UNITED STATES DEPARTMENT OF THE INTERIOR

U. S. GEOLOGICAL SURVEY

MacH2O: A Computer Interface to Calculate the Thermodynamic and Transport Properties of Pure Water

by

Robert J. Rosenbauer[1]

Open-File Report 91-366-A

[1]Menlo Park, CA 94025

1991

Open-File Report 91-366-A, B, and C. MacH_2O: a computer interface to calculate the thermodynamic and transport properties of pure water. 1991. 99p.

The program computes the thermodynamic and physical-transport properties of pure water in the vapor and liquid states over a wide range of temperature and pressure. It has been adapted from the the computer code of Haar, Gallagher, and Kell (1984) to which additional parameters have been added. The program is designed to operate on an Apple®-Macintosh™ computer and contains a complete graphical-user interface.

Requirements: Apple®-Macintosh™ computer; minimum 512K RAM; math coprocessor. OF91-366-A, Documentation, 22p., paper copy or diskette; OF91-366-B, Executable diskette; OF91-366-C, Source code, 77p. paper copy or diskette. All diskettes are $3^1/_2$", double-sided format.

MacH2O: A Computer Interface to Calculate the Thermodynamic and Transport Properties of Pure Water

Robert J. Rosenbauer
U.S. Geological Survey
Menlo Park, CA 94025

ABSTRACT

The program MacH2O is a computer code that calculates thermodynamic and physical transport properties of pure water over a wide range of temperature and pressure. This program is an extension of the equations and code originally developed by Haar, Gallagher and Kell (1984) and is designed specifically for the Apple®-Macintosh™ personal computer for which a complete user interface has been developed.

INTRODUCTION

MacH2O is a FORTRAN77 computer program that calculates thermodynamic and transport properties for the vapor and liquid states of water. It has been adapted from the the computer code of Haar, Gallagher, and Kell (1984) to which additional parameters have been added and mated to a Macintosh personal computer. This report reviews some of the theoretical basis for the calculations, provides an outline of how the program is structured, and documents the actual use of the program through the Apple®-Macintosh™ interface.

THEORY

The computer code developed by Haar, Gallagher, and Kell (1984) was based on their fundamental equation for the specific Helmholtz free energy A for ordinary water and steam as a function of temperature T and density ρ and takes the form:

$$A = A(\rho, T).$$

A single algebraic expression of this equation was adopted by the International Association for the Properties of Steam as the Provisional Formulation 1982 for the Thermodynamic Properties of Ordinary Water Substance and issued as the NBS/NRC Steam Tables. This expression is valid over the temperature range between 0 and 1000°C and pressures from 0 up to a maximum 5000 bar at 0°C and up to 15000 bar above 150°C, exclusive of a region around the critical point within 1°C and 0.3 bar. All thermodynamic quantities are defined in terms of the first and second derivatives of this function. In practice, the equation comprises three parts: a theoretical base function, a multi-term residual function containing corrections to the base function, and an ideal gas function. The combined equation has the form:

$$A(r, T) = A_{base}(r, T) + A_{residual}(r, T) + A_{ideal\ gas}(r, T)$$

Appropriate differentiation of this equation according to the first and second laws of thermodynamics provides all the thermodynamic properties. First derivatives of this expression at constant temperature or density yield pressure, specific internal energy, and specific entropy. Values for specific enthalpy and specific Gibbs free energy follow from their individual definitions. Specific heat capacities result from second derivatives of the fundamental equation with respect to internal temperature at constant volume and enthalpy at constant pressure. Additional parameters such as the speed of sound and Joule-Thomson coefficients follow from further differentiation. The individual form of each of these derivatives is given in Haar et al. (1984) and Kestin and Sengers (1986). Haar et al. (1984) also provides an

expression for viscosity and thermal conductivity as functions of density and temperature.

The original computer code calculated either pressure or density and enthalpy, entropy, heat capacities, Joule-Thompson coefficients and the velocity of sound from the input of temperature along with either pressure or density. This code was updated to include calculations for the Helmholtz and Gibbs free energies, compressibility, viscosity, internal energy and thermal conductivity (Gallagher, pers. com.). The current version presented here provides in addition, calculations of specific volume and quartz solubility. Quartz solubility is calculated as a function of temperature and density according to an empirical expression developed by Fournier (1982). MacH_2O contains updated algorithms and has been extensively re-written in terms of efficiency and user friendliness. It contains a list of user-selectable options for input in a variety of standard units and can perform either single point calculations or can compile tables, all packaged within a typical Macintosh application.

PROGRAM STRUCTURE and SYSTEM REQUIREMENTS

The program MacH_2O consists of three related parts: call statements to the tool-box subroutines of the Macintosh graphical interface (Appendix A); the computational code interspersed with additional tool-box call statements (Appendix B); and the interface resource file (Appendix C). The computational part of the program is standard FORTRAN77 code that consists of multiple subroutines to evaluate the fundamental function and its derivatives. Most of these routines have been carried over intact from the last updated version of the code from Haar et al. (1984) except where increases in efficiency could be obtained or expansion was necessary to accommodate calculations for additional properties and the Macintosh interface. This part of the code also calculates transport, geochemical, and other thermophysical properties that do not derive directly from the base equation. The main program contains standard FORTRAN77 code and calls to the Macintosh tool-box through the

MacFortran™ toolbox subroutine interface whose function is to handle activity between user input and the computational part of the program and direct file management. The resource file, *Macwater.R* provides bitmap data to the interface and includes and launches the main application.

The FORTRAN code was compiled using an Absoft MacFortran/020 compiler, version 2.4. Resources to the Macintosh *toolbx* interface were compiled using RMAKER, also provided by Absoft. The source files take 190,104 bytes of storage and the executable module uses 128,237 bytes. The suggested application memory size is 1200K.

Three files constitute this release: 1) *MacH2O* is the executable module; this is the only file actually required to run the program. *MacH2O* is created by RMAKER and includes the intermediate application *Macwat* created by the MacFortran/020 compiler. *MacH2O* is also directly linked to the run-time libraries, *f77.rl* and *m81.rl* and the toolbox subroutine, *toolbx.sub*. 2) *Macwat.f* is the FORTRAN77 source code. 3) *Macwater.R* is the resource source file. Two other files may appear but are transparent to the user. A *default* file is created automatically whenever the user changes the program default values and a temporary internal file *outdata* is created then deleted to facilitate correct character data representation within dialog boxes.

PROGRAM OPERATION

Single-Point Calculations

The program is formatted to be consistent with most applications written for the Macintosh graphical interface. It is menu driven and data are entered and displayed in a series of modal dialog boxes. Figure 1 shows the output window and the primary menu options when the program is initialized. Choosing 'run' under the 'Program' menu opens an input dialog box (Fig. 2). Temperature and pressure are the default inputs and degrees centigrade and bars

are the default units. A variety of other units for temperature, pressure, density, and energy can be selected by the user under the 'Units' menu. Figure 3 shows the list of options within each category. Default units for density are g/cm^3 and for energy are j/g. Alternate units are selected by clicking the appropriate radiobutton within each dialog box. These options are also accessible through the command key and the first letter of the option. For example, the sequence 'command key t' (⌘ t) displays the list of temperature options. New default values for input units are created by selecting the **New Default** button. The **New Default** button creates an external file 'default' with the new values. The program automatically searches the desktop for the existence of this file to use in lieu the defaults built into the program code. All variable names and units are monitored throughout the program and displayed when appropriate for prompting purposes or as output.

Various combinations of temperature, pressure, density, enthalpy and entropy are available as inputs and selected by the user under the 'Input Options' menu (Fig. 4). Results of the program calculations have been arbitrarily split between those parameters commonly determined by this user and those parameters less frequently utilized. The primary results are defined as temperature, pressure, density, specific volume, and quartz solubility. These results are displayed immediately in an output dialog box. (Fig. 5). In addition, for pressure and temperature conditions below the critical point of pure water, the vapor pressure and the saturated liquid and vapor densities are displayed in the output window. All other output is accessible through the **Expand** button. The **Expand** button causes a separate dialog box to appear that contains the remaining output. The **Return** button within this dialog box allows the user to cycle between the primary and expanded output to review the results. Every result is accompanied by its appropriate unit. The entire contents of the expanded output dialog box can be printed directly to either a laser printer or an image writer via the **Print** command. The **Continue** button returns the user to the data input mode for additional single point calculations. The user may exit the program via any of the **Quit** buttons, the window close box, or 'Quit' under the

file menu. Access to menu selections is always available to the user through the **Menu** button contained within each dialog box.

Table Calculations

Tables of output can be constructed by selecting the 'table' option under the 'Format' menu. A list of the possible table types is displayed in figure 6; this dialog box appears whenever the table format is selected. The default table type is an isothermal table with pressure as the independent variable. Whenever the table option has been selected, 'run' under the 'Program' menu opens the dependent variable dialog box. Up to three dependent variables can be user selected from the list shown in figure 7. For most table calculations the user must designate the initial, final, and incremental values of the independent variable (eq. figs. 8, 8b). Figures 9 and 9a show typical output from tabulated calculations. Note that table output along the liquid-vapor saturation curve contain data for both liquid and vapor phases. All tables under a page in length can be printed from the 'Print' command under the file menu. Tables that are greater than one page can be printed as the pages appear on the monitor.

A brief description of the program is found under the 'Apple' menu in the 'About MacH_2O' item. Some details of the program operation are available to the user under the 'Help' menu.

ACKNOWLEDGMENT

I thank John Gallagher for freely providing a copy of his latest unpublished version of the program code listed in Haar et. al. (1984). I also thank James Bischoff and Graig McHendrie for thoughtful reviews and suggestions concerning the text and program code.

REFERENCES

Fournier, R. O., 1982, A Method of Calculating Quartz Solubilities in Aqueous Sodium Chloride Solutions, Geochim. Cosmochim. Acta., Vol. 47, pp. 579-586.

Harr, L., Gallagher, J. S., and Kell, G. S., 1984, NBS/NRC STEAM TABLES, Washington, Hemisphere Pub. Corp., 320 p.

Kestin, J. and Sengers, J. V., 1986, New International Formulations for the Thermodynamic Properties of Light and Heavy Water, Jour. Phys. and Chem. Ref Data, Vol. 15, No. 1, pp. 305-320.

Rose, C., 1985, Apple® Inside Macintosh™, pub. Addison-Wesley Publishing Co. Inc., Reading, Massachusetts, Vol. I - V.

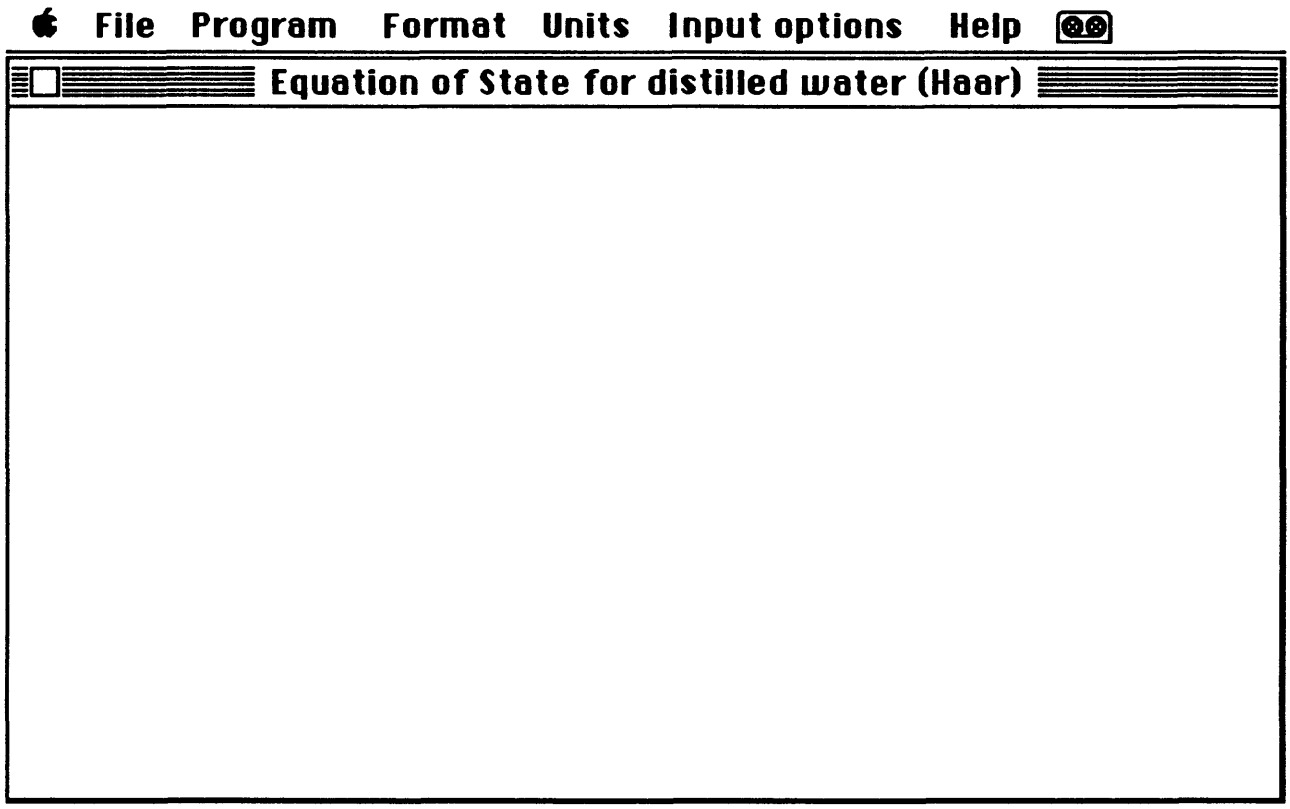

Figure 1 - Active Macintosh™ window with menu headings, opened when program MacH_2O is first initialized. This window can be moved around the desktop and de- and re-activated. Single clicking the mouse in the close box in the upper left corner of the window will close the application as will choosing 'Quit' from the file menu. Both the Apple menu and MacroMaker are accessible during program operation.

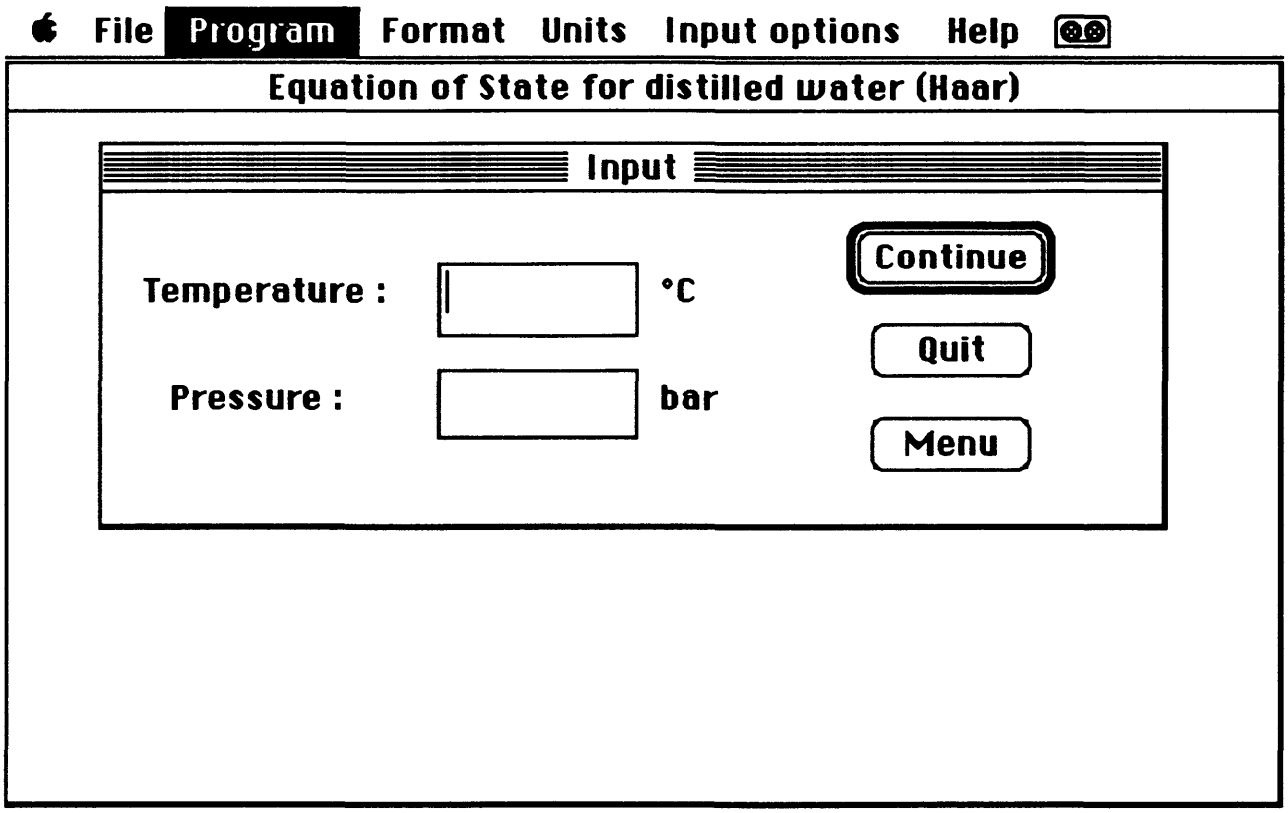

Figure 2a - Window containing modal dialog box for entering data into program to make calculations in the single phase. This window appears when 'run' is selected under the 'Program' menu. It displays the variable names and units that were selected under the input options and the units menus. Data are entered and edited in standard Macintosh™ textedit control boxes. The tab key on the keyboard scrolls between the edit boxes. The **Continue** control button is selected to load the data into the program. The **Quit** button causes execution to stop and closes the program and the **Menu** button closes the input dialog box and returns the user to the main menu selections.

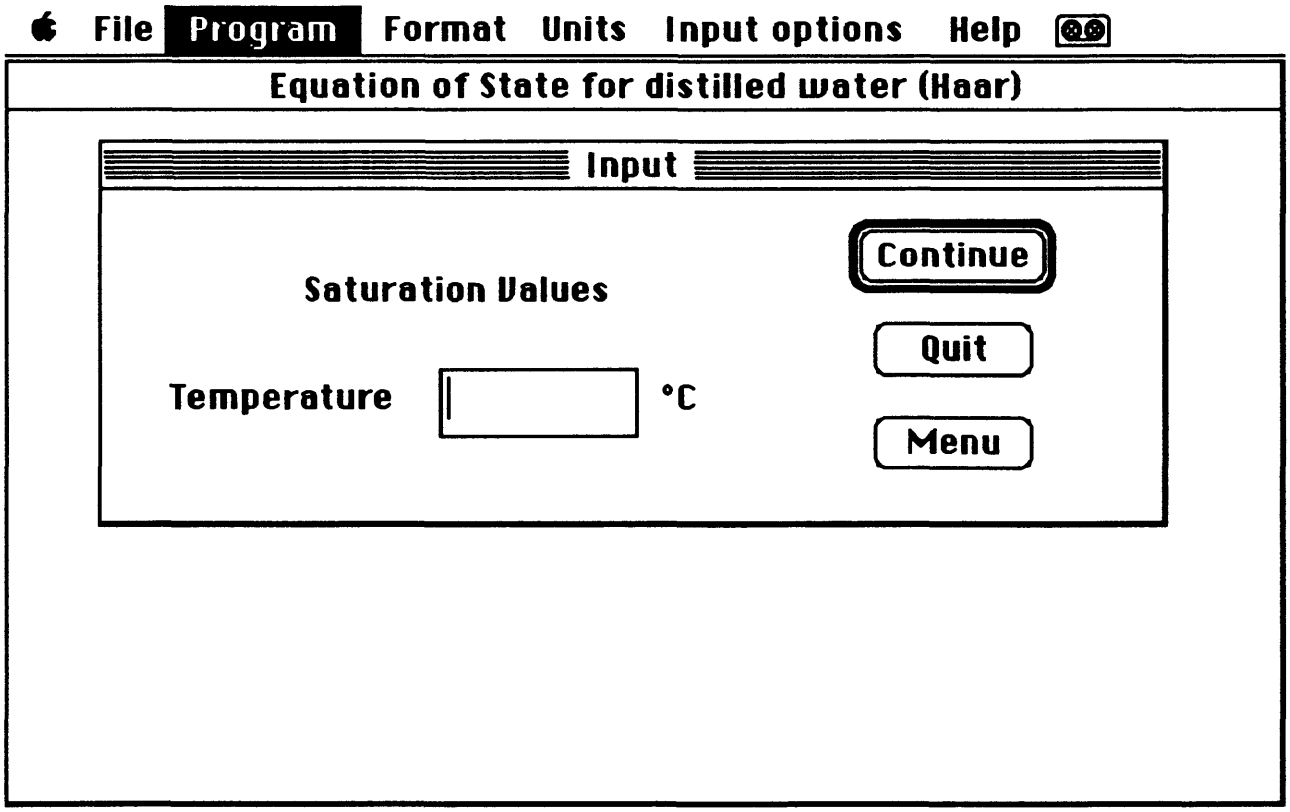

Figure 2b - Input dialog box identical to figure 2a except is displayed to enter data along the saturation curve in either temperature or pressure.

