ORNL/ENG/TM-48

Martin Marietta Energy Systems, Inc., Central Engineering Services
Technical Programs and Services

Accident Analysis Deskbook Series

LEAK: A SOURCE TERM GENERATOR FOR EVALUATING RELEASE RATES FROM LEAKING VESSELS

J. H. Clinton

Manuscript Completed—July 1993

Date of Publication—September 1994

Prepared by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
managing the

Oak Ridge K-25 Site
Oak Ridge National Laboratory
Oak Ridge Y-12 Plant
under Contract DE-AC05-84OR21400

Uranium Enrichment Organization
Including the Paducah Gaseous Diffusion Plant
and the Portsmouth Gaseous Diffusion Plant
under Contract USECHQ-93-C-0001

for the
U.S. DEPARTMENT OF ENERGY

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

TABLE OF CONTENTS

PREFACE

The preparation of safety documents for DOE-operated facilities is based on guidelines adopted by DOE. With respect to accident scenario evaluation, these guidelines require documentation of the analytical methods used in determining the consequences of the postulated accidents, including the availability of reference materials for such methods. Within Energy Systems, these requirements have resulted in the same analytical methods being described repetitively in multiple safety documents, with unreferencable supporting materials (e.g., letters or unpublished experimental data) being supplied as appendices. Cross referencing between the various safety documents would reduce the volume of materials generated, but because these documents are "living documents," subject to revision as facility usage changes and processes are upgraded; such cross referencing is not recommended. Reports in the *Accident Analysis Deskbook Series* have been prepared to document analytical techniques commonly used in the preparation of safety documents in separate, referencable documents. Included in this series are procedures for performing a variety of routine calculations encountered in preparing safety documents along with otherwise unpublished supporting information and suggested "standard text" to incorporate into the safety documents when these techniques are applied.

The primary focus of this deskbook series is on analytical methods that can be implemented readily by safety analysts having access to calculators or personal computers. Limitations of the procedures are provided to alert the analyst to conditions requiring analytical support from those having specialized training or experience with a particular class of problem. The following information is provided for each analytical method presented:

1. A descriptive title which also appears as the title of each individual report in the series.
2. A statement of purpose identifying the types of problems to which the method applies.
3. A brief description of limitations that would identify when a method would not apply or when the problem should be referred to a specialist.
4. A step-by-step analysis procedure to be followed in determining accident consequences.
5. A discussion of the experimental, theoretical, or analytical support for the analysis procedure.
6. A list of published references cited in support of the analysis method.
7. Suggested "standard text" that can be incorporated with appropriate modification into safety documents.
8. Attachments that provide supporting documentation for the analysis procedure which otherwise could not be referenced.

It is expected that the analytical techniques described in this deskbook series will be improved and extended in scope and applicability and that current uncertainties in the applicability of some techniques will be resolved. Also, other analytical methods will become routine. Therefore, reports in this series may be periodically updated and expanded, and reports on additional methods will be prepared.

Reports in the Accident Analysis Deskbook Series

W. R. Williams, *Powder Spills: Estimating the Initial Mass Airborne*, ORNL/ENG/TM-44, *in preparation.*

W. R. Williams, *Liquid Spills: Estimating the Initial Mass Airborne*, ORNL/ENG/TM-45, *in preparation.*

W. R. Williams, *Pressurized Powders: Estimating the Initial Mass Airborne*, ORNL/ENG/TM-46, *in preparation.*

W. R. Williams, *Pressurized and Flashing Liquids: Estimating the Initial Mass Airborne*, ORNL/ENG/TM-47, *in preparation.*

J. H. Clinton, *LEAK: A Source Term Generator for Evaluating Release Rates from Leaking Vessels*, ORNL/ENG/TM-48, September 1994.

J. H. Clinton, *Hydrogen Fires and Explosions*, ORNL/ENG/TM-49, *in preparation.*

W. R. Williams, *EVAPRATE: A Computer Model for Evaluating Evaporation Rates in Tanks, Rooms, and Outdoors*, ORNL/ENG/TM-50, *planned.*

ABSTRACT

An interactive computer code for estimating the rate of release of any one of several materials from a leaking tank or broken pipe leading from a tank is presented. It is generally assumed that the material in the tank is liquid. Materials included in the data base are acetonitrile, ammonia, carbon tetrachloride, chlorine, chlorine trifluoride, fluorine, hydrogen fluoride, nitric acid, nitrogen tetroxide, sodium hydroxide, sulfur hexafluoride, sulfuric acid, and uranium hexafluoride. Materials that exist only as liquid and/or vapor over expected ranges of temperature and pressure can easily be added to the data base file.

LEAK: A SOURCE TERM GENERATOR FOR EVALUATING RELEASE RATES FROM LEAKING VESSELS

J. H. Clinton

September 1994

I. PURPOSE

An interactive FORTRAN computer code called LEAK has been developed for use in determining the rate of loss and the total amount of loss of a material in the event that the vessel containing the material begins to leak. The materials included in the data base are: acetonitrile, ammonia, carbon tetrachloride, chlorine, chlorine trifluoride, fluorine, hydrogen fluoride, nitric acid, nitrogen tetroxide, sodium hydroxide, sulfur hexafluoride, sulfuric acid, and uranium hexafluoride. A major assumption of the program is that the material in a tank is liquid, with the exception of fluorine which is assumed to be in the gaseous state and uranium hexafluoride which is assumed to be in either the liquid or solid state depending upon the temperature given. If a material not in the list is in the liquid state at the temperature and pressure of the tank and does not exhibit an unusual behavior such as passing through the triple point, the material may be easily added to the data file called LEAK.DAT.

The scenario for the accident(s) upon which the program is based is that a tank is punctured or a pipe breaks off at the tank or at a point in the pipe some distance and elevation from the tank. The question then addressed is the rate of loss of the material in the tank. The initial loss will be liquid if the hole is below the liquid level; after the liquid level reaches the level of the hole, any additional loss is in gaseous form. If the hole is above the liquid level, the loss is in gaseous form.

II. LIMITATIONS

Liquid flow is assumed if the liquid level is above the hole, and no provision is made for two-phase flow, as might occur, for example, in a release of UF_6. Not considering two-phase flow should give a conservative result; that is, consideration of two-phase flow would tend to slow the rate of release. In the case that the line or hole "plugged", the actual amount of release indicated by the program would be greater

Prepared by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
managing the

Oak Ridge K-25 Site
Oak Ridge National Laboratory
Oak Ridge Y-12 Plant
under Contract DE-AC05-84OR21400

Uranium Enrichment Organization
Including the Paducah Gaseous Diffusion Plant
and the Portsmouth Gaseous Diffusion Plant
under Contract USECHQ-93-C-0001

for the
U.S. DEPARTMENT OF ENERGY

than the actual release. It is, however, difficult to impossible to predict plugging and the program result would show the "worst case"; that is, it would show the case where no plugging occurred which would result in a maximum loss. Consequently, the results obtained from LEAK, when used as input to a compartment or dispersion model, would be expected to result in higher concentrations or doses than would be expected from a more detailed analysis that would increase release duration or decrease the total amount of material released. [CYLIND, a computer code for evaluating the release of UF_6 from a cylinder, takes two-phase flow into account (Williams, 1985 and 1994). It is recommended that CYLIND be used for liquid UF_6 release studies.]

The release rate equations are also based on incompressible flow. This will result in some inaccuracy in estimating vapor release rates. However, since the primary scenario considers liquid release followed by vapor release, most material would normally be lost as a liquid and the amount and rate of vapor loss would be a secondary concern. In the case of fluorine, which is treated only as a gas in LEAK, the quantity normally stored in cylinders (6 lb maximum was the basis for this work) is very small and would be lost a matter of seconds for a hole of appreciable size. Such a release, in terms of compartment or dispersion modeling is essentially instantaneous; thus, the error in estimating release rate is inconsequential for the intended purpose of the code.[†] If large quantities of gas at high pressure are to be released, then compressible flow equations (not included in LEAK) should be utilized for those estimates.

A circular hole is assumed throughout the program. Other shapes must be approximated by a circular hole.

III. ANALYSIS PROCEDURE

1. **Input Requirements.** The following information must be known or estimated as input to LEAK for a leak from a tank:

    a. whether the tank is vertical or horizontal,
    b. the diameter of the hole, in.,
    c. the size of the tank in terms of its capacity, lb, kg, tons, gal, or ft^3 (or its diameter, ft or m),
    d. the height of the tank (or length if horizontal), ft,
    e. the amount of material in the tank at the time of the leak relative to its maximum capacity, %,
    f. the vertical location of the hole (as a percentage of the height), %,
    g. whether the tank is pressurized with an external gas (such as nitrogen) and the pressure, psia,

[†] The following message is printed to the screen if a user selects fluorine as the material of interest:

```
Fluorine is generally stored as a gas in quantities of 0.5
to 6 lb or more at up to 400 psig.  Fluorine may be stored
in quantity in liquid form cooled to below -307 F by
liquid nitrogen.  If you have the latter case, you need a
different analysis than exists in this program.  It is
assumed that you have the former case.
```

h. whether the tank is vented,
i. the outside temperature, °F,
j. the temperature of the material in the tank, °F.

If the leak is in a pipe, the following additional information is needed:

k. the inside diameter of the pipe, in.,
l. the length of the pipe from the point of fluid entry from the tank to the break, ft,
m. where the pipe enters the tank (as a percentage of the tank height referenced to the bottom of the tank), %,
n. if the pipe extends up or down into the tank, then where the end of the pipe is located in relation to the bottom of the tank (as a percentage of the tank height referenced to the bottom of the tank), %,
o. the elevation of the break in the pipe in relation to the bottom of the tank (as a percentage of the tank height referenced to the bottom of the tank), %.

2. **Running the Code.** LEAK is interactive and designed to be self explanatory. Necessary comments are printed on the screen as each question is asked. Generally, if a response to the question is incorrect or the expected range of the answer is exceeded, the user is alerted and given as many opportunities as necessary to deduce what the proper response should be. The executable file, LEAK.EXE, is run by typing **LEAK** and then entering a carriage return. If at any time in the input process it is desired to stop and try again or simply cease using the program, typing a **<ctrl-C>** will accomplish that task.

3. **Output from the Code.** Summary output is provided on-screen as LEAK completes execution. Additional output is provided in the file LEAK.OUT.

IV. SUPPORTING INFORMATION

Description of the Code. The FORTRAN source code for LEAK is presented in Attachment 1. LEAK has been compiled and linked using RM/FORTRAN, Version 2.4; the executable code, LEAK.EXE, has been run on several IBM-compatible personal computers. Attachment 2 shows the data file which the FORTRAN code requires. A detailed discussion of the code is not presented here because the code is fairly well commented. Also, the variable names used in the code are usually clear, for example: "PipeResistance, maxsize, tambient, is_vented, etc.". It should not be an insurmountable task to read the code and/or use a word processor to locate variables and follow the flow of logic in the program if that is desired.

The purpose of the code is to provide the rate of loss and to track the total amount of loss. Both quantities are functions of time. The rate of loss is given by:

$$dm/dt = A v \rho \qquad (1)$$

where dm/dt = rate of mass loss, lbm/s,
A = cross sectional area of the hole, ft^2,

v = velocity of the fluid out the hole, ft/s,
ρ = the upstream fluid density (gas or liquid), lbm/ft^3.

