = RTIME
      CALL ASCDEC (SDEPTH, STOPD, RATE)
      PAMB = STOPD + 10.0
      CALL MVCALC (PMVMAX)
      IF (STOPD .EQ. 0.0) THEN
           WRITE (8,850) SEGNUM, SGTIME, RTIME, MIXNUM, STOPD, RATE,
     *     PMVMAX, FACTOR
      ELSE
           WRITE (8,851) SEGNUM, SGTIME, RTIME, MIXNUM, STOPD, RATE,
     *     PMVMAX
      END IF
      IF (STOPD .EQ. 0.0) THEN
           GOTO 300
      END IF
      IF (CHANGE .EQ. STOPD) THEN
           READ (7,*) MIXNUM, RATE, STEPSZ
           READ (7,*) CHANGE
      END IF
      IF (STOPD - STEPSZ .LT. 0.0) THEN
           NXSTOP = 0.0
      ELSE
           NXSTOP = STOPD - STEPSZ
      END IF
      GOTO 250
300   CONTINUE
      WRITE (*,800)
      WRITE (*,860)
      WRITE (*,861)
      WRITE (*,800)
350   CONTINUE
      CLOSE (UNIT = 7, STATUS = 'KEEP')
      CLOSE (UNIT = 8, STATUS = 'KEEP')
800   FORMAT (' ')
801   FORMAT (A70)
802   FORMAT (26X,'DECOMPRESSION CALCULATION PROGRAM')
803   FORMAT ('Description:',4X,A70)
810   FORMAT ('Gasmix Summary:',24X,'FO2',4X,'FHe',4X,'FN2')
811   FORMAT (26X,'Gasmix #',I2,2X,F5.3,2X,F5.3,2X,F5.3)
812   FORMAT ('O2 Deco Factor:  80-100% Nitrox or O2 mixes calculated',
     *        1X,'at',2P,F5.0,'% of actual O2 fraction')
820   FORMAT (36X,'DIVE PROFILE')
821   FORMAT ('Seg-',2X,'Segm.',2X,'Run',3X,'|',1X,'Gasmix',1X,'|',1X,
     *       'Ascent',4X,'From',5X,'To',6X,'Rate',4X,'|',1X,'Constant')
822   FORMAT ('ment',2X,'Time',3X,'Time',2X,'|',2X,'Used',2X,'|',3X,
     *        'or',5X,'Depth',3X,'Depth',4X,'+Dn/-Up',2X,'|',2X,'Depth')
823   FORMAT (2X,'#',3X,'(min)',2X,'(min)',1X,'|',4X,'#',3X,'|',1X,
     *        'Descent',2X,'(mswg)',2X,'(mswg)',2X,'(msw/min)',1X,
     *        '|',2X,'(mswg)')
824   FORMAT ('-----',1X,'-----',2X,'-----',1X,'|',1X,'------',1X,'|',
     *        1X,'-------',2X,'------',2X,'------',2X,'---------',1X,
     *        '|',1X,'--------')
830   FORMAT (I3,3X,F5.1,1X,F6.1,1X,'|',3X,I2,3X,'|',1X,A7,F7.0,
     *        1X,F7.0,3X,F7.1,3X,'|')
831   FORMAT (I3,3X,F5.1,1X,F6.1,1X,'|',3X,I2,3X,'|',36X,'|',F7.0)
840   FORMAT (31X,'DECOMPRESSION PROFILE')
841   FORMAT ('Seg-',2X,'Segm.',2X,'Run',3X,'|',1X,'Gasmix',1X,'|',1X,
     *        'Ascent',3X,'Ascent',3X,'Max',3X,'|',2X,'DECO',3X,'STOP',
     *        3X,'RUN',5X,'Gradient')
842   FORMAT ('ment',2X,'Time',3X,'Time',2X,'|',2X,'Used',2X,'|',3X,
     *        'To',6X,'Rate',4X,'%M-',3X,'|',2X,'STOP',3X,'TIME',3X,
     *         'TIME',4X,'Factor')
843   FORMAT (2X,'#',3X,'(min)',2X,'(min)',1X,'|',4X,'#',3X,'|',1X,
     *        '(mswg)',1X,'(msw/min)',1X,'Value',2X,'|',1X,'(mswg)',
     *         2X,'(min)',2X,'(min)',4X,'(GF)')
```

```
844   FORMAT ('-----',1X,'-----',2X,'-----',1X,'|',1X,'------',1X,'|',
     *        1X,'------',1X,'---------',1X,'------',1X,'|',1X,
     *        '------',2X,'-----',2X,'-----',3X,'------')