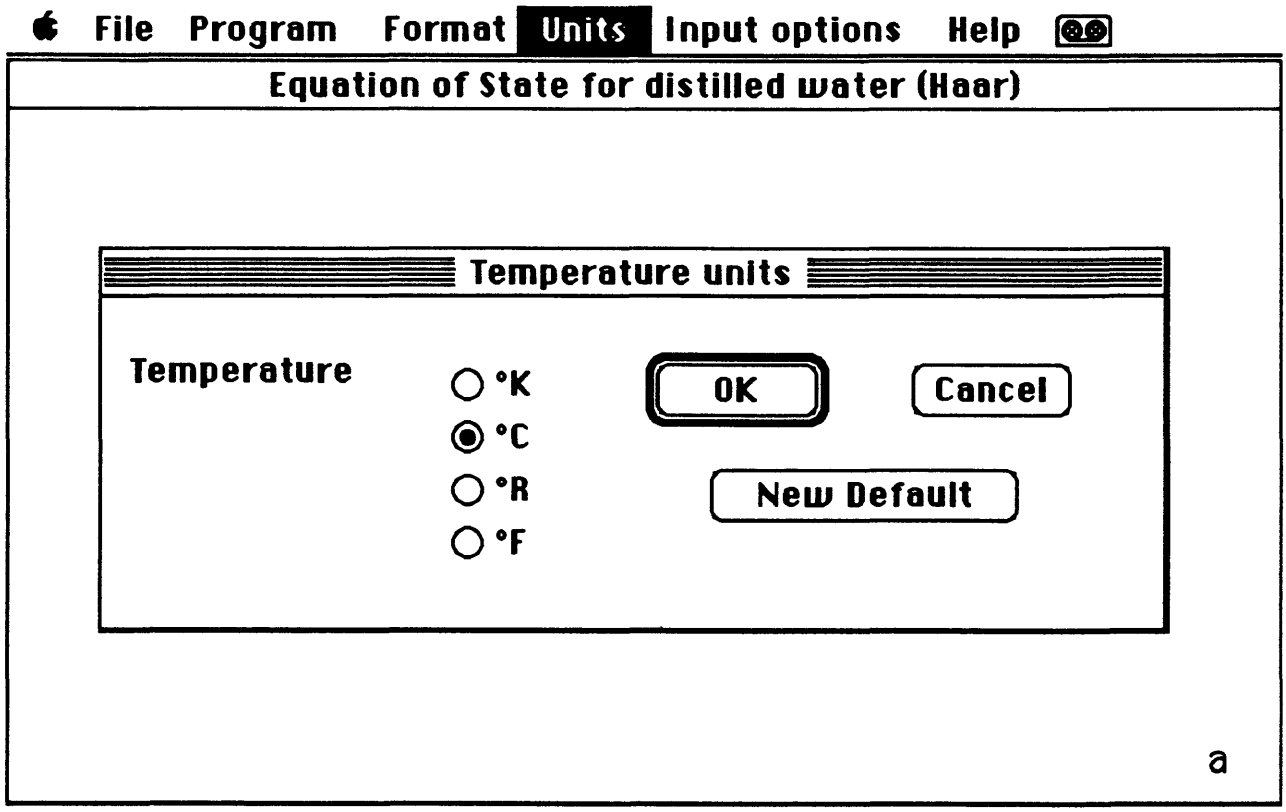

Figure 3a - Dialog box containing user selectable options for units of temperature which appears as an item under the 'Units' menu. The program defaults to °C unless the **new default** button is selected. The **OK** button accepts any new unit selection until the application is closed. The **cancel** button ignores any new selection.

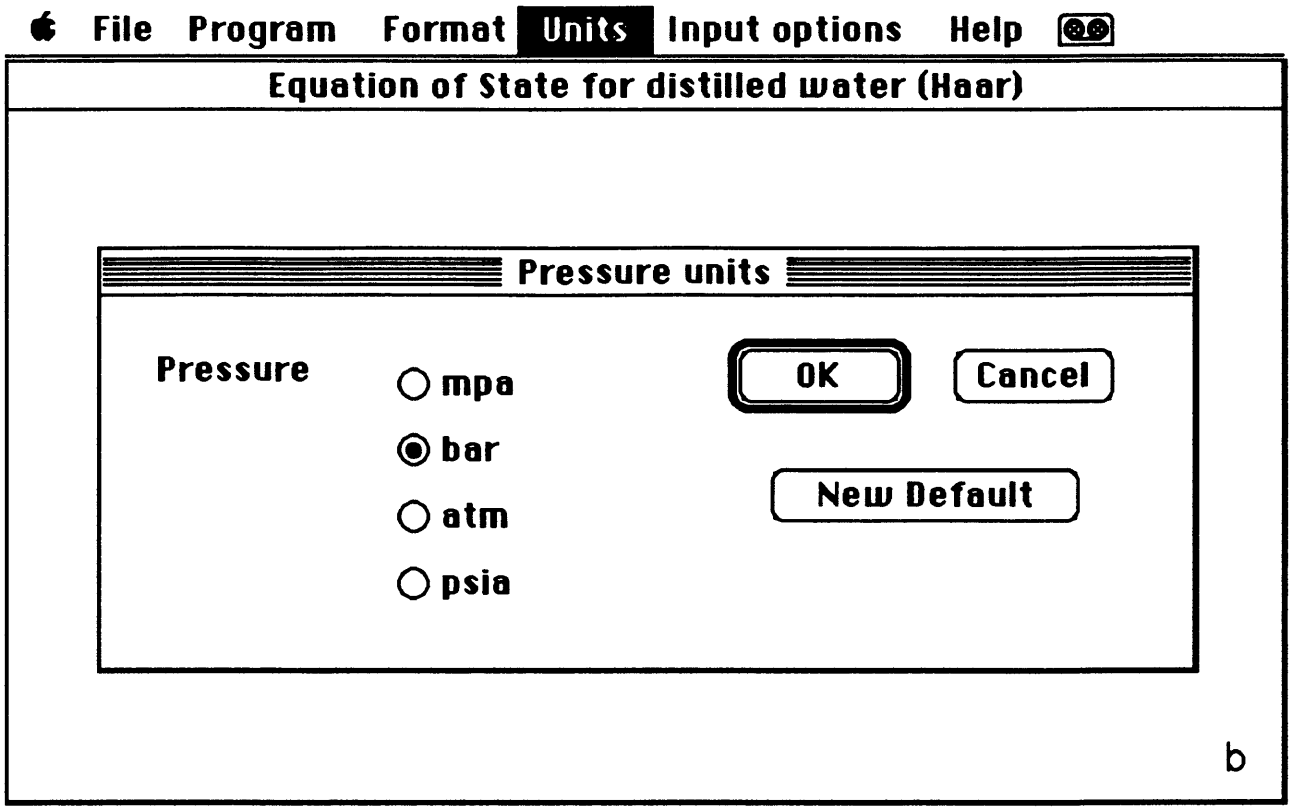

Figure 3b - Dialog box containing user selectable options for units of pressure which appears as an item under the 'Units' menu. The program defaults to bar unless the **new default** button is selected. The **OK** button accepts any new unit selection until the application is closed. The **cancel** button ignores any new selection.

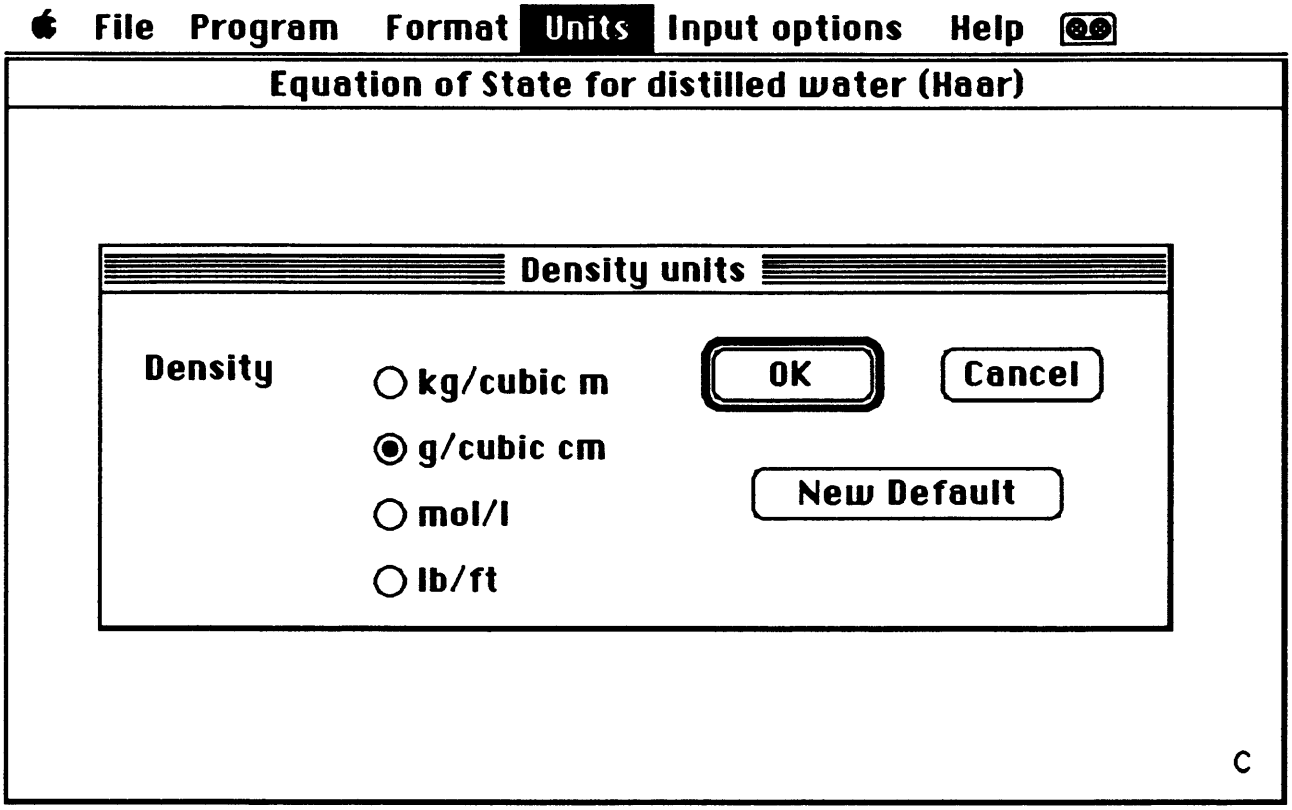

Figure 3c - Dialog box containing user selectable options for units of density which appears as an item under the 'Units' menu. The program defaults to g/cm^3 unless the **new default** button is selected. The **OK** button accepts any new unit selection until the application is closed. The **cancel** button ignores any new selection.

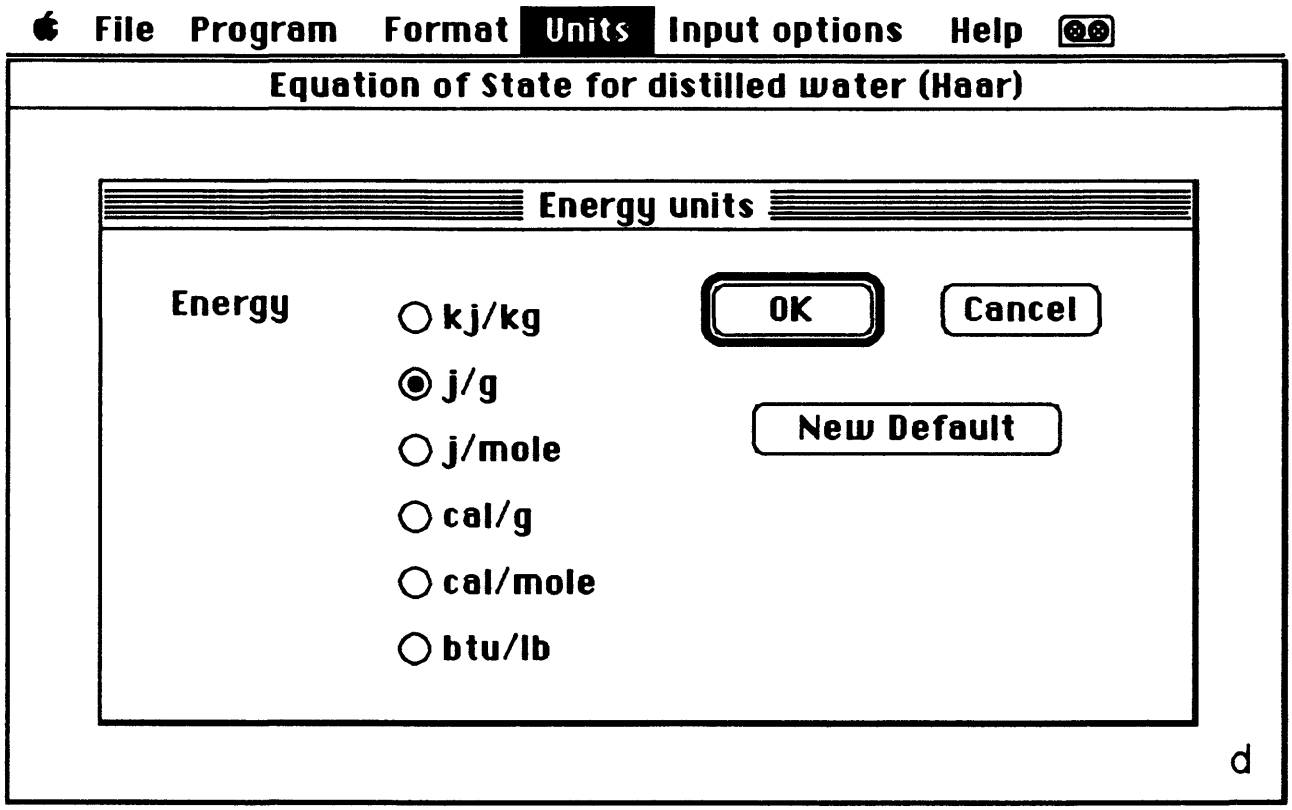

Figure 3d - Dialog box containing user selectable options for units of energy which appears as an item under the 'Units' menu. The program defaults to j/g unless the **new default** button is selected. The **OK** button accepts any new unit selection until the application is closed. The **cancel** button ignores any new selection.

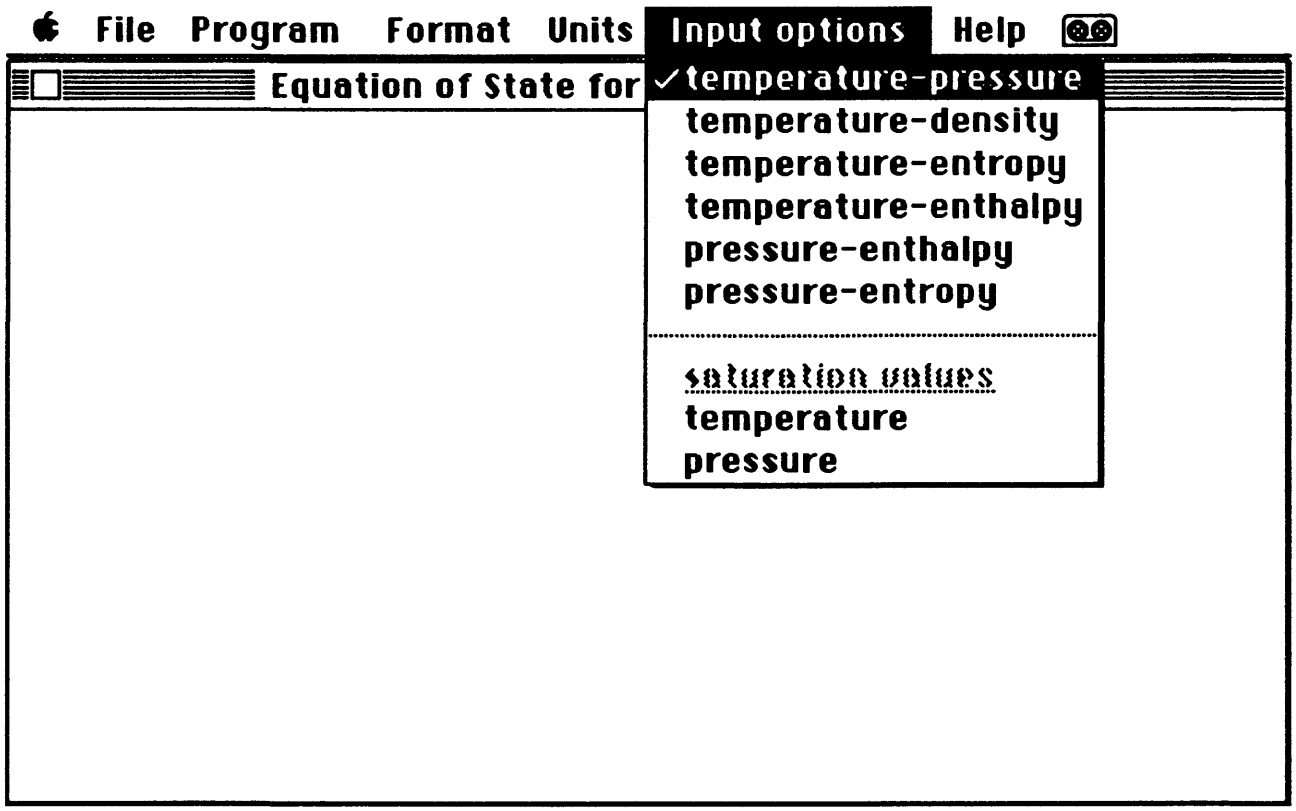

Figure 4 - Item list appearing under the 'Input options' menu. User selectable pairs of input in the vapor or liquid state of pure water or individual temperature or pressure input along the saturation curve of pure water. Temperature and pressure are the default inputs. Any new selection remains in effect for the duration of all single point calculations.

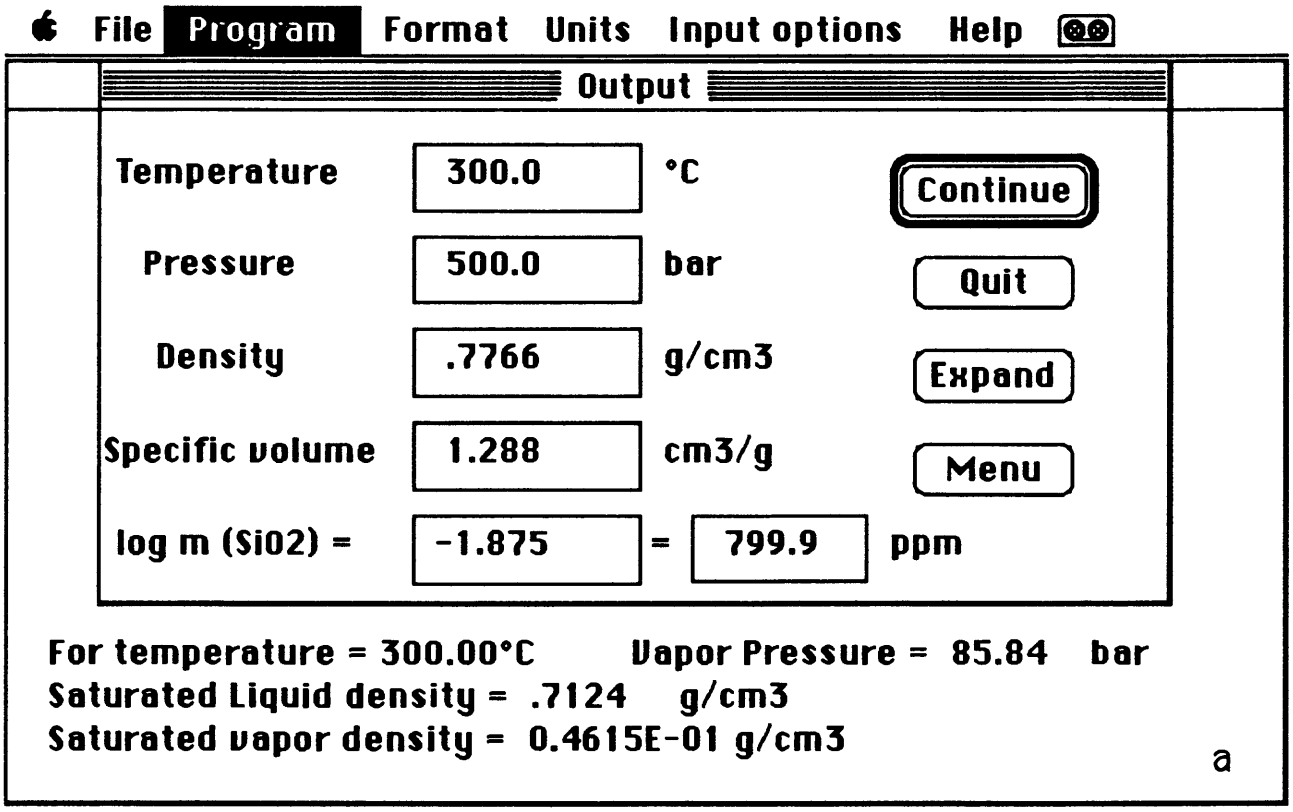

Figure 5a - Dialog box containing the primary output from single point calculations. The **Continue** button returns the user to the data input mode for a new calculation of the same type. The **Quit** button exits the program and closes the application. The **Expand** button displays additional output in a separate output dialog box and the **Menu** button returns the user to the main menu list to change options.