The velocity of fluid is calculated from the relevant terms of the Bernoulli equation for the flow of liquid or gas out of a hole:

$$v = [(2.0 \cdot \Delta P \cdot 144 \cdot g_c)/(K \cdot \rho)]^{0.5} \quad (2)$$

where ΔP = the difference in pressure within the tank at the hole and atmospheric pressure, psi,
g_c = gravitational constant
= 32.2 lbm·ft/(lbf·s^2),
K = resistance to flow out the hole including any resistance due to piping, dimensionless
= 2.5 + the pipe resistance.

For all gases, including fluorine, the theoretical sonic velocity is calculated to set an upper limit on the rate of release. Gases are treated as ideal, incompressible fluids, which tends to overestimate the release rate of the gas. The release time of gases tends to be small (seconds to a few minutes), so the error in terms of large scale atmospheric modeling is not expected to be great.

The Antoine equation, or equivalent, is used to calculate the vapor pressure of the liquid in the tank at the temperature of the system. One of two forms of the Antoine equation is used. The more common three term Antoine expression is used when the data is taken from Reid, Prausnitz, and Sherwood (1977, Appendix A):

$$\ln(P^\circ) = A - B/(T + C) \quad (3)$$

where ln = natural logarithm,
P° = vapor pressure, mm Hg,
A,B,C = constants for particular chemical species,
T = absolute temperature, K.

A two term expression for vapor pressure is used when data are taken from Lange's Handbook of Chemistry (Dean, ed., 1973, pp. 10-31 ff):

$$\log(P^\circ) = -52.23 \cdot B/T + C\ , \quad (4)$$

where log = log to the base 10,
P° = vapor pressure, mm Hg,
B,C = constants for particular chemical species,
T = absolute temperature, K.

In order to track the amount of mass lost, Eq. 1 is integrated using Euler integration with one second time intervals. The amount of mass lost during a one second interval is found by holding v and ρ constant for a one second time interval in Eq. 1 and multiplying Eq. 1 by the time increment of one second. The total amount of mass, M, lost including the n^{th} time interval is:

$$M = A\ (v_1\rho_1 + v_2\rho_2 + \cdots + v_n\rho_n)\ (1\ s) \quad (5)$$

The subscript indicates the value of the item at that particular time increment. The velocity and density are recalculated at the beginning of each time increment.

The programming involves the use of the above basic equations along with the logic necessary to treat various situations, including: the phase of the release, the flashing of liquid, additional flow resistance for the flow of material through a length of pipe, the elevation of the break in the pipe in relation to the tank, and heat exchange with the environment.

Euler integration is, at first glance, the least accurate form of integration. However, for a process where the amount of material lost is proportional to the amount remaining and/or consequent temperature, the process tends to be self-correcting. The first amount of loss will be too great because the driving force is constant over the time step; however, at the second time step, the amount of change will have been too great and consequently the driving force over the next time step will be too small. During the next time step the loss will be too great, and the process of integration will continue with this first "too great" and next "too small" alternating sequence. This alternating sequence of "too great", "too small", will come acceptably close to an answer given by a higher order integration technique. A second consideration is that, for an accident situation where sometimes the amount of material in the tank, the temperature of the contents of the tank, and even the size and elevation of the accidental hole must be a necessarily quick, and perhaps not particularly accurate estimate, any minor inaccuracies caused by a first order integration technique as opposed to a higher order technique are overwhelmed by the inherent inaccuracies of the required estimates the operator must make. The point is that any inaccuracies due to the particular integration technique used will be of little significance. Inaccuracies may occur in the situations which last only a very few seconds (less than 10 s, for example), because more release may be indicated than might actually occur. This result, however, is a conservative result.

If a leak occurs below the liquid level and the tank is not externally pressurized, the tank is not vented, and the vapor pressure of the material of interest is below atmospheric, the release of liquid will tend to build a vacuum in the vapor space above the liquid. This will retard the flow of liquid unless air enters in through the hole to rebuild the pressure toward atmospheric pressure. At most the air coming into the hole will volumetrically replace the liquid loss. One way to estimate the liquid loss in this case is to assume that the air bubbling into the hole is the volumetric equivalent of the liquid lost. The practical result is then that the cross sectional area available for the flow of liquid is reduced by approximately one half, thus resulting in a slower release. This assumption is used in the program and is considered conservative, that is the liquid release should be somewhat slower than indicated by the program, because the rate of air entering in through the hole would be expected to actually be less than volumetric and some vacuum effect would exist. Experimental data might be useful in improving this particular estimate in the program.

Limitations of the Code. Code limitations are described in Sect. II.

Input to the Code. Input requirements for LEAK are described in Sect. III, step 1. Attachment 3 is a copy of an interactive input session for one particular case. The attachment lists the questions asked with the responses entered by the user. The incorporation of additional materials into the data base is described subsequently.

Output of the Code. Upon running the program an immediate output is printed to the computer screen. The amount of output printed to the screen is purposefully limited to the amount which will fit on one screen (see the end of Attachment 3). A more comprehensive output is placed in a file called LEAK.OUT; Attachment 4 provides a listing of this output file which corresponds to the input listed Attachment 3. This file is overwritten each time the program is run, so if the output in LEAK.OUT is to be retained it should be copied before the program is run again. The file LEAK.OUT lists the rate of release with each time increment desired. Whether the release is in liquid or vapor form is also noted. This type of output can be entered into an atmospheric dispersion model for a study of the distribution of the chemical beyond the immediate area of release. For the liquid release, the amount immediately flashed into vapor form is also listed. A "print" time increment of 10 s is used in the data file. This increment may be changed to any value greater than or equal to one second by changing the appropriate entry in the data file, LEAK.DAT. The location of this datum is recognized by the comment in LEAK.DAT beyond the datum point.

Adding Materials to the Data Base. Attachment 2 shows the data file for the materials treated by the LEAK program. As can be seen from the data file, the data needed are the molecular weight, the heat capacity of the liquid (Btu/lb °F), the heat of vaporization (Btu/lb), the Antoine coefficients for vapor pressure, the specific gravity, and the boiling point (°F). The units for the Antoine coefficients can be deduced from the section above entitled "Description of the Code." Three lines are used for each material. The first line is a name and chemical abbreviation. The next line is the numerical data for the material as listed above. The third line is a blank line used to space the different materials.

In order to add a material to the data, the three lines as described above need to be added using the data for that particular material. The only potential problem is that if more than about 20 materials are present, the initial listing of the materials from which the user may choose will scroll past the computer screen when LEAK is executed. The solution is to have more than one data file and use the one with the item of current interest. The particular data file actually used must be named LEAK.DAT.

Only materials which are liquid at the initial pressure and temperature of the tank and remain in either the liquid or vapor state throughout the leak process may be added to the data file. Special materials, such those which will pass through the triple point in the temperature range of interest, would require additional programming. As examples, fluorine, which is in gaseous form through the process, and uranium hexafluoride, which can pass through the triple point in the temperature range of normal interest, required special programming beyond the programming necessary to handle the more "ordinary" materials.

Quality Assurance. As LEAK was developed, numerous hand calculations were performed to check the code. Where hand calculations were not specifically performed, the trend and magnitude of numerical output were observed. The code was informally tested by R. D. Sharp of the Computer Applications Division and his comments were favorable. Mr. Sharp also made suggestions on handling input errors made by a user of the code. His suggestions were implemented making the code more "user friendly".

The Energy Systems Safety Analysis Report (SAR) program mandates an independent verification of all calculations used in safety analysis. If the LEAK program is used in a SAR application, an independent

evaluation should be used to confirm the conclusions derived from the LEAK simulation. It is not adequate for the independent evaluation to only check the LEAK program input and output.

V. REFERENCES

John A. Dean, ed., *Lange's Handbook of Chemistry*, 11th Ed., McGraw Hill Book Co., New York, NY, 1973.

Robert C. Reid, John M. Prausnitz, and Thomas K. Sherwood, *The Properties of Gases and Liquids*, 3rd Ed., McGraw-Hill Book Co., New York, NY, 1977.

W. R. Williams, *Calculational Methods for Analysis of Postulated UF_6 Releases*, NUREG/CR-4360 (ORNL/ENG/TM-31), Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1985.

W. R. Williams, *Calculational Methods for Analysis of Postulated UF_6 Releases: Supplement 1*, NUREG/CR-4360, Supplement 1 (ORNL/ENG/TM-31/S1), Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1994.

VI. STANDARD TEXT

Text incorporated into safety analysis documents describing results obtained using LEAK should (1) describe the accident scenario; (2) cite the variables and their specific values as outlined in Sect. III, step 1, and derived from the scenario; (3) state that LEAK was used with reference to this report for details, and (4) present the results. Caveats relating to compressible flow, two-phase flow, plugging, or hole geometry (see Sect. II) should be included, if appropriate.