850   FORMAT (I3,3X,F5.1,1X,F6.1,1X,'|',3X,I2,3X,'|',2X,F4.0,3X,F6.1,
     *        3X,2P,F5.1,'%',1X,'|',24X,0P,F5.2)
851   FORMAT (I3,3X,F5.1,1X,F6.1,1X,'|',3X,I2,3X,'|',2X,F4.0,3X,F6.1,
     *        3X,2P,F5.1,'%',1X,'|')
852   FORMAT (I3,3X,F5.1,1X,F6.1,1X,'|',3X,I2,3X,'|',25X,'|',3X,I3,4X,
     *        I3,3X,I4,4X,F5.2)
860   FORMAT (' PROGRAM CALCULATIONS COMPLETE')
861   FORMAT ('0Output data is located in the file DECOCALC.OUT')
      END
C
C
      SUBROUTINE ASCDEC (SDEPTH, FDEPTH, RATE)
C
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O
      REAL FDEPTH, SDEPTH, PIHEO, PIN2O, RATE, RTIME, SGTIME, TEMPRT
      REAL HERATE, N2RATE, SPAMB, PAMB, FPAMB
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
      SGTIME = (FDEPTH - SDEPTH)/RATE
      TEMPRT = RTIME
      RTIME = TEMPRT + SGTIME
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
      SPAMB = SDEPTH + 10.0
      FPAMB = FDEPTH + 10.0
      PAMB = FPAMB
      PIHEO = (SPAMB - PH2O)*FHE(MIXNUM)
      PIN2O = (SPAMB - PH2O)*FN2(MIXNUM)
      HERATE = RATE*FHE(MIXNUM)
      N2RATE = RATE*FN2(MIXNUM)
      DO 430 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PIHEO + HERATE*(SGTIME - 1.0/KHE(I)) -
     *         (PIHEO - PHEO(I) - HERATE/KHE(I))*EXP (-KHE(I)*SGTIME)
      PN2(I) = PIN2O + N2RATE*(SGTIME - 1.0/KN2(I)) -
     *         (PIN2O - PN2O(I) - N2RATE/KN2(I))*EXP (-KN2(I)*SGTIME)
430   CONTINUE
      RETURN
      END
C
C
      SUBROUTINE CDEPTH (DEPTH, SRTIME)
C
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O, SRTIME
      REAL DEPTH, PAMB, PIHE, PIN2, RTIME, SGTIME, TEMPRT
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB
      SGTIME = SRTIME - RTIME
      TEMPRT = SRTIME
      RTIME = TEMPRT
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
      PAMB = DEPTH + 10.0
      PIHE = (PAMB - PH2O)*FHE(MIXNUM)
      PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
      DO 520 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PHEO(I) + (PIHE - PHEO(I))*
     *         (1.0 - EXP (-KHE(I)*SGTIME))
      PN2(I) = PN2O(I) + (PIN2 - PN2O(I))*
     *         (1.0 - EXP (-KN2(I)*SGTIME))
520   CONTINUE
      RETURN
      END
```

```
C
C
      SUBROUTINE SAFASC (CEILNG)
C
      REAL AHE, BHE, AN2, BN2, AHEN2, BHEN2
      REAL PHE, PN2, PHEN2, PAMBT, SAFEAD, FACTOR, CEILNG
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      DIMENSION AHEN2(16), BHEN2(16), PHE(16), PN2(16)
      DIMENSION PHEN2(16), PAMBT(16), SAFEAD(16)
      COMMON /C/ PHE, PN2, /E/ AHE, BHE, AN2, BN2, /F/ FACTOR
      CEILNG = 0.0
      DO 600 I = 1,16
      PHEN2(I) = PHE(I) + PN2(I)
      AHEN2(I) = (PHE(I)*AHE(I) + PN2(I)*AN2(I))/PHEN2(I)
      BHEN2(I) = (PHE(I)*BHE(I) + PN2(I)*BN2(I))/PHEN2(I)
      PAMBT(I) = (PHEN2(I) - AHEN2(I)*FACTOR)/(FACTOR/BHEN2(I) -
     *            FACTOR + 1.0)
      SAFEAD(I) = PAMBT(I) - 10.0