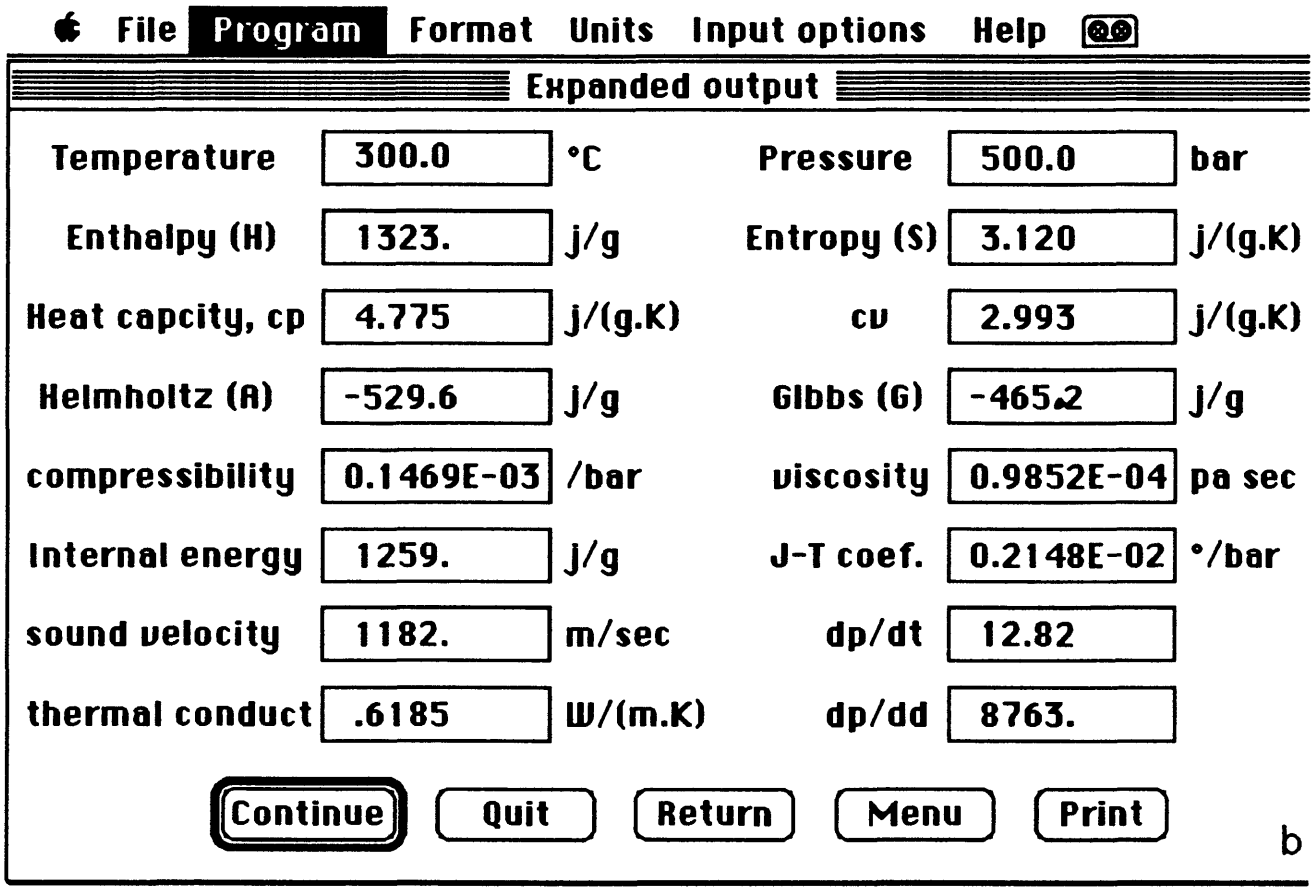

Figure 5b - Dialog box containing expanded output for single point calculations. The **Continue** button returns to the input mode; the **Quit** button closes the application; the **Return** button cycles back to display the primary output; the **Menu** button returns the user to the menu options and the **Print** button prints a hard copy of the output to either an imagewriter or a laser printer.

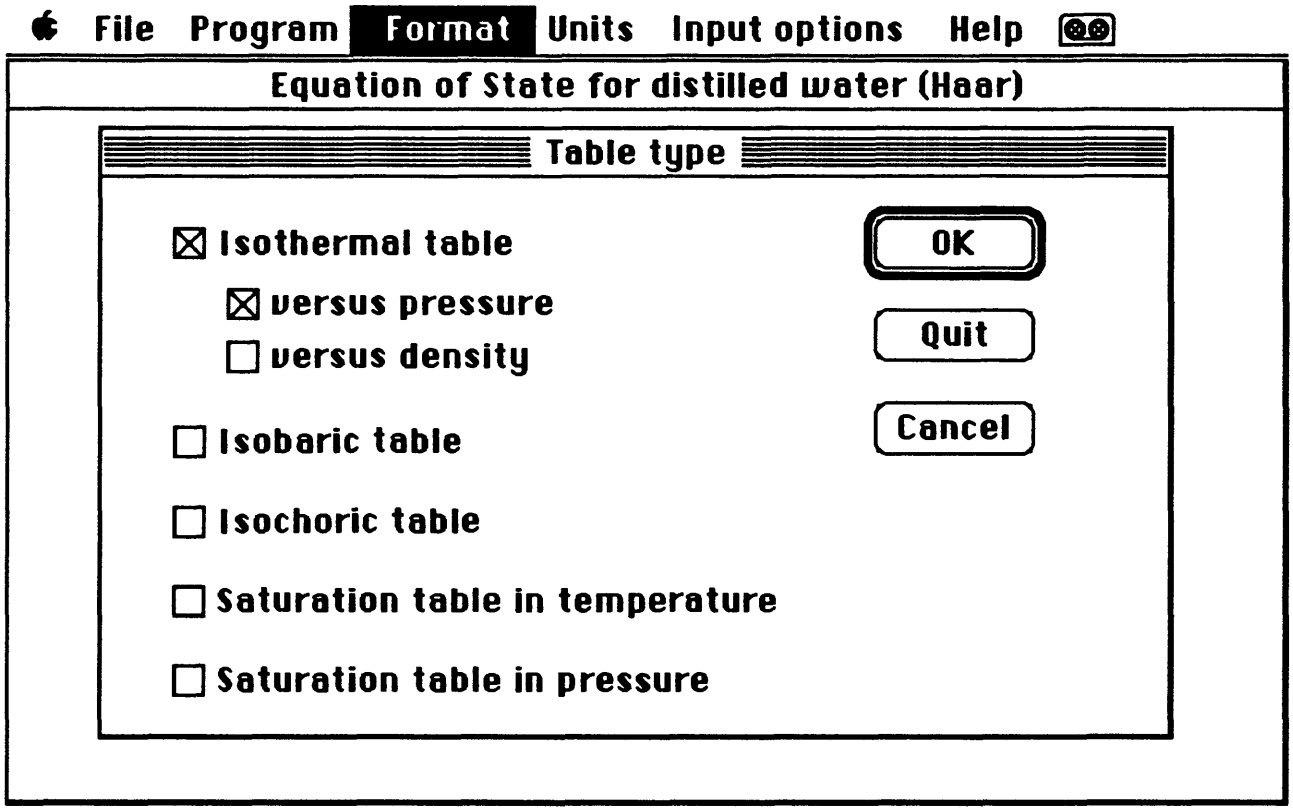

Figure 6 - Dialog box containing the various types of tables that the program can construct. These selections are displayed by the 'table' option under the 'Format' menu. The default table type indicated by the checked boxes is an isothermal table versus pressure. The **OK** button registers the table-type selection and returns the user to the menu level. The **Quit** button closes the application and the **Cancel** button ignores any change of selection and returns the user to the menu level.

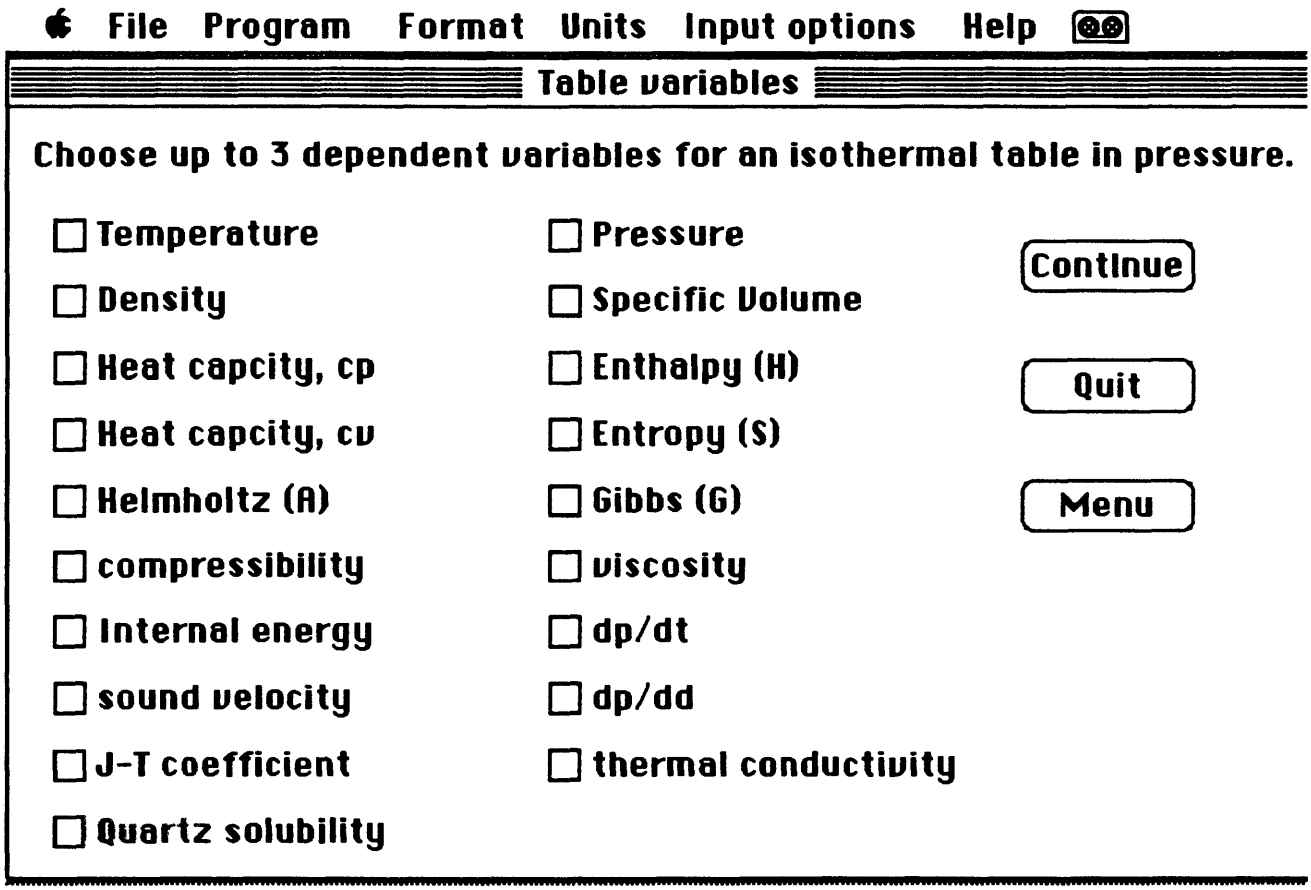

Figure 7 - Dialog box containing a list of user-selectable dependent variables to be tabulated. The user may select up to three variables in addition to the independent variables for any single table. The **Continue** button opens an input dialog box; the **Quit** button closes the application and the **Menu** button returns the user to the menu level.

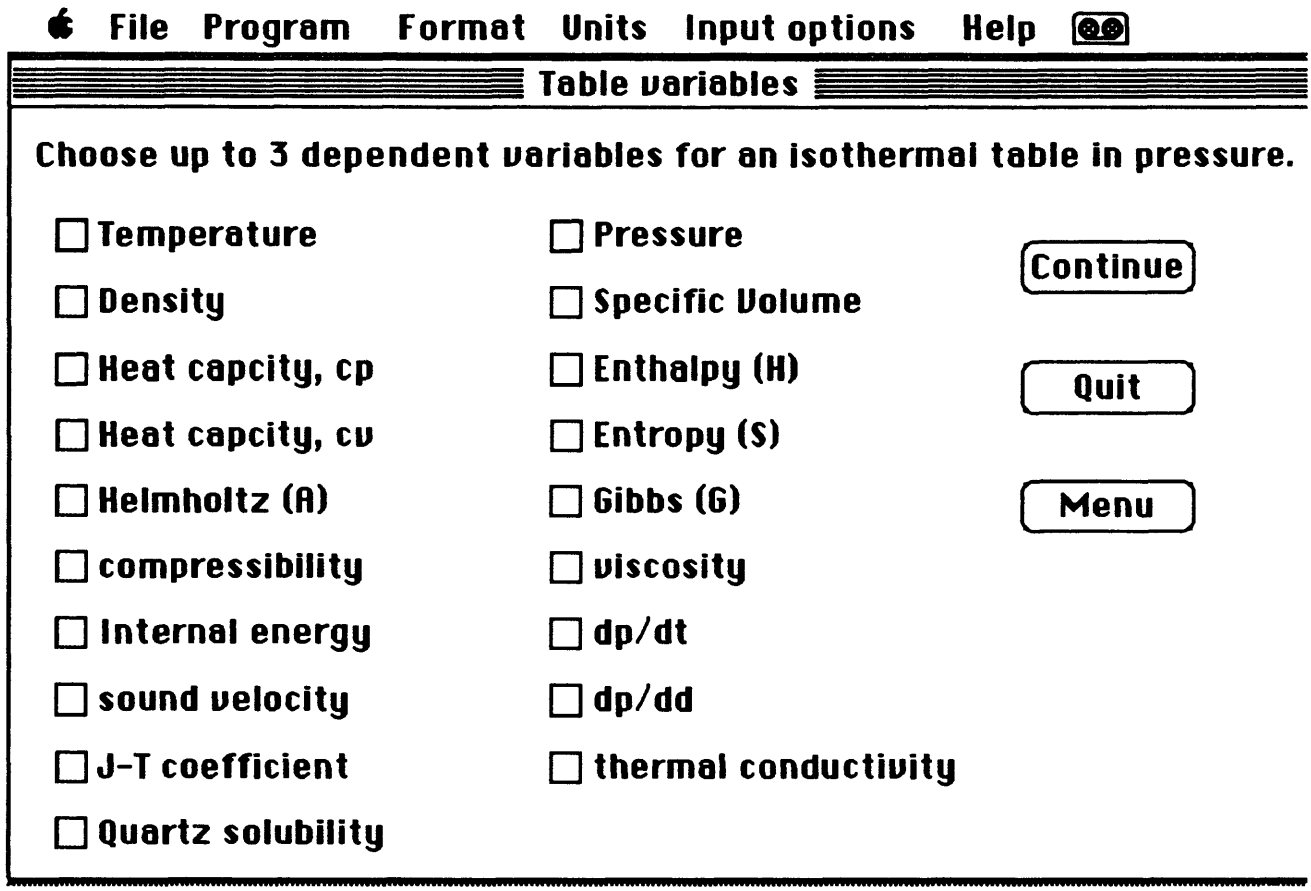

Figure 7 - Dialog box containing a list of user-selectable dependent variables to be tabulated. The user may select up to three variables in addition to the independent variables for any single table. The **Continue** button opens an input dialog box; the **Quit** button closes the application and the **Menu** button returns the user to the menu level.

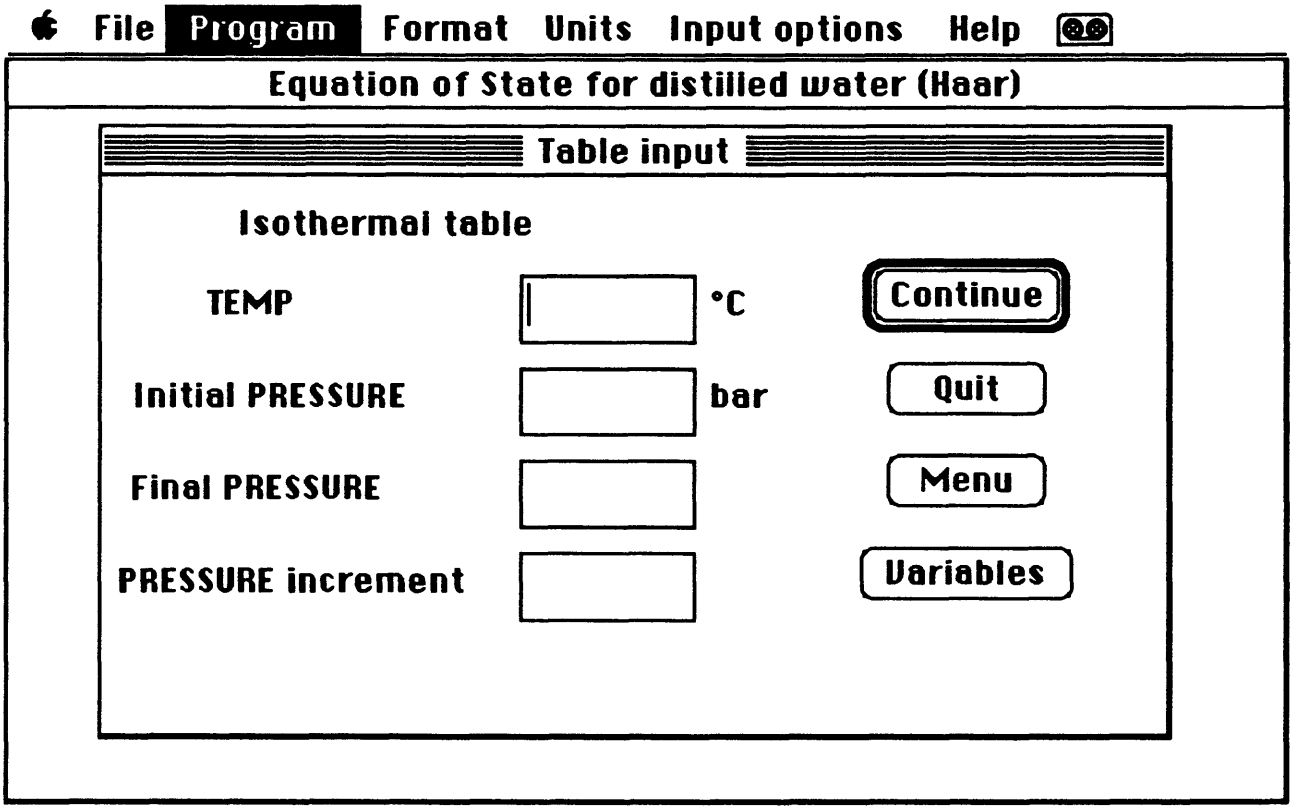

Figure 8a - Dialog box containing typical input required to construct a table in the single phase region. The **Continue** button creates the table; the **Quit** button closes the application; the **Menu** button returns the user to the menu level and the **Variables** button returns the user the list of dependent variables. The tab key on the keyboard scrolls through the input boxes.