VII. ATTACHMENTS

Attachment 1

FORTRAN Source Code for LEAK

```
*-----------------------------------------------------------------------

* THIS IS A PROGRAM TO ASSIST IN DETERMINING SOURCE TERMS FOR
* LEAKING VESSELS.

      implicit real (a-h, j-z)
      real inert_fraction
      integer Unumber, nChemicals
      character*1 TankOrPipe, tankunits, units, OneCharacter,
     .            head_units, is_pressurized, valve_open, orientation,
     .            is_vented
      character*30 Name
      character nameunits*12
      common massfrac(-1:180), circle_height(-1:180)

      open(1,file = 'leak.dat', status = 'old')
      open(2,file = 'leak.out')

* Data:
      call circle
      pi      =   3.14159

* Constants:
      gc    = 32.2
      patm =  14.7

* Some presently needed initialization
      headheight = 0.0
      pipediam   = 1.0
      pipelength = 0.0
      head_units = '%'
* Let time increments be one second.
      deltime = 1.

* -- QUESTIONS FOR PROGRAM OPERATOR --------------------------------
 6001 continue
      assign 6001 to index
      write(*,*)' ------------------------------------------------ '
      write(*,*) ' '

* Read past the six lines of comments in the data file.
      do 700 i = 1,6
  730 format(a1)
  700 read(1,730) OneCharacter

* Read the names of the items and print out the materials in the data file.
      do 712 i = 1, 1000
      read( 1,731)     Name
  731 format(a30)
      nChemicals = i-1
      if (Name .eq. 'END OF CHEMICAL DATA         ') go to 713
      write(*,711)  i, Name
  711 format(' ',i4,' - ',a30)
      read( 1,730)  OneCharacter
      read( 1,730)  OneCharacter
  712 continue
  713 continue
      rewind 1

      print*, ' '
      print*,
     .' Enter the identifying number for the material of interest: '
      read(*,*, err=6000) ID
      if(ID .lt. 1 .or. ID .gt. nChemicals) go to 6000
      go to 7000
 6000 print*, ' '
```

```
      print*,
     .' * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *'
      print*,
     .' * * * *   Your entry was incorrect.  Please try again.  * * * *'
      print*,
     .' * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *'
      print*, ' '
      go to index
 7000 continue

* Read past the six lines of comments in the data file.
      do 720 i = 1,6
  720 read(1,730) OneCharacter

      do 714 i = 1, 3*(ID - 1)
      read( 1,730)     Name
  714 continue

      read( 1,731) Name
      write(*,*)   ' '
      write(*,732) Name
  732 format(' ',a30)
      read(1,*)    MolWt, cp, heatvap, A, B, C, Specificgrav, Tboil


      if(Name .eq. ' Fluorine.............F2 ') then
       write(*,1234)
 1234 format(/,
     .'  Fluorine is generally stored as a gas in quantities of 0.5 ',/,
     .'  to 6 lb or more at up to 400 psig.  Fluorine may be stored ',/,
     .'  in quantity in liquid form cooled to below -307 F by    ',/,
     .'  liquid nitrogen.  If you have the latter case, you need a  ',/,
     .'  different analysis than exists in this program.  It is     ',/,
     .'  assummed that you have the former case.',//)

      go to 520
      endif

      write(*,716) MolWt, cp, heatvap, A, B, C, Specificgrav, Tboil
  716 format(/,
     .'   MolWt    cp     Hvap      A       B       C       SG      TB',
     ./, 8f8.2,/)

      if(name .eq. ' Uranium hexafluoride..UF6') then

      print*, ' '
      write(*,*)' ------------------------------------------------- '
 6002 write(*,*)' ------------------------------------------------- '
      assign 6002 to index
      print*, ' '
      write(*,500)
  500 format(
     . '  Enter the number (1 to 7) which shows the tank of interest',//
     . '            tank  colloquial                 normal maximum   ',/
     . '  number    type    name      diameter         capacity        ',/
     . '  ------    ----   -------    --------      --------------     ',/
     . '    1        5B                 5 in            55 lb UF6     ',/
     . '    2        8A                 8 in           255 lb UF6     ',/
     . '    3       12B                12 in           460 lb UF6     ',/
     . '    4       30B    2.5 ton     29 in         5,020 lb UF6     ',/
     . '    5       48X    10  ton     48 in        21,030 lb UF6     ',/
     . '    6       48Y    14  ton     48 in        27,560 lb UF6     ',/
     . '    7       Other    (You wish to enter a different size)     ',//
     . '  Enter the appropriate number ( 1 to 7 ): ')

      read(*,*,err=6000) Unumber
      if(Unumber .lt. 1 .or. Unumber .gt. 7) go to 6000
      print*, ' '

* units = '5' sets the volume units to ft**3 which is the way items
* 1 to 6 above will be entered.  Item 7 will allow the units to be changed.
      units = '5'
      endif

      write(*,*)' ------------------------------------------------- '
      write(*,*) ' The reservoir (tank) is assumed to be shaped like a'
```

```
      write(*,*) ' cylinder.  If it is not then approximate the shape'
      write(*,*) ' as a cylinder.  Is the cylinder horizontal?'
 6047 write(*,*)
      assign 6047 to index
      write(*,*)
     .' Enter an h if the cylinder (tank) is horizontal '
      write(*,*)
     .'     or a v if the cylinder (tank) is vertical: '
      read(*,70) orientation
   70 Format(A1)
      if(orientation .eq. 'h') orientation = 'H'
      if(orientation .eq. 'v' .or. orientation .eq. 'V' .or.
     .   orientation .eq. 'H') go to 6040
      go to 6000

 6040 write(*,*)' ------------------------------------------------ '
      assign 6040 to index
      write(*,*) ' Is the hole in the tank or in a pipe? '
      write(*,*)
      write(*,*)
     .' Enter a 1 or t for a hole in a tank or a 2 or p for a pipe break
     .: '
      read(*,70) TankOrPipe
      if(TankOrPipe .eq. 'T' .or. TankOrPipe .eq. 't') TankOrPipe = '1'
      if(TankOrPipe .eq. '1') iTankOrPipe = 1
      if(TankOrPipe .eq. 'P' .or. TankOrPipe .eq. 'p') TankOrPipe = '2'
      if(TankOrPipe .eq. '2') iTankOrPipe = 2
      if(iTankOrPipe .eq. 1 .or. iTankOrPipe .eq. 2) go to 6048
      go to 6000

 6048 if(iTankOrPipe .eq. 1) then
 6003 write(*,*)' ------------------------------------------------ '
      assign 6003 to index
      write(*,*)
     .' Enter the diameter (in inches) of the hole in the tank: '
      read(*,*,err=6000) hole
      pipelength = 0.0
      pipediam   = 1.0
      go to 40
      endif

* A pipe break has occurred.

 6004 write(*,39)
      assign 6004 to index
   39 format(' ------------------------------------------------ ' ,/,
     .'  The length of pipe will be assumed to be 20 ft, and',/,
     .'  the diameter will be assumed to be 1 inch.',/,
     .'  If this is OK, then type 1.',/,
     .'  If you wish to enter your own length and diameter, type 2.',//,
     .'  Enter 1 for the default or 2 to enter a pipe length and diamete
     .r: ')
      pipelength = 20.0
      pipediam   =  1.0
      hole = 1.0
      read(*,*,err=6000) irespond
      if(irespond .eq. 1) go to 8005
      if(irespond .ne. 2) go to 6000
 6005 write(*,35)
      assign 6005 to index
   35 format(' ------------------------------------------------ ' ,/,
     .       '  Enter the pipe length in feet: ')
      read(*,*,err=6000) pipelength
 6006 write(*,*)' ------------------------------------------------ '
      Write(*,*)' Enter the pipe diameter in inches: '
      assign 6006 to index
      read(*,*,err=6000) pipediam
      hole = pipediam
 8005 continue
      assign 8005 to index
      write(*,*)' ------------------------------------------------ '
      Write(*,8006)
 8006 format(
     .'  Where does the pipe enter the tank?  Enter the location as ',/,
     .'  a percentage of the tank height (0 to 100).  For example,  ',/,
     .'  if the pipe enters the top of the tank, enter 100.  Or, if ',/,
```

```
     .'  the pipe enters two thirds of the way up the tank, enter   ',/,
     .'  67.  The value at the bottom of the tank is 0.',//,
     .'  Where does the pipe enter the tank? (0 to 100) ')
      read(*,*,err=6000) pipe_entry
      if( pipe_entry .lt. 0. .or. pipe_entry .gt. 100.) go to 6000
   60 continue
      assign 60   to index
      write(*,*)' ------------------------------------------------- '
      Write(*,61)
   61 format(
     .'  Enter the elevation of the END of the pipe in the tank as a',/,
     .'  percentage of the tank height (0 to 100).  For example, if ',/,
     .'  the pipe enters the tank at any point but extends up or    ',/,
     .'  down into the tank so that the end of the pipe is one      ',/,
     .'  fourth the way up from bottom of the tank, then enter 25.  ',/,
     .'  If the pipe enters the tank but does not extend into the   ',/,
     .'  tank either upward or downward, the answer is the same as  ',/,
     .'  the previous answer (that is, the elevation of the end of  ',/,
     .'  the pipe is the same as its entry point into the tank). ',//,
     .'  Where is the END of the pipe in the tank? (0 to 100) ')
      read(*,*,err=6000) holeheight
      if( holeheight .lt. 0. .or. holeheight .gt. 100.) go to 6000
 6007 write(*,*)' ------------------------------------------------- '
      write(*,62)
      assign 6007 to index
   62 format(
     .'  Enter the elevation of the pipe break.  If the break is    ',/,
     .'  above or below the tank you will enter a positive number to',/,
     .'  indicate the number of feet above the tank (for example, 10',/,
     .'  maeans 10 feet above the top of the tank, or you will enter',/,
     .'  a negative number to indicate the number of feet below the ',/,
     .'  tank (for example, -3 means three feet below the tank).    ',/,
     .'  If the break is between the top and bottom of the tank, you',/,
     .'  will enter the percent (0 to 100) of the tank height from  ',/,
     .'  the bottom (for example, 34 means 34% of the distance up   ',/,
     .'  the tank.  --------  You will first enter a number and a   ',/,
     .'  carriage return and then either a ''1'' or ''f'' for feet  ',/,
     .'  above or below the tank or a ''2'' or a ''p'' for per cent ',/,
     .'  indicating that the break is between the top and bottom of ',/,
     .'  the tank.',//,
     .'  Enter the numerical value for the location of the break: ')
      read(*,*,err=6000) headheight
* Note:  Later the headheight will be translated into % of tank height.
 6049 write(*,*)' ------------------------------------------------- '
      write(*,63)
      assign 6049 to index
   63 format('  Enter the explaination for your number,',/,
     .'  (1 or f for feet relative to the top of the tank, or',/,
     .'  (2 or p for per cent of the tank height relative to the tank bo
     .ttom: ')
      read(*,730) head_units
      if(head_units .eq. '1' .or. head_units .eq. 'F') head_units = 'f'
      if(head_units .eq. '2' .or. head_units .eq. 'P') head_units = 'p'
      if(head_units .eq. 'f' .or. head_units .eq. 'p') go to 40
      go to 6000

   40 continue

      if(name .eq. ' Uranium hexafluoride..UF6') then
      go to(501,502,503,504,505,506,520) Unumber

  501 continue
      full     =    55.
      maxsize  =     0.248
      tankdiam =     0.417
      height   =     2.0825
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521

  502 continue
      full     =   255.
      maxsize  =     1.319
      tankdiam =     0.667
      height   =     3.778
      iunits = 5
```

```
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521

  503 continue
      full     =   460.
      maxsize  =     2.388
      tankdiam =     1.0
      height   =     3.03
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521

  504 continue
      full     =  5020.
      maxsize  =    26.
      tankdiam =     2.417
      height   =     5.668
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521

  505 continue
      full     = 21030.
      maxsize  =   108.9
      tankdiam =     4.0
      height   =     8.6685
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521

  506 continue
      full     = 27560.
      maxsize  =   142.7
      tankdiam =     4.0
      height   =    11.356
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      go to 521
      endif

  520 continue
      if(Name .eq. ' Fluorine.............F2 ') then
      write(*,1242)
 1242 format('  ------------------------------------------------ ' ,/,
     .'  This program needs to know the size of the tank in either' ,/,
     .'  cubic feet or height and diameter.  Which are you going to use
     .?',/,
     .'     c (or 5) for cubic feet, or',/,
     .'     h (or 6) for the height (length, if horizontal) of tank.',/,
     .'              If you enter a height (you will then be ',/,
     .'              asked to enter a diameter and whether the ',/,
     . '             the height and diameter are in feet or meters.',//)

 6050 write(*,*)' Enter c for cubic feet or enter an h for height: '
      assign 6050 to index
      read(*,70)  units
      if(units .eq. 'c' .or. units .eq. 'C' .or. units .eq. '5')
     .go to 6008
      if(units .eq. 'h' .or. units .eq. 'H' .or. units .eq. '6')
     .go to 6008
      go to 6000
 6008 write(*,*)' Enter the numerical value for cubic feet or height: '