      CEILNG = MAX(CEILNG, SAFEAD(I))
600   CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DSTOP (STOPD, NXSTOP)
C
      INTEGER MIXNUM, TEMPSG, SEGNUM
      REAL FHE, FN2, KHE, KN2, PHEO, PN2O, PHE, PN2, PH2O
      REAL STOPD, PAMB, PIHE, PIN2, RTIME, SGTIME, TEMPRT
      REAL CEILNG, NXSTOP, ROUND, TEMPST, COUNT, O2DECO
      DIMENSION FHE (10), FN2(10), KHE(16), KN2(16)
      DIMENSION PHEO(16), PN2O(16), PHE(16), PN2(16)
      COMMON /A/ FHE, KHE, KN2, PH2O, /B/ RTIME, SEGNUM, FN2, SGTIME
      COMMON /B/ MIXNUM, /C/ PHE, PN2, /D/ PAMB, /J/ O2DECO
      TEMPRT = RTIME
      ROUND = ANINT(TEMPRT + 0.5)
      SGTIME = ROUND - RTIME
      RTIME = ROUND
      TEMPST = SGTIME
      TEMPSG = SEGNUM
      SEGNUM = TEMPSG + 1
      PAMB = STOPD + 10.0
      PIHE = (PAMB - PH2O)*FHE(MIXNUM)
      IF ((FN2(MIXNUM) .GE. 0.0) .AND. (FN2(MIXNUM) .LE. 0.2)) THEN
           PIN2 = (PAMB - PH2O)*(1.0 - O2DECO + O2DECO*FN2(MIXNUM))
      ELSE
           PIN2 = (PAMB - PH2O)*FN2(MIXNUM)
      END IF
700   DO 720 I = 1,16
      PHEO(I) = PHE(I)
      PN2O(I) = PN2(I)
      PHE(I) = PHEO(I) + (PIHE - PHEO(I))*
     *         (1.0 - EXP (-KHE(I)*SGTIME))
      PN2(I) = PN2O(I) + (PIN2 - PN2O(I))*
     *         (1.0 - EXP (-KN2(I)*SGTIME))
720   CONTINUE
      CALL SAFASC (CEILNG)
      IF (CEILNG .GT. NXSTOP) THEN
           SGTIME = 1.0
           COUNT = TEMPST
           TEMPST =  COUNT + 1.0
           TEMPRT = RTIME
           RTIME = TEMPRT + 1.0
           GOTO 700
      END IF
      SGTIME = TEMPST
      RETURN
      END
C
C
      SUBROUTINE MVCALC (PMVMAX)
C
      REAL AHE, BHE, AN2, BN2, AHEN2, BHEN2, PAMB
      REAL PHE, PN2, PHEN2, MVALUE, PERCMV, PMVMAX
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
```

```
      DIMENSION AHEN2(16), BHEN2(16), PHE(16), PN2(16)
      DIMENSION PHEN2(16), MVALUE(16), PERCMV(16)
      COMMON /C/ PHE, PN2, /D/ PAMB, /E/ AHE, BHE, AN2, BN2
      PMVMAX = 0.0
      DO 800 I = 1,16
      PHEN2(I) = PHE(I) + PN2(I)
      AHEN2(I) = (PHE(I)*AHE(I) + PN2(I)*AN2(I))/PHEN2(I)
      BHEN2(I) = (PHE(I)*BHE(I) + PN2(I)*BN2(I))/PHEN2(I)
      MVALUE(I) = PAMB/BHEN2(I) + AHEN2(I)
      PERCMV(I) = PHEN2(I)/MVALUE(I)
      PMVMAX = MAX(PMVMAX, PERCMV(I))
800   CONTINUE
      RETURN
      END
C
C
      BLOCK DATA COEFFS
      REAL AHE, BHE, AN2, BN2
      DIMENSION AHE(16), BHE(16), AN2(16), BN2(16)
      COMMON /E/ AHE, BHE, AN2, BN2
      DATA AHE(1)/16.189/,AHE(2)/13.830/,AHE(3)/11.919/,AHE(4)/10.458/,
     *     AHE(5)/9.220/,AHE(6)/8.205/,AHE(7)/7.305/,AHE(8)/6.502/,
     *     AHE(9)/5.950/,AHE(10)/5.545/,AHE(11)/5.333/,
     *     AHE(12)/5.189/,AHE(13)/5.181/,AHE(14)/5.176/,
     *     AHE(15)/5.172/,AHE(16)/5.119/
      DATA BHE(1)/0.4770/,BHE(2)/0.5747/,BHE(3)/0.6527/,BHE(4)/0.7223/,
     *     BHE(5)/0.7582/,BHE(6)/0.7957/,BHE(7)/0.8279/,BHE(8)/0.8553/,