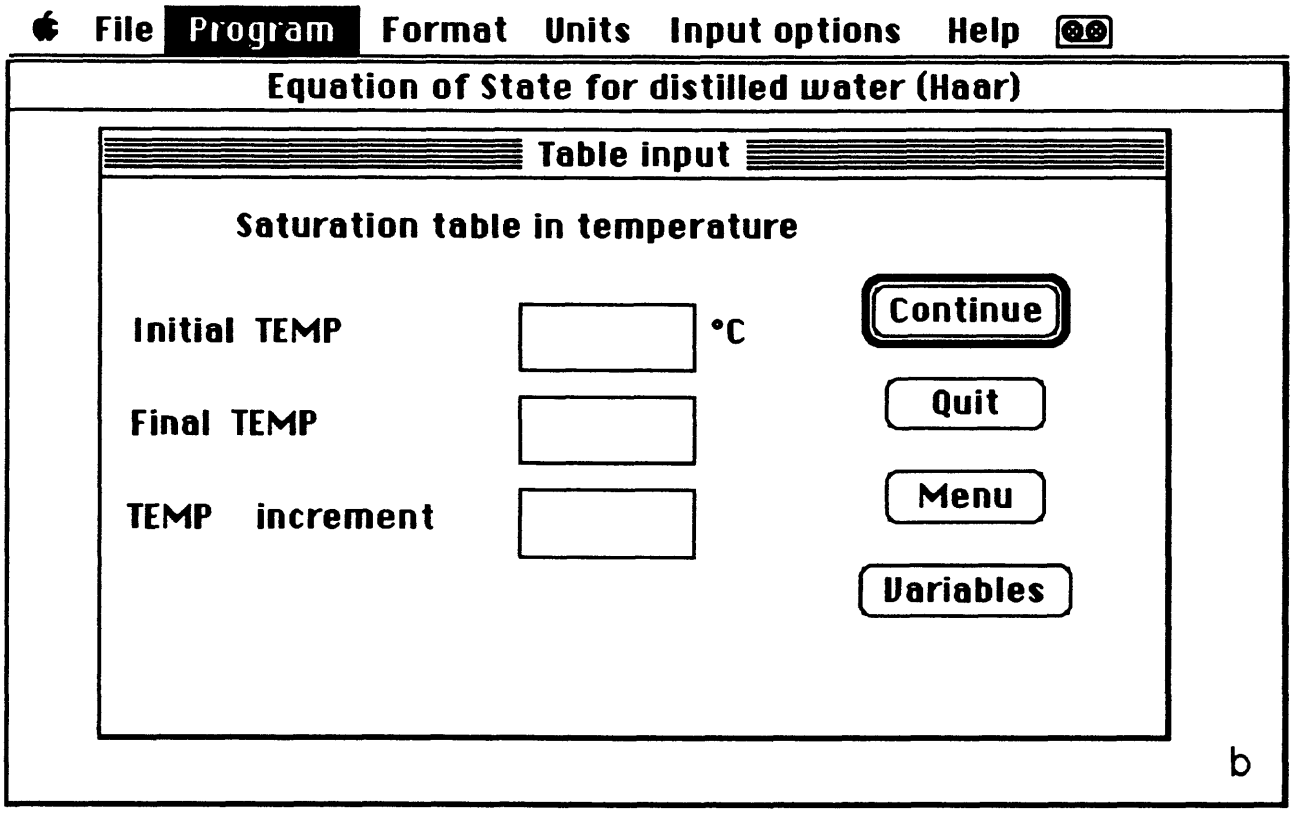

Figure 8b - Dialog box containing input required to construct a saturation table in temperature. The **Continue** button creates the table; the **Quit** button closes the application; the **Menu** button returns the user to the menu level and the **Variables** button returns the user the list of dependent variables. The tab key on the keyboard scrolls through the input boxes.

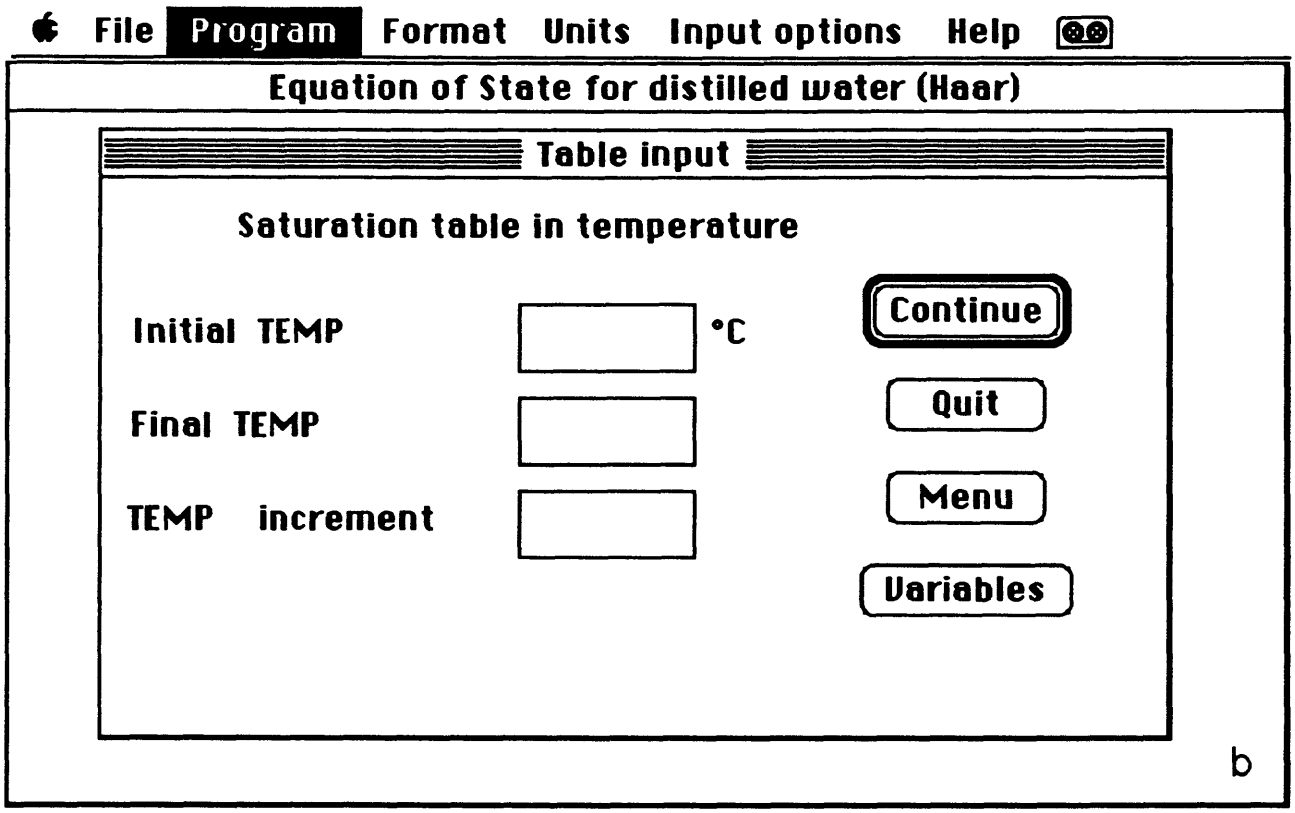

Figure 8b - Dialog box containing input required to construct a saturation table in temperature. The **Continue** button creates the table; the **Quit** button closes the application; the **Menu** button returns the user to the menu level and the **Variables** button returns the user the list of dependent variables. The tab key on the keyboard scrolls through the input boxes.

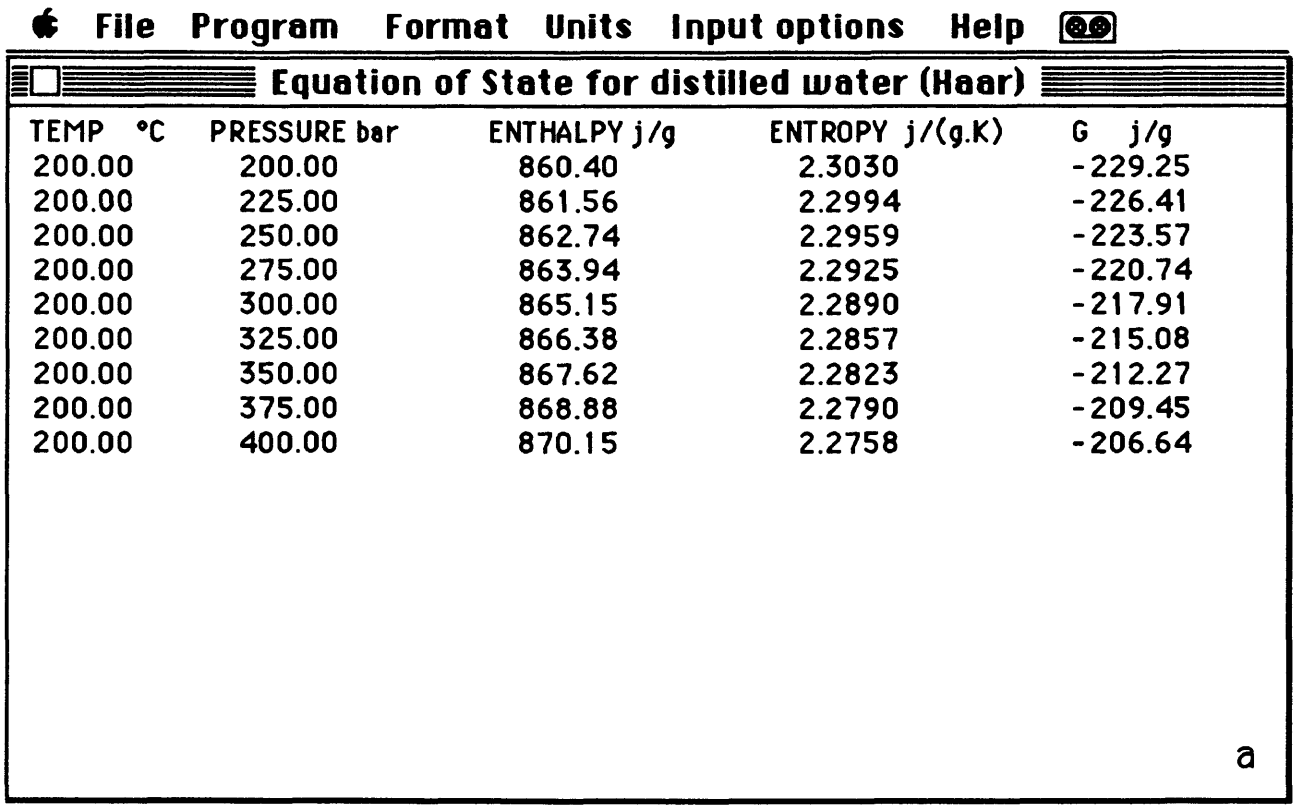

Figure 9a - Example of an isothermal table versus pressure. A hard copy of the table is obtained by selecting 'Print' under the 'File' menu.

File Program Format Units Input options Help

Equation of State for distilled water (Haar)

TEMP °C	PRESSURE bar	VOLUME cm3/g	CV j/(g.K)	CP j/(g.K)
200.00	15.536	1.1564	3.3172	4.4890
200.00	15.536	127.32	1.9227	2.7971
220.00	23.178	1.1900	3.2450	4.6036
220.00	23.178	86.157	2.0583	3.1089
240.00	33.447	1.2292	3.1819	4.7587
240.00	33.447	59.742	2.2175	3.5187
260.00	46.894	1.2758	3.1290	4.9727
260.00	46.894	42.194	2.4014	4.0685
280.00	64.132	1.3324	3.0879	5.2789
280.00	64.132	30.164	2.6118	4.8355