      assign 6008 to index
      read(*,*,err=6000) maxsize

      go to 1241
      endif

 6051 write(*,50)
      assign 6051 to index
   50 format('  ------------------------------------------------ ' ,/,
     .'  This program needs to know the maximum amount of material' ,/,
     .'  the tank or reservior can hold, or the size of the tank, or',/,
     .'  the height of the tank.  BUT FIRST, an explanation of the  ',/,
     .'  number you are going to enter is needed, specifically, enter',/
     .,
```

```
     .'     p (or 1) for pounds,    ',/,
     .'     k (or 2) for kilograms,',/,
     .'     t (or 3) for tons,      ',/,
     .'     g (or 4) for gallons,  ' ,/,
     .'     c (or 5) for cubic feet, or',/,
     .'     h (or 6) for the height (length, if horizontal) of tank.',/,
     .'               If you enter a height (you will then be ',/,
     .'               asked to enter a diameter and whether the ',/,
     .'               the height and diameter are in feet or meters.',/)
      write(*,*)
     .' Enter your choice: '
      read(*,70)  units
      if(units .eq. 'p' .or. units .eq. 'P' .or. units .eq. '1')
     .go to 6009
      if(units .eq. 'k' .or. units .eq. 'K' .or. units .eq. '2')
     .go to 6009
      if(units .eq. 't' .or. units .eq. 'T' .or. units .eq. '3')
     .go to 6009
      if(units .eq. 'g' .or. units .eq. 'G' .or. units .eq. '4')
     .go to 6009
      if(units .eq. 'c' .or. units .eq. 'C' .or. units .eq. '5')
     .go to 6009
      if(units .eq. 'h' .or. units .eq. 'H' .or. units .eq. '6')
     .go to 6009
      go to 6000

 6009 write(*,*)
     .' Now enter the numerical value: '
* In commenting the next statement, the decision is to repeat the

*      assign 6009 to index
      read(*,*,err=6000) maxsize
*   50 format('  ------------------------------------------------ ',/,
*     .'  Enter a numerical value for the maximum amount of material' ,/,
*     .'  the tank or reservior can hold, or the size of the tank, or',/,
*     .'  the height of the tank.  You will next be asked to enter an',/,
*     .'  explanation of your number value, specifically, ',/,
*     .'     p (or 1) for pounds,    ',/,
*     .'     k (or 2) for kilograms,',/,
*     .'     t (or 3) for tons,      ',/,
*     .'     g (or 4) for gallons,  ' ,/,
*     .'     c (or 5) for cubic feet, or',/,
*     .'     h (or 6) for the height (length) of a tank.',/,
*     .'               If you enter a height (you will then be ',/,
*     .'               asked to enter a diameter and whether the ',/,
*     . '              the height and diameter are in feet or meters.',//)
*      write(*,*)
*     .' Enter the numerical value for the tank size or maximum amount:
*     .'
*      read*, maxsize
*      write(*,*)
*     .' Enter the units (p, k, t, g, or c or enter an h for height)
*     .: '
*      read(*,70)  units
 1241 continue
      if(units .eq. 'H' .or. units .eq. 'h') units = '6'
      if(units .eq. '6') then
      iunits = 6
      height = maxsize
 6010 write(*,51)
      assign 6010 to index
   51 format('  ------------------------------------------------ ' ,/,
     .'  Enter the diameter of the tank, a carriage return, ',/,
     .'     and then an f if the height and diameter are in feet',/,
     .'     or       an m if in meters. ',//)
      read(*,*,err=6000) tankdiam
      read(*,70)  tankunits
      if(tankunits .eq. 'f' .or. tankunits .eq. 'F' .or.
     .   tankunits .eq. 'm' .or. tankunits .eq. 'M') go to 6052
      go to 6000
 6052 if(tankunits .eq. 'M' .or. tankunits .eq. 'm') then
      height = height * 3.281
      tankdiam = tankdiam * 3.281
      endif
      maxsize = height * pi * (tankdiam/2.)**2.
      maxmass = maxsize * 62.4 * Specificgrav
```

```
      if(Name .eq. ' Fluorine.............F2 ') go to 1239
      go to 42
      endif

      if(units .eq. 'P' .or. units .eq. 'p' .or. units .eq. '1') then
      iunits = 1
      maxmass = maxsize
      go to 41
      endif


      if(units .eq. 'K' .or. units .eq. 'k' .or. units .eq. '2') then
      iunits = 2
      maxmass = maxsize * 2.2
      go to 41
      endif

      if(units .eq. 'T' .or. units .eq. 't' .or. units .eq. '3') then
      iunits = 3
      maxmass = maxsize * 2000.
      go to 41
      endif

      if(units .eq. 'G' .or. units .eq. 'g' .or. units .eq. '4') then
      iunits = 4
      maxmass = maxsize * 8.34 * Specificgrav
      go to 41
      endif

      if(units .eq. 'C' .or. units .eq. 'c' .or. units .eq. '5') then
      iunits = 5
      maxmass = maxsize * 62.4 * Specificgrav
      if(Name .eq. ' Fluorine.............F2 ') go to 1239
      go to 41
      endif

   41 continue
      assign 41 to index
      if(iunits .ne. 6) then
      write(*,*)' ------------------------------------------------ '
      write(*,*)
     .' Enter height (length, if horizontal) of tank (in feet): '
      read(*,*,err=6000) height
      tankdiam = (4.0*maxsize/(Specificgrav * 62.4 * height * pi))**.5
      endif

  521 continue
   42 continue
* Change pipe break (headheight) into per cent of tank height.
      if(head_units .eq. 'f') then
        if(headheight .ge. 0.0)then
        headheight = 100.*(headheight + height) / height
        else
        headheight = 100.*(headheight         ) / height
        endif
      endif

* Check to see if the abs(headheight) is greater than the pipelength.
      upperlimit = pipe_entry + pipelength/height*100.
      lowerlimit = pipe_entry - pipelength/height*100.
      if(headheight .gt. upperlimit .or. headheight .lt. lowerlimit)
     .then

*       if(abs(headheight) .gt. 100.0*abs((pipelength+height)/height)
*      ..or. -pipelength   .gt. headheight*height) then
*       holefeet = height * headheight
      print*,
     .' * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *'
      print*,
     .' * In comparing the tank height, pipelength, and break height  *'
      print*,
     .' * it appears that the break height may not be consistent with *'
      print*,
     .' * the pipe length (break too high or too low).  Please enter  *'
      print*,
     .' * a different break height (this entry will not be checked).  *'
```

```
      print*,
     .' * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *'
      print*, ' '
 8000 write(*,62)
      index = 8000
      read(*,*,err=6000) headheight
 8001 write(*,63)
      assign 8001 to index
      read(*,730) head_units
      if(head_units .eq. '1' .or. head_units .eq. 'F') head_units = 'f'
      if(head_units .eq. '2' .or. head_units .eq. 'P') head_units = 'p'
      if(head_units .eq. 'f' .or. head_units .eq. 'p') go to 8002
      go to 6000
* Change pipe break (headheight) into per cent of tank height.
 8002 if(head_units .eq. 'f') then
        if(headheight .ge. 0.0)then
        headheight = 100.*(headheight + height) / height
        else
        headheight = 100.*(headheight         ) / height
        endif
      endif

      endif

      go to(201,202,203,204,205,206)iunits
  201 nameunits = ' pounds      '
      go to 220
  202 nameunits = ' kilograms   '
      go to 220
  203 nameunits = ' tons        '
      go to 220
  204 nameunits = ' gallons     '
      go to 220
  205 nameunits = ' cubic feet '
      go to 220
  206 nameunits = ' cubic feet '
      go to 220

  220 continue
      assign 220 to index
      if(head_units .eq. 'p' .or. head_units .eq. 'f') go to 9000
* "holeheight" is how far up the container the hole occurs (0 to 100%).
      write(*,38)
   38 format('  ------------------------------------------------ ' ,/,
     .'  Enter the location of the hole as a percentage of the ',/,
     .'  height of the tank (for example, 25 means 1/4 of the way',/,
     .'  up the tank).  Enter the location (0 to 100): ')
      read(*,*,err=6000) holeheight
      if( holeheight .lt. 0. .or. holeheight .gt. 100.) go to 6000
      headheight = holeheight

* Change holeheight to fraction.
 9000 holeheight = holeheight/100.
      headheight = headheight/100.

 6011 write(*,37)
      assign 6011 to index
   37 format('  ------------------------------------------------ ',/,
     .'  Enter the amount of material in the tank as a per cent of',/,
     .'  the maximum amount of the tank.  For example if the tank' ,/,
     .'  is 3/4 full, enter 75.',/,
     .'  Enter the amount (0 to 100): ')
      read(*,*,err=6000) HowFull
      if( HowFull .lt. 0. .or. HowFull .gt. 100.) go to 6000

*      print*, 'HowFull = ', Howfull
      if(name .eq. ' Uranium hexafluoride..UF6') then
      if(HowFull .gt. full/maxmass*100.) HowFull = full/maxmass*100.
      write(*,1783) int(Howfull)
 1783 format(
     .'  The normal maximum fill in this tank is',i3,'%.  This',/
     .,       '  percentage will be used in the calculations.')
      endif

      originalmass = HowFull/100. * maxmass
      massleft     = originalmass
```

```
 6053 write(*,*)' ------------------------------------------------ '
      assign 6053 to index
      write(*,*)
     .' Is the tank pressurized (for example, with nitrogen)? (Y/N): '
      read(*,70,err=6000) is_pressurized
      if(is_pressurized .eq. 'n') is_pressurized = 'N'
      if(is_pressurized .eq. 'N') then
 6055 write(*,*)' ------------------------------------------------ '
      assign 6055 to index
      write(*,*)
     .' Is the tank vented? (Y/N): '
      read(*,70,err=6000) is_vented
      if(is_vented .eq. 'n') is_vented = 'N'
      if(is_vented .eq. 'y') is_vented = 'Y'
        if(is_vented .ne. 'N') then
        if(is_vented .ne. 'Y') go to 6000
        endif
      endif
      if(is_pressurized .eq. 'y') is_pressurized = 'Y'
      if(is_pressurized .eq. 'Y' .or. is_pressurized .eq. 'N')
     .go to 6060
      go to 6000
 6060 if(is_pressurized .eq. 'Y') then
 6012 write(*,*)
     .' What is the pressure in the tank? (psig): '
      assign 6012 to index
      read(*,*,err=6000) tank_pressure
      tank_pressure = tank_pressure + 14.7
 6054 write(*,*)
     .' Is the pressure line open (to maintain the pressure)? (Y/N): '
      assign 6054 to index
      read(*,730,err=6000) valve_open
      if(valve_open .eq. 'n') valve_open = 'N'
      if(valve_open .eq. 'y') valve_open = 'Y'
      if(valve_open .eq. 'Y' .or. valve_open .eq. 'N') go to 1239
      go to 6000
      endif

 1239 continue

      if(Name .eq. ' Fluorine.............F2 ') then
 6013 write(*,1243)
      assign 6013 to index
 1243 format('  ------------------------------------------------ ',/,
     .'  Enter the pressure (psig) of the fluorine in the container: ')
      read(*,*,err=6000) pressure

 6014 write(*,*)' ------------------------------------------------ '
      assign 6014 to index
      write(*,*)
     .' Enter the diameter (in inches) of the hole in the tank: '
      read(*,*,err=6000) hole

      endif

 6015 write(*,*)' ------------------------------------------------ '
      assign 6015 to index
      write(*,*)' Enter the outside temperature (degrees F): '
      read(*,*,err=6000) tambient

 6016 write(*,*)' ------------------------------------------------ '
      assign 6016 to index
      write(*,*)
     .' Enter the temperature (degrees F) of the material in the tank: '
      read(*,*,err=6000) temp

      if(Name .eq. ' Fluorine.............F2 ') then
      massleft =
     . pressure/14.7 * maxsize * MolWt / (0.7302 * (temp + 460.0))
      rhofl = SpecificGrav * 62.4
      originalmass = massleft
      iTankorPipe  = 1
      endif

*  ---------------------- Initialization -------------------------
      d     = hole/12.0
```

```
      r     = d/2.
      rsave = r
      mass2 = 0.0
* The resistance to flow is an entrance and exit (= 2.5) plus
* any resistance due to a pipe which is f*l/d.  A good average
* friction factor is 0.022 for 1/2" to 4" pipes (f = 0.027 to 0.017).
      PipeResistance = 0.022*(pipelength*12.)/pipediam
      Resistance = 2.5 + PipeResistance
* Calculate the area (tank walls only) for vapor section heat transfer
* into tank (or out of the tank in the case of UF6).
      area   =  holeheight * height * pi * tankdiam
      areastart = area
      icount =   0
      heatin = 0.0

      do 744 i = 1, 1000
      read( 1,730) OneCharacter
      if(OneCharacter .eq. 'E') go to 745
  744 continue
  745 continue

      read(1,*) iend
* change iend to seconds and subtract 1 to get to requested end time.
      iend = iend * 60 - 1
      read(1,*) iprint

      coefficient = 0.5
* Make heat transfer coefficient a function of the temperature (more
* heat transfer in a fire).
      if(tambient-30. .gt. 1.0)
     . coefficient = coefficient * (tambient-30.)**.33333

* Change units on coefficient to Btu/(ft**2 F second)
      coefficient = coefficient / 3600.
      tcentigrade = 0.55555556 * (temp -32.)

      if(Name .eq. ' Fluorine.............F2 ') go to 1260

* vapor pressure.
      pressure = 0.0193363 * 2.7183**(A-B/(C+tcentigrade+273.))
      if(abs(A) .lt. 1.e-4)
     . pressure = 0.0193363 * 10.**(-52.23*B/(tcentigrade+273.)+C)
      vaporpressure = pressure
      if(pressure .lt. 14.7) pressure = 14.7
* If the tank is pressurized, the pressure will be the pressure setting,
* not the pressure setting plus the vapor pressure as it would in a