     *     BHE(9)/0.8757/,BHE(10)/0.8903/,BHE(11)/0.8997/,
     *     BHE(12)/0.9073/,BHE(13)/0.9122/,BHE(14)/0.9171/,
     *     BHE(15)/0.9217/,BHE(16)/0.9267/
      DATA AN2(1)/11.696/,AN2(2)/10.000/,AN2(3)/8.618/,AN2(4)/7.562/,
     *     AN2(5)/6.667/,AN2(6)/5.600/,AN2(7)/4.947/,AN2(8)/4.500/,
     *     AN2(9)/4.187/,AN2(10)/3.798/,AN2(11)/3.497/,
     *     AN2(12)/3.223/,AN2(13)/2.850/,AN2(14)/2.737/,
     *     AN2(15)/2.523/,AN2(16)/2.327/
      DATA BN2(1)/0.5578/,BN2(2)/0.6514/,BN2(3)/0.7222/,BN2(4)/0.7825/,
     *     BN2(5)/0.8126/,BN2(6)/0.8434/,BN2(7)/0.8693/,BN2(8)/0.8910/,
     *     BN2(9)/0.9092/,BN2(10)/0.9222/,BN2(11)/0.9319/,
     *     BN2(12)/0.9403/,BN2(13)/0.9477/,BN2(14)/0.9544/,
     *     BN2(15)/0.9602/,BN2(16)/0.9653/
      END
```

Example of input file, DECOCALC.IN, for use with FORTRAN Decompression Program:

```
SAMPLE DIVE TO 90 METERS OF SEAWATER GAUGE (MSWG) FOR 20 MINUTES
4
.13,.50,.37
.36,.00,.64
.50,.00,.50
.80,.00,.20
1.0
1
0,90,23,1
2
90,20,1
99
90
1,-10,3,0.75,0.30
33
2,-10,3
21
3,-10,3
9
4,-10,3
0
```

Explanation of format for input file:

```
Line 1:   Description of dive
Line 2:   Number of gas mixes
Line 2a:  FO2, FHe, FN2 for gas mix #1
Line 2b:  FO2, FHe, FN2 for next gas mix (#2)
Line 2c:  FO2, FHe, FN2 for next gas mix (#3)
```

Line 2d: FO2, FHe, FN2 for next gas mix (#4)
Line 3: Oxygen deco factor (usually between 0.8 and 1.0)
Line 4: Profile code for first dive segment (1 = ascent/descent)
Line 4a: Start depth, final depth, rate of ascent/descent, gas mix number
Line 5: Profile code for next dive segment (2 = constant depth)
Line 5a: Depth, run time at end of segment, gas mix number
Line 6: Profile code to start decompression sequence (= 99)
Line 7: Starting depth for decompression sequence
Line 8: Gas mix number, ascent rate, step size, Hi Gradient Factor, Lo GF
Line 9: Depth of next change in deco parameters
Line 10: Gas mix number, ascent rate, step size
Line 11: Depth of next change in deco parameters
Line 12: Gas mix number, ascent rate, step size
Line 13: Depth of next change in deco parameters
Line 14: Gas mix number, ascent rate, step size
Line 15: Depth of next change in deco parameters (or zero for surface)

Example of output file, DECOCALC.OUT, produced by FORTRAN Decompression Program:

DECOMPRESSION CALCULATION PROGRAM

Description: SAMPLE DIVE TO 90 METERS OF SEAWATER GAUGE (MSWG) FOR 20 MINUTES

Gasmix Summary:

	FO2	FHe	FN2
Gasmix # 1	.130	.500	.370
Gasmix # 2	.360	.000	.640
Gasmix # 3	.500	.000	.500
Gasmix # 4	.800	.000	.200

O2 Deco Factor: 80-100% Nitrox or O2 mixes calculated at 100.% of actual O2 fraction

DIVE PROFILE

Segment #	Segm. Time (min)	Run Time (min)	Gasmix Used #	Ascent or Descent	From Depth (mswg)	To Depth (mswg)	Rate +Dn/-Up (msw/min)	Constant Depth (mswg)
1	3.9	3.9	1	Descent	0.	90.	23.0	
2	16.1	20.0	1					90.