b

Figure 9b - Example of a table along the saturation curve. Data for both the liquid and vapor states are presented in pairs.

```
*                              Appendix A
* MACH2O.F
*
* U.S. Geological Computer Program 'MACH2O'
*
* Use of this program is described in U.S. Geological Survey
* OpenFile Report 91-366 by Robert J Rosenbauer
*
* Program written in fortran77 and MacFortran™ for the 020 compiler;
* Absoft version 2.4
*
* Program was last modified 7/2/91
*
* Program runs on a Macintosh II or SE personal computer
*
* Programmer:
*          Robert J. Rosenbauer
*         U.S. Geological Survey
*         Menlo Park, California
*
* The following section written in MacFortran serves as a
* user interface to the Macintosh tool box and the fortran77
* computational code contained in subroutine 'H2O1'.
*
* This program uses resources defined in the RMaker file Macwater.R
*
* All input and output is from/to the monitor device.
* Some output may be user directed to either a laser printer or image-writer

        program mach2o

        implicit none                    ! Force declaration.


* The following include files contain toolbox parameter definitions.
        include :include files:windoff.inc
        include :include files:grafport.inc
        include :include files:desk.inc
        include :include files:font.inc
        include :include files:event.inc
        include :include files:menu.inc
        include :include files:memory.inc
        include :include files:textedit.inc
        include :include files:window.inc
        include :include files:resource.inc
        include :include files:control.inc
        include :include files:quickdraw.inc
        include :include files:misc.inc
        include :include files:dialog.inc

*************************************************************************
* Constants to set-up menu bar, the applemenu includes all desk accessories  *
* and is always appears in the upper left portion of the menu bar: menu #1.  *
* Specifications for other menus are obtained from the resource file,        *
* (appemdix C) via a resource ID number. Lastmenu = total number of menus    *
*************************************************************************
        integer formatmenu, helpmenu
        integer lastmenu, applemenu, filemenu
```

```
      integer programmenu, unitsmenu, optionsmenu
      parameter (lastmenu = 7, applemenu = 1)        ! Menu numbers
      parameter (helpmenu = 252, formatmenu = 251)   ! Resource ID #'s for menus
      parameter (filemenu = 253, programmenu = 254)
      parameter (unitsmenu = 255, optionsmenu = 256)

* Window coordinates
      integer top, left, bottom, right
      parameter (top = 1, left = 2, bottom = 3, right = 4)
      integer*1 wrecord(windowsize)    ! Window record.
      integer whichwindow              ! Window pointer.

* Window boundary variables
      integer*2 screenrect(4)
      integer*2 dragrect(4)
      integer*2 sizerect(4)
      integer*2 Rect(4)
      integer*2 windrect(4)

* Dialog box parameters
      integer*4 dialogptr       ! Pointer to dialog box
      integer*4 dialogID        ! Resource id of dialog template
      integer*4 ProcPtr         ! Event filter parameter
      integer*4 dStorage        ! Pointer to storage
      integer*4 Handle          ! Handle to item list of dialog
      integer*2 itemType        ! Predefined constant for dialog items
      integer*2 itemHit         ! Handle to individual dialog item
      integer*4 behind          ! Specifies window placement (-1 = in front)

      integer code              ! Mouse location within window
      integer tvalue            ! Checkbox control monitor
      dimension tvalue(25)
      integer value             ! Checkbox control monitor
      dimension value(10)

      logical temp              ! Active event when true
      logical goway             ! Exit program when true (window box checked)
      logical dfile             ! true when default file exists

      character*256 str255      ! character function to create Pascal string
      character*256 string      ! temporary data variable storage

* Variables and constants for increasing the stack size
      integer*4 appllimit             ! Address of current aplication limit
      integer*4 newlimit              ! New application limit
      parameter (appllimit=z'00000130')

      integer*4 color              ! For background color on Mac II.

* Structure returned by MenuSelect
      integer*2 menuitem(2)
      integer*4 menusel
      equivalence (menuitem, menusel)

* Declare the toolbox interface as an integer function.
      integer toolbx

* Global variables.
```

```
      integer mywindow                ! Window pointer
      integer*1 grafport(grafsize)    ! Record for printing environment
      integer grafptr                 ! Pointer to grafport
      integer mymenus(1 : lastmenu)   ! Menu list
      integer it,ip,id,ih             ! unit variables
      integer i,iopt,iopt7,kopt,lopt,
     &  itot,jopt                     ! flags and counters
      logical doneflag                ! Exit event loop if true.
      logical dialogflag              ! Exit dialog box when true
      character*8 nv(4),njt(5),nt,nd,np,nh,ns
      character*8 ns1,ns2,name(20),nunt(20)
      double precision ft,fd,fp,fh
      common mymenus, mywindow, doneflag, grafptr,
     &  windrect, dialogflag
*************************************************************************
* common /flag/                                                         *
* iopt = flag to control input option                                   *
*     iopt = 1; input temperature and pressure  (default                *
*     iopt = 2; input temperature and density                           *
*     iopt = 3; input temperature and entropy                           *
*     iopt = 4; input temperature and enthalpy                          *
*     iopt = 5; input pressure and enthalpy                             *
*     iopt = 6; input pressure and entropy                              *
*     iopt = 7; input temperature (iopt7 =1) or pressure (iopt7 =2) along  *
saturation curve.                                                       *
*                                                                       *
*     kopt = 1 for single point calulation  (default)                   *
*     kopt = 2 for table format                                         *
*                                                                       *
*     lopt = 1, Isothermal table (default)                              *
*               jopt = 2, temperature & pressure input      (default)   *
*          jopt = 3, temperature & density input                        *
*     lopt = 2, Isobaric table                                          *
*     lopt = 3, Isochoric table                                         *
*     lopt = 4, Saturaton table in temperature                          *
*     lopt = 5, Saturaton table in pressure                             *
*                                                                       *
* tvalue = table input control. User may select up to 3 dependant variables *
* (stored in itot) from a list of 19 for a table format.                *
*                                                                       *
*************************************************************************
* common /units/                                                        *
* it = temperature units; ft converts to local temperature units        *
*       it = 1, °K;     it = 2, °C (default)                            *
*       it = 3, °R;     it = 4, °F                                      *
*                                                                       *
* id = density units; fd converts to local density units                *
*       id = 1, kg/cubic cm;  id = 2, g/cubic cm (default)              *
*       id = 3, mole/liter;   id = 4, lb/ft                             *
*                                                                       *
* ip = pressure units; fp converts to local pressure units              *
*       ip = 1, mpascals;     ip = 2, bar (default)                     *
*       ip = 3, atmospheres;  ip = 4, psia                              *
*                                                                       *
* ih = units energy; fh converts to local energy units                  *
*       ih = 1, kjoules/kg;   ih = 2, j/g (default)                     *
*       ih = 3, j/mole;       ih = 4, cal/g                             *
*       ih = 5, cal/mole;     ih = 6, btu/lb                            *
```

```
*                                                                      *
* common /unitc/                                                       *
* character arrays that apply appropriate local unit suffix            *
************************************************************************

        common /flag/ iopt,iopt7,kopt,lopt,tvalue,itot,jopt
        common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh

* Initialize variables
      do  i = 1,25
        tvalue(i) = 0
        if(i.le.10) value(i) = 0
      repeat
      itot = 0; behind = -1; dialogptr = 0
      Handle = 0; ProcPtr = 0; itemHit = 0
      itemType = 0
      dialogflag  = .false.
      doneflag = .false.
      goway = .false.
      dfile = .false.
      data color /30/       ! White.
      it = 2                ! Set default temperature units to °C
      ip = 2                ! Set default pressure units to bar
      id = 2                ! Set default density units to g/cm3
      ih = 2                ! Set default energy units to j/g
      iopt = 1              ! Set default input option to temp.& pres.
      kopt = 1              ! Set default input to single point calc.
      lopt = 1              ! Set default table option to isothermal
      jopt = 2              ! in pressure

* Read any new default values from file 'default'

      inquire(file = 'default',exist = dfile)
      if(dfile) then        ! If the file exists, read default values
        open(10,file = 'default')
        read(10,101) it, ip, id, ih
101     format( 4(i1,1x))
        close(10)
      endif

* By default, applications running on the Mac II are allocated a stack of
* 24k. This Fortran program uses a significant amount of local memory and
* will exceed this limit. The following lines increase the amount of stack
* available.

        newlimit = LONG(applimit)         ! get the current limit
        newlimit = newlimit-256000        ! allocate an additional 250k of stack
        call toolbx(SETAPPLLIMIT,newlimit)      ! set the new application limit

* Close standard MacFortran I/O window
        mywindow = toolbx(FRONTWINDOW)
        call toolbx(CLOSEWINDOW, mywindow)

* Set new window size.
        call toolbx(SETRECT, windrect, 0, 0, 490, 293)
        grafptr=toolbx(PTR, grafport)
        call toolbx(OPENPORT, grafptr)
        mywindow = toolbx(GETNEWWINDOW, 256,
```

```
     +          toolbx(PTR, wrecord), -1)
        call toolbx(SETPORT, mywindow)
        call toolbx(ERASERECT, windrect)
        call toolbx(BACKCOLOR, color)

* Set size limits for output window.
        call toolbx(SETRECT, sizerect, 150, 100, 1100, 700)

        eventmask = -1                  ! Every event.

        call toolbx(INITDIALOGS,0)      ! Initialize dialog manager
        call toolbx(TEINIT)             ! Initialize textedit manager

* Create menu bar
        call setupmenus

* Set up a drag rectangle to prevent the window from being dragged
* off the screen.
        call toolbx(SETRECT, screenrect, 0, 0, 512, 342)
        call toolbx(SETRECT, dragrect, 4, 24,
     +     screenrect(right) - 4, screenrect(bottom) - 4)


* Main event loop.  Get and process events until doneflag is set to .true.
* by docommand.
        do

* Give the system a chance to do its thing.
        call toolbx(SYSTEMTASK)

* Get the next event and process it.
        temp = toolbx(GETNEXTEVENT, eventmask, eventrecord)
        select case(what)

          case (1)                      ! Mouse down.
            code = toolbx(FINDWINDOW, where, whichwindow)
            select case (code)

              case (1)                  ! In menu bar.
                menusel = toolbx(MENUSELECT, where)
                call docommand(menuitem(1), menuitem(2))

              case (2)                  ! In system window.
                call toolbx(SYSTEMCLICK, eventrecord, whichwindow)

              case (3)                  ! In content region.
                if (whichwindow .ne. toolbx(FRONTWINDOW)) then
                  call toolbx(SELECTWINDOW, whichwindow)
            endif

              case (4)                  ! In drag region.
                call toolbx(DRAGWINDOW, whichwindow, where, dragrect)

              case (6)                  ! In close box
                goway = toolbx(TRACKGOAWAY,mywindow,where)
                if(goway) then
                call toolbx(DISPOSEWINDOW,mywindow)
            exit
```

```
            endif

              case default                ! Ignore other window types.
            end select                  ! End window type selection

          case(3)                       ! Keyboard event
            if(modifiers .and. 256) then
              menusel = toolbx(MENUKEY,message)
              call docommand(menuitem(1),menuitem(2))
            endif

          case(6)                       ! Update
            call toolbx(BEGINUPDATE, mywindow)
            call toolbx(ENDUPDATE, mywindow)

          case default                  ! Ignore other events.
        end select                      ! End Event selection

* Prevents a system crash upon exiting; 'doneflag' set by docommand
        if (doneflag) call toolbx(DISPOSEWINDOW, mywindow)
        if (doneflag) exit              ! End main event loop
        repeat
        end


*******************************************************************************
* Once only initialization for menus.                                         *
*******************************************************************************
        subroutine setupmenus

        implicit none                   ! Force declaration.

* Toolbox definitions.
        include :include files:grafport.inc
        include :include files:windoff.inc
        include :include files:desk.inc
        include :include files:font.inc
        include :include files:event.inc
        include :include files:file.inc
        include :include files:menu.inc
        include :include files:memory.inc
        include :include files:textedit.inc
        include :include files:window.inc
        include :include files:resource.inc
        include :include files:control.inc
        include :include files:quickdraw.inc
        include :include files:misc.inc
        include :include files:dialog.inc


* Constants
        integer lastmenu, applemenu, filemenu, helpmenu
        integer programmenu, unitsmenu, optionsmenu
        integer formatmenu
        parameter (formatmenu = 251)
        parameter (lastmenu = 7, applemenu = 1)
        parameter (helpmenu = 252)
        parameter (filemenu = 253, programmenu = 254)
```

```
        parameter (unitsmenu = 255, optionsmenu = 256)
        integer top, left, bottom, right
        parameter (top = 1, left = 2, bottom = 3, right = 4)

* Local variables
        character*2 appletitle          ! Title for apple menu
        character*256 menuitems         ! Menu item list
        integer toolbx                  ! Toolbx interface definition
        integer i                       ! Counter

* Global variables.
        character*256 string            ! data variable
        integer grafptr                 ! Pointer to grafport
        integer mywindow                ! Window pointer
        integer mymenus(1 : lastmenu)   ! Menu list
        integer*2 windrect(4)           ! Window boundary
        integer iopt, iopt7         ! Input option flags
        integer kopt                ! Single point or table flag
        integer lopt, jopt          ! Table type flags
        integer tvalue,itot           ! Table input control flags

        logical doneflag                ! Exit event loop if true.
        logical dialogflag              ! Exit dialog if true

        dimension tvalue(25)

        common mymenus, mywindow, doneflag, grafptr,
     &  windrect, dialogflag
        common /flag/ iopt,iopt7,kopt,lopt,tvalue,itot,jopt

* Set up the apple menu with the applec character for a title and add
* all available desk accessories to it.
        appletitle = char(1) // char(z'14')
        mymenus(1) = toolbx(NEWMENU, applemenu, appletitle)
        menuitems = char(20)//"About Macwat...;-("
        call toolbx(APPENDMENU, mymenus(1), menuitems)
        call toolbx(ADDRESMENU, mymenus(1), 'DRVR')

* Get other menus from the resource file.
        mymenus(2) = toolbx(GETMENU, filemenu)
        mymenus(3) = toolbx(GETMENU, programmenu)
        mymenus(4) = toolbx(GETMENU, formatmenu)
        mymenus(5) = toolbx(GETMENU, unitsmenu)
        mymenus(6) = toolbx(GETMENU, optionsmenu)
        mymenus(7) = toolbx(GETMENU, helpmenu)

* Put the menus in the menu bar and draw it.
        do (i = 1, lastmenu)
          call toolbx(INSERTMENU, mymenus(i), 0)
        repeat
        call toolbx(DRAWMENUBAR)

        end


* ***************************************************************************
* subroutine docommand                                                      *
* This subroutine processes a menu selection from a menu id and an item #   *
```

```
*******************************************************************************
        subroutine docommand(menu, item)
        integer*2 menu, item            ! Menu id and item # of selection.

        implicit none                   ! force declaration.

* Toolbox definitions.
        include :include files:grafport.inc
        include :include files:windoff.inc
        include :include files:desk.inc
        include :include files:font.inc
        include :include files:event.inc
        include :include files:file.inc
        include :include files:menu.inc
        include :include files:memory.inc
        include :include files:textedit.inc
        include :include files:window.inc
        include :include files:resource.inc
        include :include files:control.inc
        include :include files:quickdraw.inc
        include :include files:misc.inc
        include :include files:dialog.inc


* Constants
        integer lastmenu, applemenu, filemenu, helpmenu
        integer programmenu, unitsmenu, optionsmenu
        integer formatmenu
        parameter (formatmenu = 251)
        parameter (lastmenu = 7, applemenu = 1)
        parameter (helpmenu = 252)
        parameter (filemenu = 253, programmenu = 254)
        parameter (unitsmenu = 255, optionsmenu = 256)
        integer top, left, bottom, right
        parameter (top = 1, left = 2, bottom = 3, right = 4)

* Local variables
        character*256 name,string,string1   ! data variables
        character*256 str255          ! function to create a Pascal LSTRING
        character*64 infil, outfil    ! input & output file names
        character*8 nt,np,nd,nh
        integer refnum                ! file reference number
        integer toolbx                ! declare toolbox as integer function
        integer item4                 ! 4 byte item number.
        logical temp                  ! Active event when true
        integer err, err1             ! Location on I/O error
        integer gpdump                ! Function to print a grafport
        integer j, i, k               ! Counters

* Global variables.
        integer mywindow              ! Window pointer
        integer*2 windrect(4)         ! Window coordinates
        integer grafptr               ! Pointer to grafport
        integer*2 Rect(4)             ! Parameter coordinates
        integer mymenus(1 : lastmenu) ! Menu list
        integer it, ip, id, ih        ! unit variables
        integer iopt, iopt7           ! Input option flags
        integer kopt                  ! Single point or table flag
```

```
      integer lopt, jopt          ! Table-type flags
      integer tvalue,itot         ! Table input control flags

* Dialog box parameters
      integer*4 dialogptr         ! Pointer to dialog box
      integer*4 dialogID          ! Resource id# of dialog template
      integer*4 ProcPtr           ! Event filter parameter
      integer*4 dStorage          ! Pointer to storage usage
      integer*4 Handle            ! Handle to resource item list
      integer*2 itemType          ! Diaolog item reference number
      integer*2 itemHit           ! Handle to individual resource item
      integer*4 behind            ! Specifies window placement
      integer value, val          ! Control switches

      logical doneflag, helpflag, more     ! Exit loop when true.
      logical dialogflag                   ! Exit dialog when true
      logical stdfil, ioerrs, endlog, fileinuse  ! File flags
      logical goway                              ! Exit program when true

      dimension value(10), tvalue(25), val(10)
      double precision  ft,fd,fp,fh

      common mymenus, mywindow, doneflag, grafptr,
     & windrect, dialogflag
      common /flag/ iopt,iopt7,kopt,lopt,tvalue,itot,jopt
      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh

* Initialize variables
      data val/10*0/
      data value/10*0/

      behind = -1                  ! Put window in front
      dialogptr = 0                ! Make sure variables are zero to start
      Handle = 0
      ProcPtr = 0
      itemHit = 0
      itemType = 0

      item4 = item                 ! Convert to 4 bytes.

* The following 'select case' processes user selected menu items.
      select case (menu)           ! Mouse down in menu bar

        case (applemenu)           ! Apple menu selected
        call toolbx(ERASERECT,windrect)
        call toolbx(DRAWMENUBAR)
        select case (item)

          case (1)                 ! 'About MacH2O' selected
            dialogID = 262
            dStorage = 0
            dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
            call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
            call toolbx(SETPORT,dialogptr)
            call toolbx(PENSIZE,3,3)
            call toolbx(INSETRECT,Rect,-4,-4)
            call toolbx(FRAMEROUNDRECT,Rect,16,16)
```

```
    do                          ! Set up dialog box
      call toolbx(MODALDIALOG,0,itemHit)
      select case(itemHit)

    case(1)                     ! Exit 'About MacH2O' dialog box
      call toolbx(CLOSEDIALOG,dialogptr)
      exit

    case default

      endselect                 ! End 'About MacH2O' selection
    repeat

  case default                  ! Any other apple menu selection
    call toolbx(GETITEM, mymenus(1), item4, name)
    refnum = toolbx(OPENDSKACC, name)
    call toolbx(SETPORT, mywindow)
end select

case (filemenu)                 ! File menu selected
  call toolbx(DRAWMENUBAR)
  select case (item)

    case (1)                    ! Print data
      err1 = gpdump(grafptr,1)

    case (2)                    ! Quit program
      doneflag = .true.

    case default
  end select                    ! End file menu election

case (programmenu)              ! program menu selected
  call toolbx(ERASERECT,windrect)
  call toolbx(DRAWMENUBAR)
  select case (item)

    case (1)                    ! Run program
      call h2o1(doneflag,windrect,grafptr)

    case default
  end select                    ! End program menu selection

case (formatmenu)               ! Format menu selected
  call toolbx(ERASERECT,windrect)
  call toolbx(DRAWMENUBAR)
  select case (item)

   case (1)                     ! Single point format selected
    kopt = 1
    mymenus(3) = toolbx(GETMENU, formatmenu)
    call toolbx(CHECKITEM,mymenus(3),1,.true.)
    call toolbx(CHECKITEM,mymenus(3),2,.false.)

   case (2)
    kopt = 2                    ! Table format selected
    mymenus(3) = toolbx(GETMENU, formatmenu)
    call toolbx(CHECKITEM,mymenus(3),2,.true.)
```

```
              call toolbx(CHECKITEM,mymenus(3),1,.false.)

*************************************************************************
* Set up default table type and present table options                   *
*     lopt = 1, Isothermal table (default)                              *
*                jopt = 2, temperature & pressure input      (default)  *
*             jopt = 3, temperature & density input                     *
*     lopt = 2, Isobaric table                                          *
*     lopt = 3, Isochoric table                                         *
*     lopt = 4, Saturaton table in temperature                          *
*     lopt = 5, Saturaton table in pressure                             *
*************************************************************************
              dialogID = 291
              dialogflag = .false.
              dStorage = 0
              dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
              call toolbx(GETDITEM,dialogptr,lopt + 3,
     &              itemType,Handle,Rect)
              call toolbx(SETCTLVALUE,Handle,1)
              if(lopt.eq.1) then
                call toolbx(GETDITEM,dialogptr,jopt + 7,
     &               itemType,Handle,Rect)
                call toolbx(SETCTLVALUE,Handle,1)
              endif
              call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
              call toolbx(SETPORT,dialogptr)
              call toolbx(PENSIZE,3,3)
              call toolbx(INSETRECT,Rect,-4,-4)
              call toolbx(FRAMEROUNDRECT,Rect,16,16)

              do                        ! Event loop to select table type
              call toolbx(MODALDIALOG,0,itemHit)
              select case(itemHit)

               case(1)                  ! Retrieve user selected table type.
                 do 50, i=4,10
                   call toolbx(GETDITEM,dialogptr,i,itemType,Handle,
     &                    Rect)
                   val(j) = toolbx(GETCTLVALUE,Handle)
               if(val(j).eq.1) then
                 select case(i)

                 case(4:8)
                   lopt = i - 3

                 case(9:10)
                       jopt = i - 7

                 case default
               end select
                   endif
50               continue
                 dialogflag = .true.
             menu = programmenu                    ! Set to call variable select
                 call toolbx(SETPORT, mywindow)

               case (2)                            ! Quit, exit program
             dialogflag = .true.
```

```
            doneflag = .true.