* closed tank.
      if(is_pressurized .eq. 'Y') then
      pressure = tank_pressure
      else
      tank_pressure = pressure
      endif
      time = 0.0
* WRITE OUT SOME OF THE INPUT WHICH THE USER ENTERED. -------------
 1260 write(*,*) ' '
      write(*,*) ' '
      write(2,*) ' '
      write(2,*)
     .'  ------------------------------------------------------'
      write(*,*)
     .'  ------------------------------------------------------'
*      write(*,*) ' '
*      write(2,*) ' '
      write(*,110)Name
      write(2,110)Name
      write(2,716) MolWt, cp, heatvap, A, B, C, Specificgrav, Tboil
  110 format(
     .'  ',a30)
      write(*,*) ' '
      write(2,*) ' '

      if(Name .eq. ' Fluorine.............F2 ') then
      write(*,1262)maxsize
      write(2,1262)maxsize
 1262 format('      Tank size: ',f10.2,' cubic feet')
      go to 1261
```

```
      endif

*      imaxsize = int(maxsize + .00001)
      write(*,111)maxsize, nameunits, height, tankdiam
      write(2,111)maxsize, nameunits, height, tankdiam
  111 format(
     .'      Tank size: ',f10.2,a12,'(height = ',f4.1,' ft,  diameter =
     .',f4.1,' ft)')

      if(orientation .eq. 'H') then

      write(*,1111)
      write(2,1111)
 1111 format(
     .'      Orientation:  - - - - Tank is horizontal (on its side).')

      else
      write(*,1110)
      write(2,1110)
 1110 format(
     .'      Orientation:  - - - - Tank is vertical (upright).')

      endif

      iHowFull = int(HowFull + .00001)
      if(is_pressurized .eq. 'Y') then
      write(*,613)iHowFull, tank_pressure - 14.7
      write(2,613)iHowFull, tank_pressure - 14.7
      else
      write(*,113)iHowFull
      write(2,113)iHowFull
      endif
  613 format(
     .'      The tank is    ',i3,' %  full and pressurized to ',f6.1,
     . ' psig.')
  113 format(
     .'      The tank is    ',i3,' %  full.')

 1261 continue
      ioriginalmass = int(originalmass + .00001)
      write(*,118)ioriginalmass
      write(2,118)ioriginalmass
  118 format(
     .'      Start with:',i7,'    pounds.')

      if(iTankOrPipe .eq. 1) then
      write(*,112)hole
      write(2,112)hole
  112 format(
     .'      Hole size: ',f10.2,' inch diameter.')
      else
      write(*,115)pipediam, int(pipe_entry), pipelength,
     . int(100.*holeheight)
      write(2,115)pipediam, int(pipe_entry), pipelength,
     . int(100.*holeheight)
  115 format(
     .'      Pipe diam:   ',f8.2,
     . ' inch.              Pipe entry: ',i3,' % elevation.',/,
     .'      Pipe length: ',f7.1,
     .'  ft (outside of tank).   Pipe end: ',i3,' % elevation.')
      endif

      if(Name .eq. ' Fluorine.............F2 ') go to 1263

      iholeheight = int(.00001 + headheight*100.)
*      if(head_units .eq. 'f' .or. head_units .eq. 'p')
*     .iholeheight = int(.00001 + headheight*100.)
      holefeet = height * headheight

      if(holefeet .lt. 0.0) then
      write(*,1161) -holefeet
      write(2,1161) -holefeet
 1161 format(
     .'      Break is     ',f7.1,'  ft below the tank.')

      else
```

```
      if(holefeet .le. height) then
      write(*,116)iholeheight
      write(2,116)iholeheight
  116 format(
     .'      Break is at    ',i3,' %  of the tank height.')

      else
      write(*,1162) holefeet - height
      write(2,1162) holefeet - height
 1162 format(
     .'      Break is     ',f8.2,' ft above the tank.')
      endif
      endif

 1263 itambient = int(tambient)
      write(*,117)itambient
      write(2,117)itambient
  117 format(
     .'      Ambient temp: ',i4,' F. ')

      itemp = int(temp)
      write(*,114)itemp
      write(2,114)itemp
  114 format(
     .'      Tank temp:    ',i4,' F.')

      write(*,*) ' '
      write(2,*) ' '
      write(2,*) ' '

      if(Name .eq. ' Fluorine.............F2 ') go to 1245

      if(name .eq. ' Uranium hexafluoride..UF6') then
      if(temp .lt. 133.8) then
      write(*,*)
     .' ------------------------------------------------------'
      write(2,*)
     .' ------------------------------------------------------'
      write(*,*)
     .' ------------------------------------------------------'
      write(2,*)
     .' ------------------------------------------------------'
      write(*,1237)
      write(2,1237)
 1237 format('  UF6 is in solid form below its sublimation point.',/,
     .'  Any release will be relatively slow.')
      go to 107
      endif
      if(temp .lt. 147.3) go to 1238
      endif

      write(2,*)
     .'  time   temperature   pressure   liquid loss   mass lost'
      write(2,*)
     .' h: m: s      (F)         (psia)      (lb/min)        (lb)  '
      write(2,*)
     .' -----  -----------   --------   -----------   ---------'
      loss = 0.0
*  Initial state of variables printed to output.
*      write(2,1) time, temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout

*  VAPOR OR LIQUID ESCAPE AT FIRST?
      rhofl = SpecificGrav * 62.4
      stuff_fraction = vaporpressure/tank_pressure
      if(holeheight .ge. originalmass/maxmass) go to 99
*  LIQUID ESCAPE ------------------------------------------------

      stuff_fraction = vaporpressure/tank_pressure
      inert_fraction = 1.0 - stuff_fraction
      moles_inert = tank_pressure *
     . inert_fraction * (1.- HowFull/100.)*maxft3
     .          / (0.7302 * temp+460.)
```

```
* BEGIN LIQUID ESCAPE ----------------------------------------
* The following statement is a crude approach to account for the fact that
* a non-vented tank will (1) tend to form some vacuum as it begins to leak
* liquid and (2) air must bubble into the hole to replace lost liquid volume
* to decrease that vacuum.  The crude assumption is that the liquid volume
* lost is replaced by an equal volume of air bubbling into the hole (so no
* vacuum will be formed.  Since a volume of air equal to the volume of liquid
* lost is assummed to be coming into the hole, the effect is to reduce the
* flow area by one half.  The effective r is 0.707107 time the original r.
* Also, the flow of air into a non-vented tank is necessary only if the vapor
* pressure of the material is less than or equal to atmospheric pressure.
      if(is_vented .eq. 'N' .and. vaporpressure .le. 14.7)
     . r = 0.707107 * r
* The value for r will be reset to the original value before the vapor loss
* section.
* ----------------------------------------------------------------
      indexA = 0
* Note: Time increment is one second.
      do 98 i = 1, iend
      time = time + deltime
      icount = icount + 1

      fraction     = massleft/maxmass
      savefraction = fraction
      if(orientation .eq. 'H') then
      do 3333 indexH = indexA, 180
      if(fraction .gt. massfrac(indexH)) go to 3334
 3333 continue
 3334 indexA   = indexH
      fraction = circle_height(indexH) +
     .(fraction           - massfrac(indexH)) /
     .(massfrac(indexH-1) - massfrac(indexH)) *
     .(circle_height(indexH-1) - circle_height(indexH))
      endif
      head = (fraction - headheight) *
     . height * rhofl / 144.

* By using "= pressure" below, the assumption is made that any vapor needed
* to maintain the vapor pressure can be produced with negligible effect
* on the temperature of the liquid.
      liquidpressure = pressure + head

      if(   is_pressurized .eq. 'Y'            .and.
     .      vaporpressure  .lt. tank_pressure) then
        if( valve_open .eq. 'Y' )  then
        liquidpressure = tank_pressure + head
* Nitrogen (molecular weight = 28.0) is assummed to be the inert gas.
        MolWt = stuff_fraction * saveMolWt + inert_fraction * 28.0

        else
* Valve to pressurizing gas is closed.
        moles_stuff = vaporpressure * (1.- massleft/maxmass)*maxft3
     .              / (0.7302 * temp+460.)
        stuff_fraction = vaporpressure/tank_pressure
        inert_fraction = 1.0 - stuff_fraction
        endif
      endif

* Escape velocity, v has units of ft/s.  Must have volout in ft**3/min.
      deltaP = liquidpressure - patm

      if(deltaP .le. 0.0) then
*      print*, '    fraction at deltaP = ', fraction
      if(massout .gt. originalmass) massout = originalmass
      write(*,102) massout, time/60.
      write(2,221) massout, time/60.
  221 format(
     .'   The liquid loss is ',f9.2,' lbs in ',f7.2,' minutes.')
      if(iTankorPipe .ne. 1) write(*,120)
  120 format(/,
     .'    No flow (or further flow) will occur because the head in the
     .pipe is',/,
     .'    sufficient to stop the flow.',/)

      go to 99
      endif
```

```
      if(orientation .eq. 'H') then

      if(fraction .le. holeheight) then
      liquid_out = originalmass *
     .(1. - savefraction/(HowFull1/100.))
        if(massout .gt. liquid_out) then
        massout  = liquid_out
        massleft = originalmass - massout
        if(massout .gt. originalmass) massout = originalmass
        endif
*      write(2,1) time/60., temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
      write(2,*) ' '
      write(2,*) '  The liquid level reaches the hole.'
      go to 99
      endif

      else

      if(massleft/maxmass .le. holeheight) then
      liquid_out = originalmass * (1. - holeheight/(HowFull1/100.))
        if(massout .gt. liquid_out) then
        massout  = liquid_out
        massleft = originalmass - massout
        if(massout .gt. originalmass) massout = originalmass
        endif
*      write(2,1) time/60., temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
      write(2,*) ' '
      write(2,*) '  The liquid level reaches the hole.'

      go to 99
      endif

      endif

      v = ((2. * deltaP * 144. * gc)/(Resistance*rhofl))**0.5
      volflow  = v*pi*r**2.
      volout   = volflow * deltime
      loss     = volout  * rhofl

      massout  = massout  + loss
      massleft = massleft - loss

      if(icount .eq. iprint) then
*     write(2,1) time/60., temp, pressure, loss*60./deltime,
*    . massout
*   1 format(f8.3, f10.1, f12.1, f12.1, f14.1)
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
    1 format(i3,':',i2,':',i2, f9.1, f12.1, f12.1, f14.1)

      icount   = 0
      endif

   98 continue
* END LIQUID ESCAPE  ---------------------------------------
   99 continue
      time1 = time
      if(massout .gt. originalmass) massout = originalmass
      mass1 = massout
      write(*,102) massout, time/60.
      write(2, 10) massout, time/60.
   10 format(
     .'   The liquid loss is ',f7.1,' lbs in ',f7.2,' minutes.')
      write(2,*) ' '

      if(massleft .lt. 0.01) then
```

```
      write(*,*) '   All of the material is lost.  '
      write(2,*) '   All of the material is lost.  '
*      loss = 0.0
* The following section accounts for the flashing (immediate evaporation)
* in the event that all of the liquid is lost (the hole is at the bottom).
      if(temp .gt. Tboil) then
        if(Name .eq. ' Uranium hexafluoride..UF6')then
        firstmass    =  originalmass * 0.135 * (temp  - 147.3) / 36.0
        fusionmass   = (originalmass - firstmass)*0.395
        sublimemass = (originalmass - firstmass - fusionmass)
     1                              * 0.123 * (147.3 - 133.8) / 59.5
        poolflash = firstmass + fusionmass + sublimemass
        flashfrac    = poolflash/massout
        else
        poolflash = massout*cp*(temp - Tboil)/heatvap
        flashfrac = poolflash/massout
        endif

        if(flashfrac .gt. 1.0) then
        flashfrac = 1.0
        poolflash = massout
        endif
      iflashfrac  = int(100.*flashfrac)
      write(*,1007) iflashfrac, poolflash
      write(2,1007) iflashfrac, poolflash
      endif
      go to 107
      endif
* This ends the flashing section in the even that all the liquid is lost.

      if(i .ge. iend) go to 101

      write(2,*)
     .'  time   temperature   pressure    vapor loss   mass lost'
      write(2,*)
     .' h: m: s      (F)        (psia)      (lb/min)      (lb)   '
      write(2,*)
     .'  -----  -----------   --------    ----------   ---------'


* VAPOR ESCAPE ----------------------------------------------------
      r = rsave
      if(icount .eq. 10) icount = 0
      tcentigrade = 0.55555556 * (temp -32.)
      pressure = 0.0193363 * 2.7183**(A-B/(C+tcentigrade+273.))
      if(abs(A) .lt. 1.e-4)
     . pressure = 0.0193363 * 10.**(-52.23*B/(tcentigrade+273.)+C)
      vaporpressure = pressure
* Unless a flow capacity of pressurizing gas into the tank is taken into
* account the assumption is made that any gas lost can be replaced.
      maxft3 = height * pi * (tankdiam/2.)**2
      if(is_pressurized .eq. 'Y') then
      pressure = tank_pressure
      stuff_fraction = vaporpressure/tank_pressure
      inert_fraction = 1.0 - stuff_fraction
      moles_inert = tank_pressure *
     . inert_fraction * (1.- HowFull/100.)*maxft3
     .          / (0.7302 * temp+460.)
      endif

* deltaP is necessary for an 'if' test near the end of the program.
      deltaP = pressure - patm
      if(deltaP .le. 0.0) then
      go to 101
      endif

      iend = iend - i + 1
* BEGIN VAPOR ESCAPE ----------------------
*      if(icount .ge. iprint) icount = 0
      saveMolWt = MolWt
      stuff_fraction = 1.0

      do 100 i = 1, iend