DECOMPRESSION PROFILE

Segment #	Segm. Time (min)	Run Time (min)	Gasmix Used #	Ascent To (mswg)	Ascent Rate (msw/min)	Max %M-Value	DECO STOP (mswg)	STOP TIME (min)	RUN TIME (min)	Gradient Factor (GF)
3	.3	20.3	1	87.	-10.0	47.0%				
4	.3	20.6	1	84.	-10.0	48.6%				
5	.3	20.9	1	81.	-10.0	50.1%				
6	.3	21.2	1	78.	-10.0	51.7%				
7	.3	21.5	1	75.	-10.0	53.4%				
8	.3	21.8	1	72.	-10.0	55.1%				
9	.3	22.1	1	69.	-10.0	56.8%				
10	.3	22.4	1	66.	-10.0	58.7%				
11	.3	22.7	1	63.	-10.0	60.6%				
12	.3	23.0	1	60.	-10.0	62.6%				
13	.3	23.3	1	57.	-10.0	64.7%				
14	.3	23.6	1	54.	-10.0	66.9%				
15	.3	23.9	1	51.	-10.0	69.5%				
16	.1	24.0	1				51	1	24	.30
17	.3	24.3	1	48.	-10.0	72.2%				
18	.7	25.0	1				48	1	25	.33
19	.3	25.3	1	45.	-10.0	74.4%				
20	.7	26.0	1				45	1	26	.35
21	.3	26.3	1	42.	-10.0	76.6%				
22	1.7	28.0	1				42	2	28	.38
23	.3	28.3	1	39.	-10.0	77.5%				
24	1.7	30.0	1				39	2	30	.41
25	.3	30.3	1	36.	-10.0	78.5%				
26	1.7	32.0	1				36	2	32	.43
27	.3	32.3	1	33.	-10.0	79.8%				
28	1.7	34.0	2				33	2	34	.46
29	.3	34.3	2	30.	-10.0	78.9%				
30	.7	35.0	2				30	1	35	.49

31	.3	35.3	2	27.	-10.0	81.6%						
32	1.7	37.0	2				27	2	37	.51		
33	.3	37.3	2	24.	-10.0	82.5%						
34	1.7	39.0	2				24	2	39	.54		
35	.3	39.3	2	21.	-10.0	85.1%						
36	2.7	42.0	3				21	3	42	.56		
37	.3	42.3	3	18.	-10.0	86.0%						
38	3.7	46.0	3				18	4	46	.59		
39	.3	46.3	3	15.	-10.0	86.8%						
40	5.7	52.0	3				15	6	52	.62		
41	.3	52.3	3	12.	-10.0	87.2%						
42	6.7	59.0	3				12	7	59	.64		
43	.3	59.3	3	9.	-10.0	88.9%						
44	9.7	69.0	4				9	10	69	.67		
45	.3	69.3	4	6.	-10.0	89.1%						
46	14.7	84.0	4				6	15	84	.70		
47	.3	84.3	4	3.	-10.0	90.3%						
48	30.7	115.0	4				3	31	115	.72		
49	.3	115.3	4	0.	-10.0	91.2%				.75		