          call toolbx(SETPORT, mywindow)

            case (3)                              ! Cancel, no selection
          dialogflag = .true.
          menu = formatmenu
          call toolbx(SETPORT, mywindow)

* Activate appropriate checkboxes for table-type selections

            case (4)                              ! Isothermal table
              call setchki(dialogptr,itemType,Handle,Rect,4)
          call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,1)

            case (5)                              ! Isobaric table
              call setchki(dialogptr,itemType,Handle,Rect,5)

            case (6)                              ! Isochoric table
              call setchki(dialogptr,itemType,Handle,Rect,6)

            case (7)                          ! Saturation table in temp.
              call setchki(dialogptr,itemType,Handle,Rect,7)

            case (8)                          ! Saturation table in pres.
              call setchki(dialogptr,itemType,Handle,Rect,8)

            case (9)                          ! isothermal versus pressure
              call setchki(dialogptr,itemType,Handle,Rect,4)
              call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,1)

            case (10)                         ! Isothermal versus density
              call setchki(dialogptr,itemType,Handle,Rect,4)
          call toolbx(GETDITEM,dialogptr,10,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,1)

            case default
           end select                         ! End select table type

           if(dialogflag) then                ! Exit dialog box when true
             call toolbx(CLOSEDIALOG,dialogptr)
         exit
           endif
           repeat                             ! End loop for table type
           if (menu.eq.programmenu) then      ! Call variable selection
             call h2ol(doneflag,windrect,grafptr)
           end if
          case default
         endselect                            ! End select for format

        case (unitsmenu)                      ! Unit menu selected
          call toolbx(ERASERECT,windrect)
          call toolbx(DRAWMENUBAR)
          select case (item)

           case (1)                           ! user selected temperature units
            dialogflag = .false.
```

```
            dialogID = 257
            dStorage = 0
            dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
            call toolbx(GETDITEM,dialogptr,it+3,itemType,Handle,Rect)
            call toolbx(SETCTLVALUE,Handle,1)
            call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
            call toolbx(SETPORT,dialogptr)
            call toolbx(PENSIZE,3,3)
            call toolbx(INSETRECT,Rect,-4,-4)
            call toolbx(FRAMEROUNDRECT,Rect,16,16)

            do              ! Event loop for temp. unit selection
            call toolbx(MODALDIALOG,0,itemHit)
            select case(itemHit)

             case(1,8)                      ! OK & new default
              j = 1
              do 58, i=4,7
                call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
                value(j) = toolbx(GETCTLVALUE,Handle)
                if(value(j).eq.1) it = j              ! Retrieve temp. units
              j = j + 1
58            continue
              if(itemHit.eq.8) then             ! User selected new defaults
                open(10,file = 'default')       ! Store new defaults in file
                write(10,101) it, ip, id, ih
101             format( 4(i1,1x))
                close(10)
              endif
              dialogflag = .true.
              call toolbx(SETPORT, mywindow)

             case (2)                           ! Cancel
              dialogflag = .true.
           call toolbx(SETPORT, mywindow)
* Place mark in appropriate user selected box and remove old mark
             case (4)                           ! °K
              call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
              call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,5,7,0)

             case (5)                           ! °C
           call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
              call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,7,5)

             case (6)                           ! °R
           call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
              call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,7,6)

             case (7)                           ! °F
           call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
              call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,6,0)

             case default
            end select                          ! End temp. unit selection
```

```
            if(dialogflag) then                  ! Exit when true
              call toolbx(CLOSEDIALOG,dialogptr)
              exit
            endif
            repeat

           case (2)                              ! user selected pressure units
            dialogflag = .false.
            dialogID = 259
            dStorage = 0
            dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
            call toolbx(GETDITEM,dialogptr,ip+3,itemType,Handle,Rect)
            call toolbx(SETCTLVALUE,Handle,1)
            call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
            call toolbx(SETPORT,dialogptr)
            call toolbx(PENSIZE,3,3)
            call toolbx(INSETRECT,Rect,-4,-4)
            call toolbx(FRAMEROUNDRECT,Rect,16,16)

            do                       ! Event loop for pressure unit selection
            call toolbx(MODALDIALOG,0,itemHit)
            select case(itemHit)

             case(1, 8) ! OK & New Default
           j = 1

               do 63, i=4,7                  ! Retrieve pressure units
                call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
                value(j) = toolbx(GETCTLVALUE,Handle)
            if(value(j).eq.1) ip = j
            j = j + 1
63             continue

               if(itemHit.eq.8) then     ! Set new defaults
                 open(10,file = 'default')
                 write(10,101) it, ip, id, ih
             close(10)
           endif
               dialogflag = .true.
               call toolbx(SETPORT, mywindow)

             case (2)                        ! Cancel, no change
           dialogflag = .true.
           call toolbx(SETPORT, mywindow)

* Place mark in appropriate user selected box and remove old mark
             case (4)                        ! mpa
               call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
               call toolbx(SETCTLVALUE,Handle,1)
               call remvx (dialogptr,itemType,Handle,Rect,5,7,0)

             case (5)                        ! bar
           call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)

               call toolbx(SETCTLVALUE,Handle,1)
           call remvx (dialogptr,itemType,Handle,Rect,4,7,5)

             case (6)                        ! atm
```

```
            call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
                call toolbx(SETCTLVALUE,Handle,1)
            call remvx (dialogptr,itemType,Handle,Rect,4,7,6)

              case (7)                      ! psia
            call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
                call toolbx(SETCTLVALUE,Handle,1)
                call remvx (dialogptr,itemType,Handle,Rect,4,6,0)

              case default
             end select                     ! End pressure unit select
             if(dialogflag) then            ! Exit when true
               call toolbx(CLOSEDIALOG,dialogptr)
               exit
             endif
             repeat

            case (3)                        ! user selected density units
             dialogflag = .false.
             dialogID = 258
             dStorage = 0
             dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
             call toolbx(GETDITEM,dialogptr,id+3,itemType,Handle,Rect)
             call toolbx(SETCTLVALUE,Handle,1)
             call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
             call toolbx(SETPORT,dialogptr)
             call toolbx(PENSIZE,3,3)
             call toolbx(INSETRECT,Rect,-4,-4)
             call toolbx(FRAMEROUNDRECT,Rect,16,16)

             do        ! Event loop for density unit selection
             call toolbx(MODALDIALOG,0,itemHit)
             select case(itemHit)

              case(1,8)                     ! OK & New Default
            j = 1

                do 68, i=4,7                ! Retrive density units
                 call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
                 value(j) = toolbx(GETCTLVALUE,Handle)
             if(value(j).eq.1) id = j
                 j = j + 1
68           continue

                if(itemHit.eq.8) then       ! Change default
                  open(10,file = 'default')
                  write(10,101) it, ip, id, ih
              close(10)
            endif
                dialogflag = .true.
            call toolbx(SETPORT, mywindow)

              case (2)                       ! Cancel, no change
                dialogflag = .true.
            call toolbx(SETPORT, mywindow)

* Place mark in appropriate user selected box and remove old mark
              case (4)                       ! kg/cubic m
```

```
      call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
      call toolbx(SETCTLVALUE,Handle,1)
      call remvx (dialogptr,itemType,Handle,Rect,5,7,0)

    case (5)                          ! g/cubic cm
  call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
      call toolbx(SETCTLVALUE,Handle,1)
  call remvx (dialogptr,itemType,Handle,Rect,4,7,5)

    case (6)                          ! mol/l
  call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
      call toolbx(SETCTLVALUE,Handle,1)
      call remvx (dialogptr,itemType,Handle,Rect,4,7,6)

    case (7)                          ! lb/ft
  call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
      call toolbx(SETCTLVALUE,Handle,1)
      call remvx (dialogptr,itemType,Handle,Rect,4,6,0)

    case default
   end select                         ! End density unit selection
   if(dialogflag) then
   call toolbx(CLOSEDIALOG,dialogptr)
   exit
   endif
   repeat                             ! End loop for density units

  case (4)                            ! Select energy units
    dialogflag = .false.
    dialogID = 260
    dStorage = 0
    dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
    call toolbx(GETDITEM,dialogptr,ih+3,itemType,Handle,Rect)
    call toolbx(SETCTLVALUE,Handle,1)
    call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
    call toolbx(SETPORT,dialogptr)
    call toolbx(PENSIZE,3,3)
    call toolbx(INSETRECT,Rect,-4,-4)
    call toolbx(FRAMEROUNDRECT,Rect,16,16)

    do       ! Event loop for energy units
    call toolbx(MODALDIALOG,0,itemHit)
    select case(itemHit)

     case(1,10)                         ! OK & New Default
   j = 1

       do 73, i=4,9              ! Retrieve energy unit selection
         call toolbx(GETDITEM,dialogptr,i,itemType,Handle,
&                    Rect)
         value(j) = toolbx(GETCTLVALUE,Handle)
     if(value(j).eq.1) ih = j
     j = j + 1
         if(itemHit.eq.10) then      ! New default
           open(10,file = 'default')
           write(10,101) it, ip, id, ih
       close(10)
     endif
```

```
73              continue

                dialogflag = .true.
              call toolbx(SETPORT, mywindow)

                case (2)                          ! Cancel
              dialogflag = .true.
              call toolbx(SETPORT, mywindow)

                case (4)                          ! kg/kg
                  call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
                  call remvx (dialogptr,itemType,Handle,Rect,5,9,0)

                case (5)                          ! j/g
                  call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,9,5)

                case (6)                          ! j/mole
              call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,9,6)

                case (7)                          ! cal/g
              call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,9,7)

                case (8)                          ! cal/mole
              call toolbx(GETDITEM,dialogptr,8,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,9,8)

                case (9)                          ! btu/lb
              call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
                  call toolbx(SETCTLVALUE,Handle,1)
              call remvx (dialogptr,itemType,Handle,Rect,4,8,0)

                case default
               end select                         ! End energy unit selection
               if(dialogflag) then                ! Done
               call toolbx(CLOSEDIALOG,dialogptr)
               exit
               endif
               repeat                             ! End loop for energy units

              case default
             end select                           ! End units menu

            case (optionsmenu)                    ! Select input variable
            call toolbx(ERASERECT,windrect)
            call toolbx(DRAWMENUBAR)
*****************************************************************************
* Place checkmark in user-selected menu items and remove others.            *
*****************************************************************************
            select case (item)
```

```
      case (1)
        iopt = 1                    ! Input temperature and pressure
        mymenus(6) = toolbx(GETMENU, optionsmenu)
        call toolbx(CHECKITEM,mymenus(6),1,.true.)
        do 80 i = 2,10          ! Remove checkmark from all other boxes
        call toolbx(CHECKITEM,mymenus(6),i,.false.)
80      continue

      case (2)                      ! Input temperature and density
        iopt = 2
        kopt = 1
        call setx(2,3,10)

      case (3)                      ! Input temperature & entropy
        iopt = 3
        kopt = 1
        call setx(3,4,10)

      case (4)                      ! Input temperature & enthalpy
        iopt = 4
        kopt = 1
        call setx(4,1,10)
        call toolbx(CHECKITEM,mymenus(6),4,.true.)

      case (5)                      ! Input pressure & enthalpy
        iopt = 5
        kopt = 1
        call setx(5,1,10)
        call toolbx(CHECKITEM,mymenus(6),5,.true.)

      case (6)                      ! Input pressure & entropy
        iopt = 6
        kopt = 1
        call setx(6,1,10)
        call toolbx(CHECKITEM,mymenus(6),6,.true.)

      case (9)                      ! Input @ saturation temperature
        iopt = 7
        iopt7 = 1
        kopt = 1
        call setx(9,1,10)
        call toolbx(CHECKITEM,mymenus(6),9,.true.)

      case (10)                     ! Input @ saturation pressure
        iopt = 7
        iopt7 = 2
        kopt = 1
        call setx(10,1,10)
        call toolbx(CHECKITEM,mymenus(6),10,.true.)

      case default
    end select                      ! End options menu

    case (helpmenu)                 ! Help menu selected
    call toolbx(ERASERECT,windrect)
    more = .false.
    select case (item)
```

```
            case (1,2,3)
              dialogID = 273              ! Help on single point calculations
              if(item.eq.2) dialogID = 274      ! Help on table calculations
88            if(item.eq.3.or.more) dialogID = 276 ! More help
              dStorage = 0
              dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
              call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
              call toolbx(SETPORT,dialogptr)
              call toolbx(PENSIZE,3,3)
              call toolbx(INSETRECT,Rect,-4,-4)
              call toolbx(FRAMEROUNDRECT,Rect,16,16)
              call toolbx(TEXTFONT,0)
              call toolbx(TEXTSIZE,12)
              call toolbx(DRAWMENUBAR)
              i = 2
              j = 15
              if(dialogID.eq.274) then
                i = 3; j = 17
              else if(dialogID.eq.276) then
                i = 3; j = 9
              endif
              do 89 k = i,j
                call toolbx(GETDITEM,dialogptr,k,itemType,Handle,Rect)
                call toolbx(GETITEXT,Handle,string1)
                call toolbx(TEXTFONT,3)
                call toolbx(SETITEXT,Handle,string1)
89            continue

              do        ! Event loop for help dialog boxes
                call toolbx(MODALDIALOG,0,itemHit)
                select case(itemHit)

              case(1) ! OK
                call toolbx(CLOSEDIALOG,dialogptr)
                call toolbx(SETPORT, mywindow)
                exit

                  case(2) ! More
                call toolbx(CLOSEDIALOG,dialogptr)
                more = .true.
                go to 88

              case default
                endselect           ! End help box selection
                repeat              ! End loop help dialog boxes
              case default
            end select              ! End help menu selection
          case default
        end select                  ! End all menu selections
        call toolbx(TEXTFONT,0)     ! Return to system font & size
        call toolbx(TEXTSIZE,12)
        call toolbx(DRAWMENUBAR)
        call toolbx(HILITEMENU, 0)  ! Unhilite the selected menu.
        return
        end

*************************************************************************
* Activate checkboxes for table selections                              *
```

```
************************************************************************
        subroutine setchki(dialogptr,itemType,Handle,Rect,iloc)
        integer*4 dialogptr,Handle
        integer*2 itemType,Rect(4),iloc

        include :include files:control.inc
        include :include files:dialog.inc

        integer i
        do 10 i = 4,10
          call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,0)
10      continue
        call toolbx(GETDITEM,dialogptr,iloc,itemType,Handle,
     &              Rect)
        call toolbx(SETCTLVALUE,Handle,1)
        return
        end

************************************************************************
* Place check marks in selected menu items: input variables.           *
************************************************************************
        subroutine setx(icase,ibeg,iend)
        integer icase,ibeg,iend

        include :include files:menu.inc

        integer lastmenu,optionsmenu,formatmenu
        parameter (lastmenu=7,optionsmenu=256,formatmenu=251)

* Global variables
        integer mymenus(1:lastmenu)
        integer toolbx

        mymenus(3) = toolbx(GETMENU,formatmenu)
        call toolbx(CHECKITEM,mymenus(3),1,.true.)
        call toolbx(CHECKITEM,mymenus(3),2,.false.)
        mymenus(6) = toolbx(GETMENU,optionsmenu)
        if(icase.eq.2) then
          call toolbx(CHECKITEM,mymenus(6),1,.false.)
          call toolbx(CHECKITEM,mymenus(6),2,.true.)
        elseif(icase.eq.3) then
          call toolbx(CHECKITEM,mymenus(6),1,.false.)
          call toolbx(CHECKITEM,mymenus(6),2,.false.)
          call toolbx(CHECKITEM,mymenus(6),3,.true.)
        endif
        do 10 i=ibeg,iend
          call toolbx(CHECKITEM,mymenus(6),i,.false.)
10      continue
        return
        end


************************************************************************
* Remove 'X' from non-selected boxes: temp., pres., dens., energy.     *
************************************************************************
        subroutine remvx(dialogptr,itemType,Handle,Rect,ibeg,iend,iskp)
```

```
      integer*4 dialogptr,Handle
      integer*2 itemType, Rect(4)

      integer ibeg,iend,iskp

      implicit none

      include :include files:control.inc
      include :include files:dialog.inc

      integer i

      do 10 i = ibeg,iend
        if(i.ne.iskp) then
          call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,0)
        endif
10    continue
      end


*  str255: converts a FORTRAN string to a Pascal LSTRING
*  reprinted from propram 'DEMO.FOR' supplied by Absoft
      character*256 function str255(string)
      character*(*) string

      str255 = char(len(trim(string)))//string
      return
      end
```

```
*************************************************************************
*                        Appendix B                                     *
* Subroutine h2o1 drives the computational part of the program. It and  *
* subsequent routines have been extensively modified from the computer  *
* code of Haar, Gallagher and Kell (1984).                              *
* doneflag = 'true' when done; windrect = window coordinates;           *
* grafptr = pointer to grafport                                         *
* ***********************************************************************
      subroutine h2o1(doneflag,windrect,grafptr)
      implicit double precision (a-h,o-z)

      include :include files:dialog.inc
      include :include files:file.inc
      include :include files:textedit.inc
      include :include files:resource.inc
      include :include files:control.inc
      include :include files:quickdraw.inc
      include :include files:window.inc

      character*8 nv(4),njt(5),nt,nd,np,nh,ns,ns1,ns2,name(20),nunt(20)
      character*255 string,stringp,stringn
      character*9 ntr(4),nk(5),nht(6)
      character*13 density, entropy
      character*13 pressure, enthalpy
      character*13 temp
      character*35 string5
      character*40 title
      character*256 str255
      character*4 sec

* Common block 'units' contains integer and character variables
* associated with user selected units; common block 'params' contains
* variable parameters defined in block data; common block 'props'
* contains the calculated properties of water; common block 'flag'
* contains option and unit flags.
      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)
      common /props/ p,ad,gd,hd,ud,sd,cvd,cpd,dpdd,dpdt,dvdt,w,cjtt,cjth
      common /flag/ iopt,iopt7,kopt,lopt,tvalue,itot,jopt

      logical doneflag,variable

* Dialog box parameters
        integer*4 dialogptr       ! Pointer to dialog box
        integer*4 dialogID        ! Resource id of dialog template
        integer*4 ProcPtr         ! Event filter parameter
        integer*4 dStorage        ! Pointer to storage
        integer*4 Handle          ! Handle to item list of dialog
        integer*2 itemType        ! Predefined constant for dialog items
        integer*2 itemHit         ! Handle to individual dialog item
        integer*4 behind          ! Specifies window placement (-1 = in front)
        integer item              ! Item of item list
        integer  tvalue           ! Checkbox control monitor

* Window boundary and pointer variables
      integer grafptr
      integer*2 Rect(4)
```

```
      integer*2 windrect(4)

* Declare the toolbox interface as an integer function
      integer*4 toolbx

* Global variables  (Flags & counters)
      integer iopt,i,iopt7,kopt,lopt,jopt
      integer it,ip,id,ih
      integer ipi
      integer line1,ct,err1

      dimension stringp(25),stringn(25)
      dimension tvalue(25),ipi(4),title(20)
      dimension prop(20)

      parameter (sec = '/sec')
      parameter (pressure = 'Pressure :    ')
      parameter (enthalpy = 'Enthalpy :    ')
      parameter (entropy = 'Entropy :     ')
      parameter (density = 'Density :     ')
      parameter (temp = 'Temperature :')

* Unit headings for table output
      data njt/'°/mpa    ','°/bar    ','°/atm    ','°/psi    ','dcm2/kg '/
      data nk/'/mpa     ','/bar     ','/atm     ','/psi     ','cm2/kg  '/
      data nv/'m3/kg    ','cm3/g    ','l/mol    ','ft3/lb  '/
      data nht/'kj/kg.K ','j/(g.K) ','j/mol.K ','cal/g.K ','cal/molK'
     1,'btu/lb.F '/
      data ns1,ns2/'m/sec    ','ft/sec  '/
      data ntr/'pa sec  ','psi sec ','w/(m.K) ','btu/hftF '/
      data stringn/25*0/

* Property headings for table output
      data name/' TEMP    ','PRESSURE','DENSITY ','VOLUME  ','ENTHALPY'
     1,'ENTROPY ','INT ENER','    CV    ','    CP    ','    A     ','    G     '
     2,'  DP/DD ','  DP/DT ','  COMPR ',' SP SND ',' TH COND',
     3 '   VISC ','J-T COEF','         ','         '/
      data stringp/25*0/
      data ipi/4*0/

* Table titles
      title(1) = ' for an isothermal table in pressure.'
      title(2) = ' for an isobaric table.'
      title(3) = ' for an isochoric table.'
      title(4) = ' for a saturation table in temperature.'
      title(5) = ' for a saturation table in pressure.'
      title(6) = ' for an isothermal table in density.'
      string5 = ' Choose up to 3 dependent variables'

* Initialize variables
      behind = -1; Handle = 0
      dialogptr = 0; ProcPtr = 0
      itemType = 0; itemHit = 0
      line1 = 0; ct = 0
      jjopt = jopt
      llopt = lopt

      call toolbx(TEINIT)                ! Initialize textedit manager
```

```
      call toolbx(INITDIALOGS,0)        ! Initialize dialog manager

  50  call unit

      tz=.999778D0
      ns=ns1
      if(id.eq.4) ns=ns2
      nunt(1)=nt                        ! Array nunt contains property unit
      nunt(2)=np
      nunt(3)=nd
      nunt(4)=nv(id)
      nunt(5)=nh
      nunt(6)=nht(ih)
      nunt(7)=nh
      nunt(8)=nht(ih)
      nunt(9)=nht(ih)
      nunt(10)=nh
      nunt(11)=nh
      nunt(12)=name(20)
      nunt(13)=name(20)
      nunt(14)=nk(ip)
      nunt(15)=ns
      nunt(16)=ntr(3)
      if(it.eq.4) nunt(16)=ntr(4)
      nunt(17)=ntr(1)
      if(it.eq.4) nunt(17)=ntr(2)
      nunt(18)=njt(ip)

*************************************************************************
*  When kopt = 1, single point calculations have been selected          *
*  When kopt = 2, table format has been selected                        *
*************************************************************************
      if(kopt.lt.0) go to 96
      if(kopt-1) 50,56,51
51    continue

* Set up dialog box for the user to select which dependent variables are
* to be tabulated
      dialogID = 290
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETDITEM,dialogptr,24,itemType,Handle,Rect)
      if(lopt.eq.1.and.jopt.eq.3) then              ! Input table heading
      call toolbx(SETITEXT,Handle,str255(string5 // title(6)))
      else
      call toolbx(SETITEXT,Handle,str255(string5 // title(lopt)))
      endif

      itot = 0

      do 52 i = 4,22          ! Activate check boxes with 'X'
        if(tvalue(i).eq.1) then
          itot = itot + 1
          call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
          call toolbx(SETCTLVALUE,Handle,1)
        endif
52    continue
```

```
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)