      time = time + deltime
      icount = icount + 1
      tcentigrade = 0.55555556 * (temp -32.)
```

```
      pressure = 0.0193363 * 2.7183**(A-B/(C+tcentigrade+273.))
      if(abs(A) .lt. 1.e-4)
     . pressure = 0.0193363 * 10.**(-52.23*B/(tcentigrade+273.)+C)
      vaporpressure = pressure
* Unless a flow capacity of pressurizing gas into the tank is taken into
* account the assumption is made that any gas lost can be replaced.
      if(   is_pressurized .eq. 'Y'            .and.
     .      vaporpressure  .lt. tank_pressure  .and.
     .      valve_open     .eq. 'Y' )  then
        pressure = tank_pressure
        stuff_fraction = vaporpressure/tank_pressure
        inert_fraction = 1.0 - stuff_fraction
* Nitrogen (molecular weight = 28.0) is assummed to be the inert gas.
        MolWt = stuff_fraction * saveMolWt + inert_fraction * 28.0

        else
* Valve to pressurizing gas is closed.
        moles_stuff = vaporpressure * (1.- HowFull/100.)*maxft3
     .              / (0.7302 * temp+460.)
      endif

      rhofg = pressure/14.7 * MolWt /(.7302 * (temp+460.))
* Escape velocity, v has units of ft/s.  Must have volout in ft**3/min.
* If UF6 then go to 108 after triple point is reached.
      if(name .eq. ' Uranium hexafluoride..UF6'
     .        .and.  pressure .le. 22.04 ) then

      if(massout .gt. originalmass) massout = originalmass
*      write(2,1001) time/60., temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
      write(2,1002)
 1002 format(//,
     .' Triple point reached.  During the next period of time the UF6',/
     .,
     .' will solidify, giving up the heat of fusion.',/)

      write(2,*)
     .'  time   temperature   pressure    vapor loss   mass lost'
      write(2,*)
     .' h: m: s      (F)         (psia)      (lb/min)      (lb)    '
      write(2,*)
     .'  -----  -----------   --------    ----------   ---------'

      go to 108
      endif

      deltaP = pressure - patm
      if (deltaP .gt. 0.0) then
      v = ((2. * deltaP * 144. * gc)/(Resistance*rhofg))**0.5
      else
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
      go to 108
      endif

* Check for sonic velocity.
      vsonic = 223.1 * ((temp+460.)/MolWt)**0.5
      if(vsonic .lt. v) v = vsonic
* Account for the fact that the material leaving is part inert and part
* 'stuff' of interest.
      v = v * stuff_fraction

* convert vapor to liquid equivalent.
      v = v * rhofg/rhofl
      volflow  = v*pi*r**2.
      volout   = volflow * deltime
      losssave = loss
      loss     = volout  * rhofl
      if(name .ne. ' Uranium hexafluoride..UF6') then
      if(abs(losssave - loss) .lt. 1.0e-8) go to 108
      endif
      massout  = massout  + loss
```

```
      massleft = massleft - loss

      if(massleft .lt. 0.0001) then
      write(*,*) '   All of the material is lost.  '
      write(2,*) '   All of the material is lost.  '
*      loss = 0.0
      go to 108
      endif

* Heat transfer in

      heatin = area * coefficient * (tambient - temp) * deltime
      area   = areastart * massleft/maxmass

* Replace the gas lost with flashing from the liquid.
* Heat required to evaporate more "dummy".
      heatflash = loss * heatvap
* New temperature of liquid (and gas for next increment).
      heatnet = heatflash - heatin
      temp = temp - heatnet/((massleft + loss)*cp)

      if(ipressureflag .eq. 0) then
      if(abs(pressure-15.0) .lt.  .01      .or.
     .                loss  .lt.  0.015 )  then
      ipressureflag = 1
      if(massout .gt. originalmass) massout = originalmass
      mass2 = massout
      time2 = time
      endif
      endif

      if(icount .eq. iprint) then
      if(massout .gt. originalmass) massout = originalmass
*      write(2,1001) time/60., temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss*60./deltime, massout
 1001 format(i3,':',i2,':',i2, f9.1, f12.1, f15.4, f11.1)

      icount = 0
      if(loss .lt. .000017) go to 108
      endif
  100 continue
* END VAPOR ESCAPE ---------------------------------
      go to 109
  108 continue
* THIS SECTION ACCOUNTS FOR THE HEAT OF FUSION OF UF6.
      if(name .eq. ' Uranium hexafluoride..UF6') then
** Account for heat of fusion:
* (1-x)(heat of fusion) = (x)(heat of vaporization); x = fraction vaporized
* x = 0.395
      mass_evaporated = 0.395 * massleft
      pressure = 22.04
** flow rate:
      deltaP = pressure - patm
      v = ((2. * deltaP * 144. * gc)/(Resistance*rhofg))**0.5
** convert vapor to liquid equivalent.
      v = v * rhofg/rhofl
      volflow    = v*pi*r**2.
*     fusiontime = (    VOLUME LOST      )/volflow
      fusiontime = (mass_evaporated/rhofl)/volflow
      volout   = volflow * fusiontime
      loss     = volout  * rhofl

      massout  = massout  + loss
      massleft = massleft - loss
      time  = time + fusiontime
      time2 = time
      mass2 = massout
      if(massout .gt. originalmass) massout = originalmass
*      write(2,1001) time/60., temp, pressure, loss*60./fusiontime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss*60./fusiontime, massout
```

```
*
* Now account for the sublimation of UF6 from the solid mass (from 22.04
* to 14.7 psia (this will be dependent upon the heat transfer through
* the solid, probably jumbled, crystalline mass of UF6).  Assume
* (conservatively) that heat transfer is not a hinderance to the
* sublimation process.
*
      temp = 147.3
 1238 btus = massleft * cp * (temp - 133.8)
      mass_evaporated = btus / 59.5
      pressure = (22.04 + patm)/2.0
* change "temp = and rhofg =" later if go to time steps on sublimation.
      temp = 133.8
      rhofg = pressure/14.7 * MolWt /(.7302 * (temp+460.))

** flow rate:
      deltaP = pressure - patm
      v = ((2. * deltaP * 144. * gc)/(Resistance*rhofg))**0.5
** convert vapor to liquid equivalent.
      rhofl = SpecificGrav * 62.4
      v = v * rhofg/rhofl
      volflow    = v*pi*r**2.
*     sublimetime = (    VOLUME LOST      )/volflow
      sublimetime = (mass_evaporated/rhofl)/volflow
      volout   = volflow * sublimetime
      loss     = volout  * rhofl

      massout  = massout  + loss
      massleft = massleft - loss
      time = time + sublimetime

      pressure = patm
      temp = 133.8

      write(2,1004)
*      write(3,1004)
 1004 format(//,
     .' Sublimation of the solid UF6 will now occur.  This is tracked un
     .til',/,' the vapor pressure reaches 1 atm.',/)

      write(2,*)
     .'  time   temperature   pressure    vapor loss   mass lost'
      write(2,*)
     .' h: m: s      (F)         (psia)      (lb/min)      (lb)    '
      write(2,*)
     .'  -----  -----------   --------    ----------   ---------'

      if(massout .gt. originalmass) massout = originalmass
*      write(2,1001) time/60., temp, pressure, loss*60./sublimetime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss*60./sublimetime, massout

      endif
      go to 109
 1245 continue
* BEGIN Fluorine ESCAPE ++++++++++++++++++++++++++++++++++++++++++++++

      write(2,*)
     .'  time   temperature   pressure     gas loss    mass lost'
      write(2,*)
     .' h: m: s      (F)         (psia)       (lb/s)       (lb)    '
      write(2,*)
     .'  -----  -----------   --------    ----------   ---------'

      do 1240 i = 1, iend
      time = time + deltime
      icount = icount + 1

      rhofg = pressure/14.7 * MolWt /(.7302 * (temp+460.))
* Escape velocity, v has units of ft/s.  Must have volout in ft**3/min.

      deltaP = pressure - patm
      if (deltaP .gt. 0.0) then
```

```
      v = ((2. * deltaP * 144. * gc)/(Resistance*rhofg))**0.5
      else
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss/deltime, massout
      go to 1255
      endif

* Check for sonic velocity.
      vsonic = 223.1 * ((temp+460.)/MolWt)**0.5
      if(vsonic .lt. v) v = vsonic

* convert vapor to liquid equivalent.
      v = v * rhofg/rhofl
      volflow  = v*pi*r**2.
      volout   = volflow * deltime
      losssave = loss
      loss     = volout  * rhofl
      massout  = massout  + loss
      massleft = massleft - loss

      if(massleft .lt. 0.0001) then
      write(*,*) '   All of the material is lost.  '
      write(2,*) '   All of the material is lost.  '
*      loss = 0.0
      go to 1255
      endif

* Heat transfer in

      heatin = area * coefficient * (tambient - temp) * deltime
      area   = areastart * massleft/maxmass

* New temperature of gas for next increment.
      temp = temp - heatin/((massleft + loss)*cp)

      if(ipressureflag .eq. 0) then
      if(abs(pressure-15.0) .lt.  .01      .or.
     .                loss  .lt.  0.015 )  then
      ipressureflag = 1
      if(massout .gt. originalmass) massout = originalmass
      mass2 = massout
      time2 = time
      endif
      endif

      if(icount .eq. iprint) then
      if(massout .gt. originalmass) massout = originalmass
*      write(2,1001) time/60., temp, pressure, loss*60./deltime,
*     . massout
      call TIMESUB(time+.5,ihour,imin,isec)
      write(2,1001) ihour, imin, isec, temp, pressure,
     . loss/deltime, massout

      icount = 0
      if(loss .lt. .000017) go to 1255
      endif
* Update fluorine pressure.
      pressure =
     . 14.7 * massleft/MolWt * 0.7302 * (temp+460.0) / maxsize

 1240 continue

 1255 write(2,1003)

      if(massout .gt. originalmass) massout = originalmass
      write(*,1251) massout, time/60.
      write(2,1251) massout, time/60.
 1251 format(
     .'    The loss is ',f6.0,' lbs in ',f7.2,' minutes.')

      go to 107
* END Fluorine ESCAPE ++++++++++++++++++++++++++++++++++++++++++++++++

  109 continue
```

```
      write(2,1003)
 1003 format(//'                         Summary',/
     .'                         -------',/)

      write(2,*) ' '
      write(2,102) mass1, time1/60.
  102 format(
     .'     The liquid   loss is ',f9.2,' lbs in          ',f7.2,' minute
     .s.')

      flashfrac = (massout - mass1)/(originalmass - mass1)
      poolflash = flashfrac * mass1
      iflashfrac = int(100.*flashfrac)
      if(mass1 .gt. .01) then
      write(*,1007) iflashfrac, poolflash
      write(2,1007) iflashfrac, poolflash
      endif
 1007 format(
     .'     (',i2,' % of the liquid, ', f9.2,' lbs, flashes quickly.)')
      mass2save = mass2
      if(mass2 .gt. 1.e-4) then
      write(*,103) mass2-mass1, (time2-time1)/60.
      write(2,103) mass2-mass1, (time2-time1)/60.
      else
      mass2 = mass1
      time2 = time1
      endif
      if(massout .gt. mass1) then
      write(*,103) massout-mass2, (time-time2)/60.
      write(2,103) massout-mass2, (time-time2)/60.
      endif
  103 format(
     .'     The vapor    loss is ',f9.2,' lbs in the next ',f7.2,' minute
     .s.')

  101 continue

      if(massout .gt. originalmass) massout = originalmass
      write(*,104) massout, time/60.
      write(2,104) massout, time/60.
  104 format(
     .'     The total loss is    ',f9.2,' lbs in          ',f7.2,' minute
     .s.')

      if(temp .gt. tambient) then
      write(*,106)
      write(2,106)
  106 format(/,
     .'     The vapor pressure of the material is one atmosphere or less'
     .,/,
     .'     at the ambient temperature, so any further loss will be due'
     .,/,
     .'     to diffusion out the hole, and thus small after this point.')
      go to 107

      else
      loss = heatin / heatvap
      if(loss*60. .gt. 1.66667e-6) then
      write(*,105) loss*60.
      write(2,105) loss*60.
  105 format(/,
     .'     Mass is still being lost at a rate of ',f8.4,' lbs per minute
     .'
     .,/,'     due to heat transfer into the tank.')
      endif
      endif

  107 continue

*      write(*,*) ' '
*      write(2,*) ' '
      write(*,*)
     .' ------------------------------------------------------------'
      write(2,*)
     .' ------------------------------------------------------------'
```

```
      stop
      end

      subroutine circle
      implicit real (a-h, j-z)
      common massfrac(-1:180), circle_height(-1:180)
*     open(3,file = 'area.out')

*     1/pi   = 0.3183098
      massfrac(      -1) = 1.00001
      circle_height( -1) = 1.00001

      circle_height(  0) = 1.0
      circle_height( 50) = 0.5
      circle_height(100) = 0.0
      massfrac(  0) = 1.0
      massfrac( 50) = 0.5
      massfrac(100) = 0.0

      do 1 i = 1,49
      angle = acos((1.0 - real(i)/50.0)/1.0)

      massfrac(100-i) = angle*5.555556e-3/0.0174532
     .          - sin(angle)*cos(angle)*0.3183098
      massfrac(  i  ) = 1.0 - massfrac(100-i)
      circle_height(100-i) =        real(i)/50.0 /2.0
    1 circle_height(  i  ) = (2.0 - real(i)/50.0)/2.0

*     write(3,*) '     height    fraction of circle'
*     write(3,*) '     ------    ------------------'
*     write(3,*) ' '

*     do 20,i=100,0,-1
*  20 write(3,210) circle_height(i), massfrac(i)
* 210 format(4x,f10.6,5x,f10.6)

      return
      end

* Subroutine TIMESUB changes seconds to integer hours, minutes, and seconds.
      SUBROUTINE TIMESUB(SECONDS,IHOUR,IMIN,ISEC)
      XTIME = SECONDS/3600.
      IHOUR = INT(XTIME)
      IMIN  = INT(( XTIME - REAL(IHOUR))*60.)
      ISEC  = INT(((XTIME - REAL(IHOUR))*60. -REAL(IMIN))*60.)
      RETURN
      END
```

Attachment 2

Data Base Input File

The following is the data file, LEAK.DAT, for the code, LEAK.FOR. The first line of the actual file is "Name of chemical".

```
Name of chemical
                                Antoine Coefficients
       liquid   heat   ----------------------
MolWt    cp    of vap    A        B       C       SG       BP
------  -----  ------  ------  ------  ------  -----    ----
                                                                  Cp: Lange's
                                                                    11th,
Acetonitrile.....C2H3N
  41.    0.541  329.    16.2874 2945.47  -49.15  0.782    179.    p. 9-120.

Ammonia..........NH3
  17.    1.1    589.    16.9481 2132.50  -32.98  0.817   -28.5

Carbon tetrachloride..CCl4                                      Cp: Perry's 5th.
 153.8   0.2     83.9   15.8742 2808.19  -45.99  1.584   169.5

Chlorine.........Cl2
  70.91  1.0    123.94 15.96   1978.32  -27.01  1.56    -30.3

Chlorine trifluoride..ClF3
  92.45  0.275   71.17  7.3671 1096.915 232.75  1.82     52.6

Fluorine.........F2
  38.    1.11   148.0  15.67    714.1    -6.00  1.51    -307.

Hydrogen fluoride.....HF
  20.    0.87   162.    17.6958 3404.49   15.06  0.967    66.9

Nitric acid.....HNO3
  63.    0.52   970.0   1.0      0.       0.    1.1     186.

Nitrogen tetroxide....N2O4                                    (-8 C to 43.2 C)
  92.    0.52   702.0   0.0     33.43     8.814 1.45     70.

Sodium hydroxide......NaOH
  92.45  0.78   970.0   1.0      0.       0.    2.13   2533.   (pure NaOH)

Sulfur hexafluoride...SF6
 146.    0.3     27.1   0.0     18.3209 7.4533  1.2     -82.7  (sublimation)

Sulfuric acid.....H2SO4
  63.    0.335  970.0   1.0      0.       0.    1.8     643.

Uranium hexafluoride..UF6
 352.    0.132   36.    0.0     41.73    9.521  4.      133.8

END OF CHEMICAL DATA
300     This is iend, the minutes at the end of the simulation.
 10     This is iprint, the seconds between printing intervals.
END OF ALL DATA

THE FOLLOWING ARE COMMENTS ON THE DATA.

* Antoine coefficients for ClF3: Matheson Gas Data Handbook, 5th Ed., p. 131.
* Antoine coefficients of (A,B,C) = (1,0,0) give a low vapor
*   pressure for leak purposes.
* Antoine coefficients are taken from Reid, Prausnitz, and Sherwood unless
*   the initial coefficient is 0.0.  In this case the coefficients were taken
*   from Lange's Handbook of Chemistry.
* The heat capacity (cp) of the liquid has units of Btu/pound/degrees F.
* Where heat capacity is 1.0, this is a guess.
* Heat of vaporization units are Btu/pound.
* BP - Boiling point (degrees F)
```