53    do                ! loop for dialog box selection

      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

        case(1)                  ! Continue
          j = 3
          do 54 i = 4,22       ! Sum number of dependent variables
            call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
            tvalue(i) = toolbx(GETCTLVALUE,Handle)
            if(tvalue(i).eq.1) then
              ipi(j) = i - 3
              j = j + 1
            endif
54          continue
          if(itot.eq.0) then  ! None selected, warn user
            call toolbx(ALERT,202,0)
            go to 53
          endif
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT, grafptr)
          exit

        case(2)                                ! QUIT
          call toolbx(CLOSEDIALOG,dialogptr)
          doneflag = .true.
          call toolbx(SETPORT,grafptr)
          go to 96

        case(3)                                ! Return to menu
          do 55 i = 4,22       ! Remember selected items
            call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
            tvalue(i) = toolbx(GETCTLVALUE,Handle)
55          continue
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          go to 96

        case(4:22)
          item = itemHit                       ! User selected variable
          call toolbx(GETDITEM,dialogptr,item,itemType,Handle,Rect)
          tvalue(item) = toolbx(GETCTLVALUE,Handle)
          if(tvalue(item).eq.0) then
            if(itot.eq.3) call toolbx(ALERT,203,0) ! Make sure only 3 variables
            if(itot.lt.3) then                     ! are chosen
              itot = itot + 1
              call toolbx(SETCTLVALUE,Handle,1)
            endif
          else if (tvalue(item).eq.1) then
            call toolbx(SETCTLVALUE,Handle,0)
            itot = itot - 1
          endif
```

```
        case default
       end select        ! Ends dialog box selection
       repeat                  ! Ends loop for dialog box

       itot1 = itot
       call iso(name,nunt,ipi,doneflag,jjopt,variable,itot1,
      &windrect,llopt,line1,ct)       ! table calculations and output
       if(variable) go to 51
       return

56     continue
*************************************************************************
* Set appropriately labeled dialog box for single input with correct    *
* independent variable name in the selected units.                      *
* iopt = 1, input temperature & pressure                                *
* iopt = 2, input temperature and pressure                              *
* iopt = 3, input temperature and entropy                               *
* iopt = 4, input temperature and enthalpy                              *
* iopt = 5, input pressure and enthalpy                                 *
* iopt = 6, input pressure and entropy                                  *
* iopt = 7, input temperature or pressure along saturation curve        *
*************************************************************************
57     dialogID = 270
       dStorage = 0
       dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
       go to (58,59,60,61,62,63,64), iopt

58     call setbox(dialogptr,itemType,Handle,Rect,
      &            temp,nunt(1),pressure,nunt(2))
       go to 65

59     call setbox(dialogptr,itemType,Handle,Rect,
      &            temp,nunt(1),density,nunt(3))
       go to 65

60     call setbox(dialogptr,itemType,Handle,Rect,
      &            temp,nunt(1),entropy,nunt(6))
       go to 65

61     call setbox(dialogptr,itemType,Handle,Rect,
      &            temp,nunt(1),enthalpy,nunt(5))
       go to 65

62     call setbox(dialogptr,itemType,Handle,Rect,
      &            pressure,nunt(2),enthalpy,nunt(5))
       go to 65

63     call setbox(dialogptr,itemType,Handle,Rect,
      &            pressure,nunt(2),entropy,nunt(6))
       go to 65

64     call toolbx(CLOSEDIALOG,dialogptr)
       dialogID = 278        ! Input temp. or pres. along saturation curve
       dStorage = 0
       dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
       if(iopt7.eq.1) then
       call toolbx(GETDITEM,dialogptr,3,itemType,Handle,Rect)
```

```
      call toolbx(SETITEXT,Handle,str255(temp))
      call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(nunt(1)))

      else if(iopt7.eq.2) then
      call toolbx(GETDITEM,dialogptr,3,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(pressure))
      call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(nunt(2)))
      endif
      go to 65

65    call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

66    continue

*************************************************************************
* Retrieve user supplied input of indpendent variables from dialog box *
*************************************************************************
      do                ! Loop to get independent variables
67    call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

        case(1)                          ! Continue
          yy = 0.; x = 0.
          string = 0
          call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
          call toolbx(GETITEXT,Handle,string)
          string = string(2:)
          read(string,100) yy
100       format(f30.0)
          if(yy.le.0) then
            call toolbx(ALERT,201,0)
            go to 67
          end if
          string = 0
          if(iopt.eq.7.and.iopt7.eq.1) then
            x = 0
            go to 68
          else if(iopt.eq.7.and.iopt7.eq.2) then
            x = yy
            yy = 0
            go to 68
          endif
          call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
          call toolbx(GETITEXT,Handle,string)
          string = string(2:)
          read(string,100) x
          if(x.le.0) then
            call toolbx(ALERT,201,0)
            go to 67
          end if
68        call toolbx(CLOSEDIALOG,dialogptr)
```

```
          call toolbx(SETPORT, grafptr)
          exit

        case(2)                          ! QUIT
          call toolbx(CLOSEDIALOG,dialogptr)
          doneflag = .true.
          call toolbx(SETPORT,grafptr)
          go to 96

        case(7)                          ! Menu
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          go to 96

        case default
      end select                 ! End independent variable selection
      repeat                     ! End loop for independent vars.
      call toolbx(ERASERECT,windrect)
      if(iopt.le.0) go to 96     ! Exit, no selection
      i111=0
      tt=yy
      if(iopt.eq.5 .or. iopt.eq.6) go to 78    ! Calculate reduced pressure
      t=ttt(tt)
      if(iopt.eq.7) go to 69     ! Calculate saturation values
      dll=3.5d0-.2d0*t
      dvv=0.d0
      psat=1.d3
      if(t.gt..999778d0) go to 71 ! Temperature is above critical point

* Calculate saturation values
      call pcorr(t,psat,dll,dvv)
      sv=sd
      hv=hd
      call therm(dll,t)
      sl=sd
      hl=hd
      ddll=dll*ds/fd
      ddvv=dvv*ds/fd
      psatx=psat*fp*ps

* Print saturation values for all samples below the critical point
      call toolbx(MOVETO,0,225)
      write(*,101)tt,nunt(1),psatx,nunt(2),ddll,nunt(3),ddvv,nunt(3)
101   format(/2x,' for temperature = ',f6.2,a8,
     &'   vapor pressure = ',g10.4,a8/
     12x,' saturated liquid density = ',g10.4,1x,a8,/
     22x,' saturated vapor density =  ',g10.4,1x,a8)
      go to 71
69    dll=0.d0
      dvv=0.d0
      if(tt.ne.0.d0) call pcorr(t,psat,dll,dvv) ! Calc. corresponding vapor pres.
      if(x.ne.0.d0) then
          pp=x
          psat=pp/fp/ps
          if (psat.ge..9972d0) then  ! Above critical point
            call toolbx(MOVETO,10,245)
            print '(a)', 'pressure is supercritical'
            return
```

```
          endif
          call tcorr(t,psat,dll,dvv) ! Calc. corresponding vapor temperature
          tt=tti(t)
          endif
      pres=psat*fp
      i111=1
      call toolbx(MOVETO,10,245)
      write(*,102)
102   format('  saturated liquid properties ')
      d=dll
      if(t.ge..999778d0) then  ! Above critical point
        print '(a)', '  temperature is supercritical.'
        return
      endif
      go to 84
70    i111=0
      call toolbx(MOVETO,10,245)
      write(*,103)
103   format('  saturated vapor properties ')
      d=dvv
      go to 84

*************************************************************************
* iopt = 1, input temperature & pressure                                 *
* iopt = 2, input temperature and pressure                               *
* iopt = 3, input temperature and entropy                                *
* iopt = 4, input temperature and enthalpy                               *
*************************************************************************
71    goto (72,73,74,76),iopt
72    pres=x/ps               ! temperature & pressure input
      p=pres/fp
      if(p.gt.2000.) then
      write(*,104)
104   format('    pressure is too far outside of range of equation.')
      goto 56
      endif
      dvv=2.d3*ps*p/(t*ts*ds)
      dgss=dvv
      if(p.gt.psat) dgss=dll
      if(psat.ge.1.d3) dgss=dvv
      if(psat.gt.0..and.p.ge.psat .and. t.lt..8d0) dgss=3.5d0-1.5d-6*t
      call dfind(d,p,dgss,t,dpdd)
      dd=d*ds/fd
      goto 84
73    dd=x                    ! Temperature & density input
      d=dd*fd/ds
      if(t.lt.tz .and. d.lt.dll .and. d.gt.dvv) go to 90
      call pdp(d,t,p,dpd)
      pres = fp*p
      dpdd=dpd*ps/ds
      go to 84
74    ss=x                    ! Temperature & entropy input
      sx=ss/fh/ft
      if(t.lt.tz .and. sx.lt.sv .and. sx.gt.sl) go to 90
      dg=1.d0
      if(t.ge.tz .and. sx.gt.9.d0) dg=.01d0
      if(t.lt.tz .and. sx.gt.sv) dg=dvv*.9d0
      if(t.lt.tz .and. sx.lt.sl)  dg=dll*1.01d0
```

```
75    call therm(dg,t)
      ddsdd=-dpdt*1000./ds/dg
      dels=sx-sd
      dg=dg*(1.+.9*dels/ddsdd)
      if(dg.gt.4.d0) dg=4.d0
      if(dg.lt.1.d-6) dg=1.d-6
      if(dabs(dels).gt.1.d-6)goto 75
      d=dg
      call pdp(d,t,p,dpd)
      pres=p*fp
      go to 84
76    hh=x                    ! Temperature & enthalpy input
      hx=hh/fh
      if(t.lt.tz .and. hx.lt.hv .and. hx.gt.hl) go to 90
      dg=.3
      if(t.ge.tz .and. hx.gt.1.d3) dg=.01
      if(t.lt.tz .and. hx.gt.hv)  dg=dvv*.9
      if(t.lt.tz .and. hx.lt.hl)  dg=dll*1.01
77    dg1=dg*1.001
      call therm(dg1,t)
      h0=hd
      call therm(dg,t)
      ddhdd=1000.*(h0-hd)
      dh=hx-hd
      dg=dg*(1.+dh/ddhdd)
      if(dg.lt.1.d-6) dg=1.d-6
      if(dabs(dh).gt.1.d-6)goto 77
      d=dg
      call pdp(d,t,p,dpd)
      pres=p*fp
      go to 84
78    pres=yy/ps  ! Calc. internal pressure from pressure input
      pp=pres/fp
      tg=1.1
      dg=1.
      dll=3.5d0-.2d0*t
      dvv=0.
      if(pp.gt..99724d0) go to 79
      call tcorr(tts,pp,dll,dvv)
      tt=tti(tts)
      pps=pp*fp
      sv=sd
      hv=hd
      call therm(dll,tts)
      sl=sd
      hl=hd
      t=tts
      psat=pp
79    if(iopt-6) 80,82,57
80    hh=x                          ! iopt = 5; Pressure & enthalpy input
      hx=hh/fh
      if(pp.gt..99724d0) go to 81   ! Pressure is supercritical
      if(hx.gt.hl .and. hx.lt.hv) goto 90
      if(hx.le.hl) dg=1.02*dll
      if(hx.ge.hv) dg=.9*dvv
      tg=tts
81    continue
      call dfind(d,pp,dg,tg,dq)
```

```
      dg=d
      call therm(d,tg)
      dhdt=cpd*ts
      dt=(hx-hd)/dhdt
      tg=tg+dt
      if(dabs(hx-hd).gt.1.d-5) goto 81
      t=tg
      tt=tti(t)
      go to 84
82    ss=x                          ! iopt = 6; Pressure & entropy input
      sx=ss/fh/ft
      if(pp.gt..99724d0) go to 83   ! Pressure is supercritical
      if(sx.gt.sl .and. sx.lt.sv) goto 90
      if(sx.le.sl) dg=1.02*dll
      if(sx.ge.sv) dg=.9*dvv
      tg=tts
83    continue
      call dfind(d,pp,dg,tg,dq)
      dg=d
      call therm(d,tg)
      dsdt=cpd/tg
      dt=(sx-sd)/dsdt
      tg=tg+dt
      if(dabs(sx-sd).gt.1.d-6) goto 83
      t=tg
      tt=tti(t)

*************************************************************************
* Call subroutine 'calc' to calculate final property values based on   *
* density (d) and temperature (t) and arranges them in array 'prop'.   *
*************************************************************************
84    call calc(d,t,prop)
      prop(1)=tt
      prop(2)=pres*ps
      call qtz(d,tt,prop(2),sio2)
      sio2p=10**sio2*60*1000

* Temporary storage of property and unit in external file
* 'outdata'; this file is re-read and then deleted
        open(15,file = 'outdata')
        do 85 i = 1,18
        write(15,105) prop(i), nunt(i)
105     format( g10.4,a8)
85      continue
        write(15,106) sio2
106     format( g10.4)
        write(15,107) sio2p
107     format( g10.4)
        rewind(15)
        do 86 i = 1,18
        read(15,108) stringp(i), stringn(i)
108     format( a10,a8)
        stringp(i) = stringp(i)(0:)
86      continue
        read(15,109) stringp(19)
        read(15,109) stringp(20)
109     format( a10)
        stringp(19) = stringp(19)(0:)
```

```
        stringp(20) = stringp(20)(0:)
        close(15,status = 'delete')

*************************************************************************
* Set up dialog box containing primary output                           *
*************************************************************************
87      dialogID = 271
        dStorage = 0
        dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
        call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(1)))
        call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringn(1)))
        call toolbx(GETDITEM,dialogptr,8,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(2)))
        call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringn(2)))
        call toolbx(GETDITEM,dialogptr,11,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(3)))
        call toolbx(GETDITEM,dialogptr,12,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringn(3)))
        call toolbx(GETDITEM,dialogptr,14,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(4)))
        call toolbx(GETDITEM,dialogptr,15,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringn(4)))
        call toolbx(GETDITEM,dialogptr,17,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(19)))
        call toolbx(GETDITEM,dialogptr,19,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(20)))
        call toolbx(GETPORT, grafptr)
        call toolbx(SETPORT, dialogptr)
        call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
        call toolbx(PENSIZE,3,3)
        call toolbx(INSETRECT,Rect,-4,-4)
        call toolbx(FRAMEROUNDRECT,Rect,16,16)

        do
        call toolbx(MODALDIALOG,0,itemHit)
        select case (itemHit)

          case(1)                          ! Continue to next calulation
            call toolbx(SETPORT, grafptr)
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(ERASERECT, windrect)
            call toolbx(MOVETO,0,225)
            go to 56

          case(2)                          ! QUIT, exit program
            call toolbx(CLOSEDIALOG,dialogptr)
            doneflag = .true.
            call toolbx(SETPORT, grafptr)
            go to 96

          case(3)                          ! Expand, display additional output
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(SETPORT, grafptr)
            call toolbx(ERASERECT, windrect)
            call toolbx(MOVETO,0,225)
```

```
          go to 88

        case(21)                        ! Return to menu selections
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT, grafptr)
          call toolbx(ERASERECT, windrect)
          call toolbx(MOVETO,0,225)
          go to 96
        case default
      end select            ! End primary output selection
      repeat                ! End do for primary output

*************************************************************************
* Open dialog box to display expanded output                            *
*************************************************************************
88    dialogID = 272
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(stringp(1)))
      call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(stringn(1)))

      do 89  i = 2,18
        if(i.eq.2) j = 9
        if(i.eq.3.or.i.eq.4) go to 89
        if(i.eq.5) j = 12
        if(i.eq.6) j = 15
        if(i.eq.7) j = 36
        if(i.eq.8) j = 21
        if(i.eq.9) j = 18
        if(i.eq.10) j = 24
        if(i.eq.11) j = 27
        if(i.eq.12) j = 50
        if(i.eq.13) j = 45
        if(i.eq.14) j = 30
        if(i.eq.15) j = 42
        if(i.eq.16) j = 47
        if(i.eq.17) j = 33
        if(i.eq.18) j = 39

        call toolbx(GETDITEM,dialogptr,j,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(i)))
        if(i.eq.12.or.i.eq.13) go to 89
        call toolbx(GETDITEM,dialogptr,j+1,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringn(i)))
89    continue
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

      do                                ! Get user preferences
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)
```

```
          case(1)                          ! Continue
            call toolbx(SETPORT, grafptr)
            call toolbx(CLOSEDIALOG,dialogptr)
            go to 56

          case(2)                          ! Return
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(SETPORT, grafptr)
            go to 87

          case(3)                          ! Quit
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(SETPORT, grafptr)
            doneflag = .true.
            go to 96

          case(4)                          ! Menu
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(SETPORT, grafptr)
            go to 96

          case(51)                         ! Print
            err1 = gpdump(grafptr,1)

          case default
        end select
        repeat
      if(i111.eq.1) go to 70
      go to 66

*************************************************************************
* iopt = 1, input temperature and pressure                               *
* iopt = 2, input temperature and density                                *
* iopt = 3, input temperature and entropy                                *
* iopt = 4, input temperature and enthalpy                               *
* iopt = 5, input pressure and enthalpy                                  *
* iopt = 6, input pressure and entropy                                   *
* iopt = 7, input temperature or pressure along saturation curve         *
*************************************************************************
90    go to (57,91,92,93,93,92,57),iopt
91    q = (1./d-1./dll)/(1./dvv-1./dll)  ! input temperature and pressure
      go to 94
92    q = (sx-sl)/(sv-sl)  ! input temperature or pressure and entropy
      go to 94
93    q = (hx-hl)/(hv-hl)  ! input temperature or pressure and enthalpy
94    v = q/dvv+(1-q)/dll
      d = 1./v
      sd = q*sv+(1-q)*sl
      hd = q*hv+(1-q)*hl
      h = hd*fh
      s = sd*fh*ft
      dd=d*ds/fd
      pps=psat*ps*fp

* Temporary storage of property and unit in external file
* 'outdata'; this file is re-read and then deleted
      open(15,file = 'outdata')
      write(15,110) q
```

```
      write(15,110) tt
      write(15,110) pps
      write(15,110) dd
      write(15,110) s
      write(15,110) h
110   format( g9.3)
      rewind(15)
      do 95 i = 1,6
      read(15,111) stringp(i)
111   format( a9)
      stringp(i) = stringp(i)(0:)
95    continue
      close(15)

*************************************************************************
* Display dialog box containing expanded output                         *
*************************************************************************
        dialogID = 277
        dStorage = 0
        dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
        call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(1)))
        call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(2)))
        call toolbx(GETDITEM,dialogptr,8,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(nunt(1)))
        call toolbx(GETDITEM,dialogptr,10,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(3)))
        call toolbx(GETDITEM,dialogptr,11,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(nunt(2)))
        call toolbx(GETDITEM,dialogptr,13,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(4)))
        call toolbx(GETDITEM,dialogptr,14,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(nunt(3)))
        call toolbx(GETDITEM,dialogptr,16,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(5)))
        call toolbx(GETDITEM,dialogptr,17,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(nunt(6)))
        call toolbx(GETDITEM,dialogptr,19,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(stringp(6)))
        call toolbx(GETDITEM,dialogptr,20,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(nunt(5)))
        call toolbx(GETPORT, grafptr)
        call toolbx(SETPORT, dialogptr)
        call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
        call toolbx(PENSIZE,3,3)
        call toolbx(INSETRECT,Rect,-4,-4)
        call toolbx(FRAMEROUNDRECT,Rect,16,16)
        do                          ! Get user preferences
        call toolbx(MODALDIALOG,0,itemHit)
        select case (itemHit)

          case(1)                        ! Continue to next calculation
            call toolbx(SETPORT, grafptr)
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(ERASERECT, windrect)
            go to 56
```

```
          case(2)                        ! QUIT, exit program
            call toolbx(CLOSEDIALOG,dialogptr)
            doneflag = .true.
            call toolbx(SETPORT, grafptr)
            go to 96

          case(21)                       ! Return to menu selections
            call toolbx(CLOSEDIALOG,dialogptr)
            call toolbx(SETPORT, grafptr)
            call toolbx(ERASERECT, windrect)
            go to 96
          case default
        end select
        repeat    ! End user preference selection
96      return          ! End loop for user preference
        end

*************************************************************************
* block data                                                            *
*************************************************************************
      block data
      implicit double precision(a-h,o-z)
      common/coefs/a0(18),a1(5),a20,y(4),a3(36),r(4),tee(4),a4(4)
     &,k(36),l(36),m(4),n(4)
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)
      data a0/-.130840393653D2,-.857020420940D2, .765192919131D-2,
     &-.620600116069D0,-.106924329402D2,-2.80671377296D0,
     & .119843634845D3,-.823907389256D2, .555864146443D2,
     &-31.0698122980D0, 13.6200239305D0,-4.57116129409D0,
     & 1.15382128188D0,-.214242224683D0, .0282800597384D0,
     & -.250384152737D-2, .132952679669D-3, -.319277411208D-5/
      data a1/ 1.538305300D0, -.8104836700D0, -6.830574800D0,
     & 0.D0, .86756271D0/, a20/ 4.2923415D0/
      data y/ .59402227D-1, -.28128238D-1, .56826674D-3,-.27987451D-3/
      data a3/-7.6221190138079D0, 32.661493707555D0, 11.305763156821D0,
     &-1.0015404767712D0,  .12830064355028D3, -.28371416789846D3,
     & .24256279839182D3, -.99357645626725D2, -.12275453013171D4,
     & .23077622506234D4, -.16352219929859D4,  .58436648297764D3,
     & .42365441415641D4, -.78027526961828D4,  .38855645739589D4,
     &-.91225112529381D3, -.90143895703666D4,  .15196214817734D5,
     &-.39616651358508D4, -.72027511617558D3,  .11147126705990D5,
     &-.17412065252210D5,  .99918281207782D3,  .33504807153854D4,
     &-.64752644922631D4,  .98323730907847D4,  .83877854108422D3,
     &-.27919349903103D4,  .11112410081192D4, -.17287587261807D4,
     &-.36233262795423D3,  .61139429010144D3,  .32968064728562D2,
     & .10411239605066D3, -.38225874712590D2, -.20307478607599D3/
      data k/4*1,4*2,4*3,4*4,4*5,4*6,4*7,4*9,3,3,1,5/
      data l/ 1, 2, 4, 6, 1, 2, 4, 6, 1, 2, 4, 6, 1, 2, 4, 6, 1, 2,
     & 4, 6, 1, 2, 4, 6, 1, 2, 4, 6, 1, 2, 4, 6, 0, 3, 3, 3/
      data r/ 3*1.0038928D0,4.8778492D0/
      data tee/ 2*.98876821D0,.99124013D0,.41713659D0/
      data a4/-.32329494D-2,-.024139355D0,.79027651D-3,-1.3362857D0/
      data m,n/2,2,2,4,0,2,0,0/
      data ts,ds,ps/647.27D0,317.763D0,22.115D0/,z0/0.317763D0/
      data ass,sss,rss/69.5958938D0,.1075222D0,77.32738D0/
      data alpha/34.D0,40.D0,30.D0,1050.D0/,beta/2*2.D4,4.D4,25.D0/
      end
```

```
************************************************************************
* subroutine setbox                                                    *
* subroutine to correctly label dialog box with correct parameter and  *
* unit. All handles and pointers are passed from h2o2.                 *
************************************************************************
      subroutine setbox (dialogptr,itemType,Handle,Rect,
     &par1,par2,par3,par4)

      integer*4 dialogptr, Handle
      integer*4 toolbx

      integer*2 itemType, Rect(4)

      include :include files:dialog.inc
      include :include files:file.inc
      include :include files:textedit.inc
      include :include files:resource.inc
      include :include files:control.inc
      include :include files:quickdraw.inc
      include :include files:window.inc

      character*13 par
      character*13 par1,par3
      character*8 par2,par4
      character*256 str255

      call toolbx(GETDITEM,dialogptr,3,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(par1))
      call toolbx(GETDITEM,dialogptr,8,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(par2))
      call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(par3))
      call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(par4))

      return
      end



************************************************************************
* subroutine qtz                                                       *
*                                                                      *
* Subroutine 'qtz' calculates the solubility of quartz (sio2) at the   *
* temperature (tt), pressure (pres) & density (dd) of interest,        *
* based on the empirical equation of Fournier (1983)                   *
************************************************************************
        subroutine qtz(dd,tt,pres,sio2)
        double precision ft,fd,fp,fh
        real*8 a,b,c,d,e,sio2,tt,tke,prs,pres,dd,ddl
        character*8 nt,nd,np,nh
        common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh

* it = temperature units
*       it = 1, °K;      it = 2, °C (default)
*       it = 3, °R;      it = 4, °F
        goto (50,51,52,53),it  ! Convert to K
50      tke = tt
```

```
        go to 54
51      tke = tt + 273.15
        go to 54
52      tke = tt/1.8
        go to 54
53      tke = (tt+459.67)/1.8

* ip = pressure units
*       ip = 1, mpascals;      ip = 2, bar (default)