Attachment 3

Example of Interactive Input Session

```
------------------------------------------------          Material Selection

  1 -  Acetonitrile.....C2H3N
  2 -  Ammonia..........NH3
  3 -  Carbon tetrachloride..CCl4
  4 -  Chlorine.........Cl2
  5 -  Chlorine trifluoride..ClF3
  6 -  Fluorine.........F2
  7 -  Hydrogen fluoride.....HF
  8 -  Nitric acid......HNO3
  9 -  Nitrogen tetroxide....N2O4
 10 -  Sodium hydroxide......NaOH
 11 -  Sulfur hexafluoride...SF6
 12 -  Sulfuric acid.....H2SO4
 13 -  Uranium hexafluoride..UF6

Enter the identifying number for the material of interest: 2          <- INPUT

Ammonia..............NH3

 MolWt     cp      Hvap       A        B        C        SG      TB
 17.00     1.10   589.00    16.95  2132.50   -32.98     0.82   -28.50

------------------------------------------------          Container/Breach
The reservoir (tank) is assumed to be shaped like a       Characterization
cylinder.  If it is not then approximate the shape
as a cylinder.  Is the cylinder horizontal?
Enter an h if the cylinder (tank) is horizontal
    or a v if the cylinder (tank) is vertical: v          <- INPUT

------------------------------------------------
Is the hole in the tank or in a pipe?

Enter a 1 or t for a hole in a tank or a 2 or p for a pipe break: t   <- INPUT

------------------------------------------------
Enter the diameter (in inches) of the hole in the tank: 0.5          <- INPUT

------------------------------------------------
This program needs to know the maximum amount of material
the tank or reservior can hold, or the size of the tank, or
the height of the tank.  BUT FIRST, an explanation of the
number you are going to enter is needed, specifically, enter
   p (or 1) for pounds,
   k (or 2) for kilograms,
   t (or 3) for tons,
   g (or 4) for gallons,
   c (or 5) for cubic feet, or
   h (or 6) for the height (length, if horizontal) of tank.
            If you enter a height (you will then be
            asked to enter a diameter and whether the
            the height and diameter are in feet or meters.

Enter your choice: p                                          <- INPUT

Now enter the numerical value: 1300                           <- INPUT

------------------------------------------------
Enter height (length, if horizontal) of tank (in feet): 10    <- INPUT

------------------------------------------------          Initial Conditions
Enter the location of the hole as a percentage of the
height of the tank (for example, 25 means 1/4 of the way
up the tank).  Enter the location (0 to 100): 25              <- INPUT
```

```
-------------------------------------------------
Enter the amount of material in the tank as a per cent of
the maximum amount of the tank.  For example if the tank
is 3/4 full, enter 75.
Enter the amount (0 to 100): 50                                  ◄ INPUT

-------------------------------------------------
Is the tank pressurized (for example, with nitrogen)? (Y/N): n   ◄ INPUT

-------------------------------------------------
Is the tank vented? (Y/N): n                                     ◄ INPUT

-------------------------------------------------
Enter the outside temperature (degrees F): 80                    ◄ INPUT

-------------------------------------------------
Enter the temperature (degrees F) of the material in the tank: 80  ◄ INPUT
```

Output Summary

```
-----------------------------------------------------------
Ammonia.............NH3

   Tank size:    1300.00 pounds     (height = 10.0 ft,  diameter =  1.8 ft)
   Orientation:  - - - - Tank is vertical (upright).
   The tank is     50 %  full.
   Start with:    650    pounds.
   Hole size:       0.50 inch diameter.
   Break is at     25 %  of the tank height.
   Ambient temp:   80 F.
   Tank temp:      80 F.

 The liquid   loss is    325.00 lbs in            0.82 minutes.
 (18 % of the liquid,     59.45 lbs, flashes quickly.)
 The vapor    loss is     58.75 lbs in the next   7.55 minutes.
 The vapor    loss is      0.70 lbs in the next   1.78 minutes.
 The total loss is       384.45 lbs in           10.15 minutes.

 Mass is still being lost at a rate of   0.0163 lbs per minute.
 due to heat transfer into the tank.
-----------------------------------------------------------
```

Attachment 4

LEAK.OUT File for Example in Attachment 3

```
-----------------------------------------------------------
Ammonia.............NH3

MolWt    cp     Hvap      A       B       C       SG      TB
17.00    1.10  589.00   16.95 2132.50  -32.98    0.82  -28.50


   Tank size:     1300.00 pounds     (height = 10.0 ft,  diameter =  1.8 ft)
   Orientation:  - - - - Tank is vertical (upright).
   The tank is     50 %  full.
   Start with:    650    pounds.
   Hole size:       0.50 inch diameter.
   Break is at     25 %  of the tank height.
   Ambient temp:   80 F.
   Tank temp:      80 F.

 time    temperature   pressure    liquid loss   mass lost
h: m: s     (F)         (psia)     (lb/min)       (lb)
 -----  -----------   --------   -----------   ---------
0: 0: 0      80.0        149.3         0.0          0.0
0: 0:10      80.0        149.3       413.9         69.0
0: 0:20      80.0        149.3       413.6        138.0
0: 0:30      80.0        149.3       413.3        206.9
0: 0:40      80.0        149.3       413.0        275.7
0: 0:49      80.0        149.3       412.8        325.0

 The liquid level reaches the hole.
 The liquid loss is   325.0 lbs in    0.82 minutes.

 time    temperature   pressure    vapor loss    mass lost
h: m: s     (F)         (psia)     (lb/min)       (lb)
 -----  -----------   --------   -----------   ---------
0: 0:50      78.9        149.3       38.2599      325.6
0: 1: 0      69.3        126.4       32.3481      331.5
0: 1:10      60.9        108.7       27.7572      336.4
0: 1:20      53.5         94.9       24.1147      340.7
0: 1:30      47.0         83.8       21.1712      344.4
0: 1:40      41.2         74.7       18.7544      347.7
0: 1:50      36.0         67.3       16.7423      350.7
0: 2: 0      31.3         61.1       15.0468      353.3
0: 2:10      27.0         55.8       13.6025      355.7
0: 2:20      23.1         51.3       12.3603      357.8
0: 2:30      19.5         47.4       11.2826      359.8
0: 2:40      16.1         44.1       10.3404      361.6
0: 2:50      13.1         41.2        9.5108      363.2
0: 3: 0      10.2         38.6        8.7755      364.7
0: 3:10       7.6         36.3        8.1200      366.1
0: 3:20       5.2         34.3        7.5324      367.4
0: 3:30       2.9         32.5        7.0030      368.6
0: 3:40       0.7         30.9        6.5238      369.8
0: 3:50      -1.3         29.4        6.0881      370.8
0: 4: 0      -3.1         28.1        5.6903      371.8
0: 4:10      -4.9         26.9        5.3258      372.7
0: 4:20      -6.5         25.8        4.9905      373.6
0: 4:30      -8.1         24.8        4.6809      374.4
0: 4:40      -9.6         23.9        4.3943      375.1
0: 4:50     -10.9         23.1        4.1280      375.8
0: 5: 0     -12.2         22.4        3.8799      376.5
0: 5:10     -13.5         21.7        3.6481      377.1
0: 5:20     -14.6         21.0        3.4308      377.7
0: 5:30     -15.7         20.5        3.2266      378.3
0: 5:40     -16.7         19.9        3.0343      378.8
0: 5:50     -17.7         19.4        2.8527      379.3
0: 6: 0     -18.6         19.0        2.6807      379.7
```

0: 6:10	-19.4	18.5	2.5175	380.2
0: 6:20	-20.2	18.1	2.3623	380.6
0: 6:30	-21.0	17.8	2.2142	380.9
0: 6:40	-21.7	17.4	2.0727	381.3
0: 6:50	-22.3	17.1	1.9372	381.6
0: 7: 0	-23.0	16.9	1.8071	381.9
0: 7:10	-23.5	16.6	1.6820	382.2
0: 7:20	-24.1	16.4	1.5613	382.5
0: 7:30	-24.6	16.1	1.4448	382.7
0: 7:40	-25.0	15.9	1.3319	383.0
0: 7:50	-25.4	15.7	1.2224	383.2
0: 8: 0	-25.8	15.6	1.1159	383.4
0: 8:10	-26.2	15.4	1.0122	383.6
0: 8:20	-26.5	15.3	0.9110	383.7
0: 8:30	-26.8	15.2	0.8119	383.9
0: 8:40	-27.0	15.1	0.7149	384.0
0: 8:50	-27.2	15.0	0.6197	384.1
0: 9: 0	-27.4	14.9	0.5262	384.2
0: 9:10	-27.6	14.8	0.4341	384.3
0: 9:20	-27.7	14.8	0.3436	384.3
0: 9:30	-27.8	14.7	0.2546	384.4
0: 9:40	-27.8	14.7	0.1678	384.4
0: 9:50	-27.9	14.7	0.0852	384.4
0:10: 0	-27.9	14.7	0.0221	384.4

Summary

```
The liquid   loss is    325.00 lbs in             0.82 minutes.
(18 % of the liquid,     59.45 lbs, flashes quickly.)
The vapor    loss is     58.75 lbs in the next    7.55 minutes.
The vapor    loss is      0.70 lbs in the next    1.78 minutes.
The total loss is       384.45 lbs in            10.15 minutes.

Mass is still being lost at a rate of   0.0163 lbs per minute
due to heat transfer into the tank.
```

INTERNAL DISTRIBUTION

1. J. C. Anderson
2. M. B. Andriulli
3. J. J. Angelelli
4. T. A. Angelelli
5. G. S. Bass
6. J. P. Belk
7. S. G. Bloom
8. J. S. Bullock, IV
9. M. S. Childers
10-19. J. H. Clinton
20. P. R. Cotten
21. W. K. Crowley
22. F. E. Denny
23. C. K. Ford
24. J. D. Gass
25. M. A. Harrison
26. H. F. Hartman, Jr.
27. W. A. Hartman
28. W. A. Heineken
29-30. J. A. Hoffmeister
31. W. P. Huxtable
32-36. D. J. Inman
37. J. W. Insalaco
38-39. R. A. Just
40. K. D. Keith, Jr.
41. M. W. Knazovich
42. M. G. Kreger
43. A. F. Love
44. R. S. McKeehan
45. L. R. Moore
46. L. W. Nelms
47. R. D. Nipper
48. G. L. Pfennigwerth
49-50. D. G. Renfro
51. T. L. Ryan
52. R. D. Sharp
53. R. B. Smith
54. W. C. T. Stoddart
55. K. M. Warnock
56. C. C. Watson
57-58. B. K. Williams
59-68. W. R. Williams
69. D. W. Wilson
70. S. R. Wilson
71. Central Research Library
72. Laboratory Records
73. Laboratory Records-ORNL RC
74. ORNL Patent Section

EXTERNAL DISTRIBUTION

U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831.

75. Office of Assistant Manager for Energy Research and Development
76. M. A. Boyd
77. R. M. Devault
78. N. Karne

79. W. S. Gregory, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545.

80. M. Y. Ballinger, Pacific Northwest Laboratories, P.O. Box 999, MS K6-55, Richland, WA 99352.

81. J. Mishima, Science Applications International Corp., 1845 Terminal Dr., Richland, WA 99352.

82. J. C. Dean, Martin Marietta Utility Services, Inc., Paducah Gaseous Diffusion Plant, P.O. Box 1410, Paducah, KY 42201.

83. C. E. Gamm, Martin Marietta Utility Services, Inc., Portsmouth Gaseous Diffusion Plant, P.O. Box 628, Piketon, OH 45661.

84. C. B. Sawyer, U.S. Nuclear Regulatory Commission, Washington, DC 20555.

85-86. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831.