*       ip = 3, atmospheres;   ip = 4, psia
54      go to (55,56,57,58,59), ip  ! Convert to bar pressure
55      prs = pres * 10
        go to 60
56      prs = pres
        go to 60
57      prs = pres * 1.01325
        go to 60
58      prs = pres * 0.0689476
        go to 60
59      prs = pres * 0.9800665
60      dd1 = dd * .31774   ! convert internal density to mg/cm3
        a=-1.01142
        b=1.79547
        c=-983.077
        d=.00176910
        e=.0173200
        sio2=a+b*log10(dd1)+(c+d*tke**2)/tke+e*prs/tke
100     format( 'sio2=',g10.4,2x,'t=',g10.4,2x,'p=',g10.4,2x,'d=',g10.4)
        return
        end

*************************************************************************
* Subroutine 'calc' calculates final property values based on density  *
* (d) and temperature (t) and arranges them in array 'prop'.           *
*************************************************************************
      subroutine calc(d,t,prop)
      implicit double precision(a-h,o-z)
      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)
      common /props/ p,ad,gd,hd,ud,sd,cvd,cpd,dpdd,dpdt,dvdt,w,cjtt,cjth
      character*8 nt,nd,np,nh
      dimension prop(20)
      dd=d*ds/fd
      call therm(d,t)
      u = ud*fh
      if(id.eq.4) w=w*3.280833
      h = hd*fh
      s = sd*fh*ft
      cp=cpd*fh*ft
      cv=cvd*fh*ft
      vl=1./dd
      a = ad*fh
      gg = gd*fh
      dpdd1 = dpdd*fd*fp
      comp = 1./dd/dpdd1
      dpdt1=dpdt*fp*ft
      call transp(t,d,dpdt,dpdd,visc,thcond)
      prop(3) = dd
```

```
      prop(4 )= vl
      prop(5) = h
      prop(6) = s
      prop(7) = u
      prop(8) = cv
      prop(9) = cp
      prop(10) = a
      prop(11) = gg
      prop(12) = dpdd1
      prop(13) = dpdt1
      prop(14) = comp
      prop(15) = w
      prop(16) = thcond
      if(it.eq.4) prop(16) = thcond*.5777892
      prop(17) = visc
      if(it.eq.4) prop(17) = visc*1.450377D-4
      prop(18) = cjth*ft*fd/fp
      return
      end

*************************************************************************
* Subroutine 'iso' manages the accounting for the table format.         *
* name = property name; nunit = property unit; ipi & itot = individual  *
* & total number of dependent variables; doneflag = 'true' when finished; *
* variable = 'true' when variable list is user selected; llopt & jjopt = *
* table type flags; widrect = window coordinates; linel & ct = print flags *
* ***********************************************************************
      subroutine iso(name,nunit,ipi,doneflag,jjopt,variable,
     &itot1,windrect,llopt,linel,ct)
      implicit double precision(a-h,o-z)

      include :include files:dialog.inc
      include :include files:file.inc
      include :include files:textedit.inc
      include :include files:resource.inc
      include :include files:control.inc
      include :include files:quickdraw.inc
      include :include files:window.inc

      character*8 nt,nd,np,nh,name(20),nunit(20),init
      character*3 i2ph,ibl,ifl,icr,isc
      character*255 string, title(20)
      character*256 str255
      character*19 string1
      character*17 string2
      character*15 string3
      character*10 inc
      character*6 fin

* Dialog box and window parameters
      integer*4 dialogptr           ! Pointer to dialog box
      integer*4 dialogID            ! Resource id# of dialog template
      integer*4 dStorage            ! Pointer to storage usage
      integer*4 Handle              ! Handle to resource item list
      integer*4 behind              ! Specifies window placement
      integer*2 itemType            ! Diaolog item reference number
      integer*2 itemHit             ! Handle to individual resource item
      integer*2 Rect(4)             ! Window boundary
```

```
      integer*2 penloc(2)           ! Pen location in local coordinates

      logical doneflag,variable     ! routing flags

* Specify the tool box interface as an interger function
      integer*4 toolbx

* Variables
      integer ipi,pos,dh1,dv1,opt,line,ct,line1

      dimension props(20),ipi(4), pos(10)

      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh
      common /fcts/ ad,gd,sd,ud,hd,cvd,cpd,dpdt,dvdt,cjtt,cjth
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)
      common /flag/ iopt,iopt7,kopt,lopt,tvalue,itot,jopt

* Initialize variables
      parameter (init = 'Initial ')
      parameter (fin = 'Final ')
      parameter (inc = ' increment')
      parameter (string1 = 'Initial temperature')
      parameter (string2 = 'Final temperature')
      parameter (string3 = 'Temp. increment')
      data ib1,if1/'   ','2PH'/,ISC/'CR*'/
      title(1) = 'Isothermal table'
      title(2) = 'Isobaric table'
      title(3) = 'Isochoric table'
      title(4) = 'Saturation table in temperature'
      title(5) = 'Saturation table in pressure'
      name(19) = ' SiO2   '
      nunit(19) = 'ppm     '
      variable = .false.
      line = 0
      Handle = 0
      behind = -1
      ProcPtr = 0
      dialogptr = 0
      itemHit = 0
      itemType = 0
      iopt = 1lopt
      jopt = jjopt
      itot = itot1
      i2ph=ib1
      icr=ib1
50    continue
      ip3 = ipi(3)
      ip4 = ipi(4)
      ip5 = ipi(5)
      if(ip3.le.0) return  ! Invalid entry, no variables selected
51    continue

*************************************************************************
*     oopt = 1, Isothermal table (default)                              *
*               jopt = 2, temperature & pressure input      (default)   *
*             jopt = 3, temperature & density input                     *
*     iopt = 2, Isobaric table                                          *
*     iopt = 3, Isochoric table                                         *
```

```
*     iopt = 4, Saturaton table in temperature                          *
*     iopt = 5, Saturaton table in pressure                             *
*************************************************************************
52    if(iopt.gt.3) go to 55 ! 'goto' for saturation tables

*************************************************************************
* Set dialog box so that user can input data for independent variables  *
* for an isothermal table in pressure or density                        *
*************************************************************************
      dialogID = 292
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(title(lopt)))
      call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(name(iopt)))
      call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(nunit(iopt)))

*************************************************************************
* Set dialog box for user input to isobaric or isochoric table          *
*************************************************************************
      if(iopt.eq.2.or.iopt.eq.3) jopt = 1
      call toolbx(GETDITEM,dialogptr,8,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(init // name(jopt)))
      call toolbx(GETDITEM,dialogptr,10,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(fin // name(jopt)))
      call toolbx(GETDITEM,dialogptr,12,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(name(jopt) // inc))
      call toolbx(GETDITEM,dialogptr,15,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(nunit(jopt)))
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

53    do                        ! Retrieve data from dialog box
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

        case(1)           ! Continue, get data
          xiso = 0.; y1 = 0.; y2 = 0.; yi = 0.
          t1 = 0.; t2 = 0.; p1 = 0.; p2 = 0.
          string = 0
          call toolbx(GETDITEM,dialogptr,6,itemType,Handle,Rect)
          call toolbx(GETITEXT,Handle,string)
          string = string(2:)
          read(string,100) xiso

          if(xiso.le.0)  then       ! Warn user of null entry
            call toolbx(ALERT,204,0)
            go to 53
          endif

          do 54 i = 9,13,2          ! Retrieve values
            string = 0
```

```
            call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
            call toolbx(GETITEXT,Handle,string)
            string = string(2:)
            if(i.eq.9) read(string,100) y1      ! Initial value
            if(i.eq.11) read(string,100) y2     ! Final value
            if(i.eq.13) read(string,100) yi     ! Increment
54        continue

          if(y1.le.0.or.y2.le.0.or.yi.le.0.or.y2.le.y1) then
            call toolbx(ALERT,204,0)    ! Warn user of null entry
            go to 53
          endif

          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT, grafptr)
          exit

        case(2)                         ! QUIT, exit program
          call toolbx(CLOSEDIALOG,dialogptr)
          doneflag = .true.
          call toolbx(SETPORT,grafptr)
          return

        case(3)                         ! Return to menu selections
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          return

        case(14)                        ! Return to variable selections
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          variable = .true.
          return

        case(16)                         ! table variables
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          variable = .true.
          return

        case default
      endselect                          ! End dialog box selection
      repeat                             ! End loop for dialog box

55    call toolbx(TEXTSIZE,10)
      if(iopt.gt.3) then                 ! Other than an isothermal table

*************************************************************************
* Set up dialog box for user to input data to create a table           *
* with correct headings and units                                      *
*************************************************************************
      dialogID = 293
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
*************************************************************************
* Set up dialog box for input to saturation table in temp. or pres.    *
*************************************************************************
      if(iopt.eq.4) then
```

```
        opt = 1          ! Saturation table in temperature
        call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(string1))
        call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(string2))
        call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
        call toolbx(SETITEXT,Handle,str255(string3))
      else
        opt = 2          ! Saturation table in pressure
      endif
      call toolbx(GETDITEM,dialogptr,12,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(nunit(opt)))
      call toolbx(GETDITEM,dialogptr,4,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(title(iopt)))
      call toolbx(GETDITEM,dialogptr,5,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(init // name(opt)))
      call toolbx(GETDITEM,dialogptr,7,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(fin // name(opt)))
      call toolbx(GETDITEM,dialogptr,9,itemType,Handle,Rect)
      call toolbx(SETITEXT,Handle,str255(name(opt) // inc))
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

56    do                       ! Event loop to retrieve data
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

        case(1)                              ! Continue
          t1 = 0.; t2 = 0.; p1 = 0.; p2 = 0.

          do 57 i = 6,10,2                   ! Get data from appropriate box
            string = 0
            call toolbx(GETDITEM,dialogptr,i,itemType,Handle,Rect)
            call toolbx(GETITEXT,Handle,string)
            string = string(2:)
            if(i.eq.6) read(string,100) y1
            if(i.eq.8) read(string,100) y2
            if(i.eq.10) read(string,100) yi
100         format(f30.0)
57        continue

          if(y1.le.0.or.y2.le.0.or.yi.le.0.or.y2.le.y1)  then
            call toolbx(ALERT,204,0)         ! Warn user of null values
            go to 56
          endif
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT, grafptr)
          exit

        case(2)                    ! QUIT, and exit program
          call toolbx(CLOSEDIALOG,dialogptr)
          doneflag = .true.
          call toolbx(SETPORT,grafptr)
          return
```

```
        case(3)                  ! Menu
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          return

        case(11)                 ! Variables
          call toolbx(CLOSEDIALOG,dialogptr)
          call toolbx(SETPORT,grafptr)
          variable = .true.
          return

        case default
      endselect                  ! End retrieve data selection
      repeat                     ! End loop for retrieve data

      endif

      call toolbx(TEXTFONT,1)    ! Return to application font
      call toolbx(TEXTSIZE,10)

*************************************************************************
*     iopt = 1, Isothermal table (default)                              *
*               jopt = 2, temperature & pressure input      (default)   *
*            jopt = 3, temperature & density input                      *
*     iopt = 2, Isobaric table                                          *
*     iopt = 3, Isochoric table                                         *
*     iopt = 4, Saturaton table in temperature                          *
*     iopt = 5, Saturaton table in pressure                             *
*************************************************************************
      go to (58,69,72,76,82),iopt
58    continue                 ! Isothermal table
      if(jopt-1.1e.0) return ! Invalid for isothermal table
59    continue
      tt=xiso                  ! temperature
      t=ttt(tt)
      if(jopt.eq.2) dgss=y1/fp/t/.4
      iz=0
      pss=200.
      dvv=0.
      dll=0.
      if(t.ge..999778d0) go to 60   ! Greater then critical point
      call pcorr(t,pss,dll,dvv)

*************************************************************************
* jopt = 2; isothermal table in pressure                                *
* jopt = 3; isothermal table in density                                 *
*************************************************************************
      if(jopt.ne.3) go to 60
      dll1=ds*dll/fd
      dvv1=ds*dvv/fd

*************************************************************************
* Print table headings for isothermal table                             *
*************************************************************************
      call toolbx(MOVETO,0,245)
      write(*,103)tt,nunit(1),dll1,nunit(3),dvv1,nunit(3)
60    dgss=dvv
```

```
      call toolbx(MOVETO,5,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,75,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
*************************************************************************
* itot = 1; one dependent variable (ip3) tabulated                      *
* itot = 2; two dependent variables (ip3, ip4) tabulated                *
* itot = 3; three dependent variables (ip3,ip4,ip5) tabulated           *
*************************************************************************
      if(itot.eq.1) go to 61              ! one dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 61         ! two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
61    continue
      dh1 = 13
101   format( a8,1x,a8)
      if(dgss.eq.0.) dgss=3.5-.2*t
      psat=pss*fp*ps
      if(jopt.eq.2 .and. y1.gt.psat) iz=3   ! single phase
      if(y1.gt.psat) dgss=dll
      if(jopt.ge.3) iz=3

*************************************************************************
* Increment pressure or density for isothermal table                    *
* jopt = 2, increment pressure; jopt = 3, increment density             *
* y1 = pres. or dens. start; y2 = pres. or dens. end; yi = increment    *
*************************************************************************
      pin=y1-yi
      din=pin
      pinc=yi/fp/ps
      if(iopt.eq.2) pinc=0.
      dinc=yi*fd/ds
62    if(jopt.eq.2) pin=pin+yi       ! input incremented pressure
      if(jopt.eq.3) din=din+yi       ! input incremented density
      if(jopt.eq.2 .and. pin.gt.y2 .and. yi.gt.0.) return  ! finished, + inc.
      if(jopt.eq.3 .and. din.gt.y2 .and. yi.gt.0.) return  ! finished, + inc.
      if(jopt.eq.2 .and. pin.lt.y2 .and. yi.lt.0.) return  ! finished, - inc.
      if(jopt.eq.3 .and. din.lt.y2 .and. yi.lt.0.) return  ! finished, - inc
      if(jopt.eq.2) pres=pin/fp/ps
      if(jopt.le.2) go to 63
      d=din*fd/ds
      if(pss.gt.0. .and. d.lt.dll .and. d.gt.dvv) i2ph=ifl
63    continue
      if (iopt.eq.1) tss=t
      if(jopt.eq.3 .or. (jopt.eq.2 .and. pin.lt.psat)) go to 65
      if(jopt.eq.1 .and. t.lt.tss) go to 65
      tsave=tt-yi
      if(jopt.eq.1 .and. iopt.eq.2 .and. iz.le.2) tt=tti(tss)
      psave=pin-yi
      if(jopt.eq.2 .and. pres.gt.pss .and. iz.ge.2) go to 65
      if(jopt.eq.1 .and. t.gt.tss .and. iz.ge.2) go to 65
64    iz=iz+1
      if(jopt.eq.2) pres=pss
      if(jopt.eq.1) t=tss
```

```
      if(iz.eq.1 .and. jopt.eq.1) d=dll     ! liquid density
      if(iz.eq.1 .and. jopt.eq.2) d=dvv     ! vapor density
      if(iz.eq.2 .and. jopt.eq.2) d=dll     ! liquid density
      if(iz.eq.2 .and. jopt.eq.1) d=dvv     ! vapor density
      if(iz.lt.3) go to 66
65    if(iopt.ne.3 .and. jopt.ne.3) call dfind(d,pres,dgss,t,dq)
66    call pdp(d,t,pdum,dq)
      if(jopt.eq.3 .or. iopt.eq.3) pres=pdum
      if(iopt.eq.3) pinc=0.
      dgss = d
      if (jopt.eq.2) dgss=d + pinc/dq

* Calculate properties for isothermal table
      call calc(d,t,props)
      props(1)=tt
      props(2)=pres*ps*fp
      call qtz(d,props(1),props(2),sio2)
      sio2p=10**sio2*60*1000
      props(19) = sio2p
      if(i2ph.eq.ifl) then
      props(9)=0.
      props(15)=0.
      props(18)=0.
      endif
      line = line + 1
      line1 = line1 +1
      if(line.ge.16) then ! Present print option before lines scroll off page

*************************************************************************
* Set up dialog box to allow the option of printing table               *
*************************************************************************
      dialogID = 261
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

      do                        ! Print option loop
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

        case(1)           ! Continue,resume table output
           call toolbx(CLOSEDIALOG,dialogptr)
           call toolbx(SETPORT,grafptr)
           exit

        case(2)                 ! print, call print function
           call toolbx(CLOSEDIALOG,dialogptr)
           call toolbx(SETPORT,grafptr)
           err1 = gpdump(grafptr,1)
           exit

         case default
      end select        ! End options selection
```

```
      repeat                    ! End loop for options

*************************************************************************
* Print table heading to monitor                                        *
* Print only total number of dependent variables (itot)                 *
*************************************************************************
      call toolbx(ERASERECT,windrect)
      call toolbx(MOVETO,5,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,75,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 67        ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 67        ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
67    continue
      dh1 = 13
      line = 1
      endif
      dh1 = dh1 + 13
      dv1 = 0
      if(iopt.eq.2.or.iopt.eq.3) dv1 = 20

*************************************************************************
* Print table of properties to monitor                                  *
*************************************************************************
      write(*,102) props(iopt)
      call toolbx(MOVETO,80+dv1,dh1)
      write(*,102) props(jopt)
      call toolbx(MOVETO,185,dh1)
      write(*,102) props(ip3)
      if(itot.eq.1) go to 68        ! One dependent variable
      call toolbx(MOVETO,290,dh1)
      write(*,102) props(ip4)
      if(itot.eq.2) go to 68        ! Two dependent variables
      call toolbx(MOVETO,395,dh1)
      write(*,102) props(ip5)
      i2ph=ibl
68    continue
102   format(g12.5)
103   format(//' the coexisting densities for t =',f7.2,2x,a8,' are',f9.4
     &,2x,a8,' and',f9.4,2x,a8)
104   format( ' ------------------------------------------------',
     &'--------------')
      if(iz.eq.1) go to 64              ! Get new parameters
      if(iz.eq.2 .and. jopt.eq.2) pin=psave
      if(iz.eq.2 .and. jopt.eq.1) tt=tsave
      if(iz.eq.2) iz = 3

*************************************************************************
*     iopt = 1, Isothermal table (default)                              *
*               jopt = 2, temperature & pressure input      (default)   *
*            jopt = 3, temperature & density input                      *
*     iopt = 2, Isobaric table                                          *
```

```
*     iopt = 3, Isochoric table                                              *
*************************************************************************
      go to (62,71,74),iopt    ! Routing via variable increment
69    jopt=1                   ! Isobaric table, routed directly from input
      pres = xiso/fp/ps
      pinc=0.
      psat=pres
      pss=pres
      pin=psat
      t1 = y1
      t2 = y2
      if(t1.le.-1.) stop
      if(t1.eq.0. .and. t2.eq.0.) return

* Display table headings for isobaric table
      call toolbx(MOVETO,3,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,95,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 70        ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 70        ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
70    continue
      dh1 = 13
      tt=t1-yi
      t=ttt(tt)
      dll=0.
      dvv=0.
      tss=0.
      if(pres.lt.1.) call tcorr(tss,pres,dll,dvv)
      d=1.
      tp=ttt(t1)
      if((tp.lt.tss.and.tss.gt.0.).or.(pres.ge.1..and.tp.lt.1.))then
        dlx=0.
        dvx=0.
        call pcorr(tp,pdum,dlx,dvx)
        d=dlx*1.01
      endif
      if(tp.gt.tss .and. tss.gt.0.) d=dvx*.9
      iz=3
      if(tp.lt.tss)iz=0
71    tt=tt+yi
      t=ttt(tt)
      if(tt.gt.t2 .and. yi.gt.0.) return
      if(tt.lt.t2 .and. yi.lt.0.) return
      dgss=d
      go to 63
72    jopt=1           ! isochoric table, routed directly from input
      d=xiso*fd/ds
      psat=0.
      pinc=0.
      t1 = y1
      t2 = y2
```

```
      if(t1.le.-1.) stop
      if(t1.eq.0. .and. t2.eq.0.) return

* Display table headings for isochoric table
      call toolbx(MOVETO,3,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,95,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 73              ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 73
      call toolbx(MOVETO,390,13)          ! Two dependent variables
      write(*,101) name(ip5),nunit(ip5)
73    continue
      dh1 = 13
      tt=t1-yi
      iz=3
      t=ttt(tt)
      i2ph=ibl
74    tt=tt+yi
      if (yi.ge.0. .and. tt.gt.t2) return
      if (yi.lt.0. .and. tt.lt.t2) return
      t=ttt(tt)
      if(t.ge..999775d0) go to 75
      call pcorr (t,psat,dl,dv)
      if(d.gt.dv .and. d.lt.dl) i2ph=ifl
75    go to 66
76    continue        ! Saturation table in temperature, routed from input
      t1 = y1
      t2 = y2
      if(t1.eq.0. .and. t2.eq.0.) return
      iopt=1
      jopt=2

* Display table headings for saturation table
      call toolbx(MOVETO,3,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,75,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 77         ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 77         ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
77    continue
      dh1 =  13
      tt=t1-yi
      dl=0.
      dv=0.
78    tt=tt+yi
      if (yi.ge.0. .and. tt.gt.t2) return
      if (yi.lt.0. .and. tt.lt.t2) return
```

```
      t=ttt(tt)
      if(t.ge..999775d0) return   ! If critical temperature exceeded, return
      call pcorr(t,p,dl,dv)       ! get pressure & density at temperature

* Calculate properties of liquid along temperature saturation curve
      call calc(dl,t,props)
      props(1)=tt
      props(2)=p*ps*fp
      call qtz(dl,props(1),props(2),sio2)
      sio2p=10**sio2*60*1000
      props(19) = sio2p
      prp3=props(ip3)
      prp4=props(ip4)
      prp5=props(ip5)
      line = line + 1
      if(line.gt.5) then   ! Present print option before data scroll off page
      dialogID = 261
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

      do                          ! Print option loop
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

          case(1)         ! Continue,resume table output
             call toolbx(CLOSEDIALOG,dialogptr)
             call toolbx(SETPORT, grafptr)
             exit

           case(2) ! print
             call toolbx(CLOSEDIALOG,dialogptr)
             call toolbx(SETPORT, grafptr)
             err1 = gpdump(grafptr,1)
             exit

          case default
        end select            ! End print option selection
      repeat                  ! End loop for print option

      call toolbx(ERASERECT,windrect)
      call toolbx(MOVETO,3,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,75,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 79        ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 79        ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
```

```
79    continue
      dh1 = 13
      line = 1
      endif
      dh1 = dh1 + 13

* print properties of liquid along temperatrure saturation curve
      write(*,102) props(iopt)
      call toolbx(MOVETO,80,dh1)
      write(*,102) props(jopt)
      call toolbx(MOVETO,185,dh1)
      write(*,102) props(ip3)
      if(itot.eq.1) go to 80        ! One dependent variable
      call toolbx(MOVETO,290,dh1)
      write(*,102) props(ip4)
      if(itot.eq.2) go to 80        ! Two dependent variables
      call toolbx(MOVETO,395,dh1)
      write(*,102) props(ip5)
80    continue

* Calculate vapor properties along temperature saturation curve
      call calc(dv,t,props)
      call qtz(dv,props(1),props(2),sio2)
      sio2p=10**sio2*60*1000
      props(19) = sio2p
      dh1 = dh1 + 13
      write(*,102) props(iopt)
      call toolbx(MOVETO,80,dh1)
      write(*,102) props(jopt)
      call toolbx(MOVETO,185,dh1)
      write(*,102) props(ip3)
      if(itot.eq.1) go to 81        ! One dependent variable
      call toolbx(MOVETO,290,dh1)
      write(*,102) props(ip4)
      if(itot.eq.2) go to 81        ! Two dependent variables
      call toolbx(MOVETO,395,dh1)
      write(*,102) props(ip5)
81    continue
      dh1 = dh1 + 10
      call toolbx(MOVETO,3,dh1)
      write (*,104)
      go  to 78
82    continue   ! Saturation table in pressure, routed directly from input
      p1 = y1
      p2 = y2
      pi = yi
      if(p1.eq.0. .and. p2.eq.0.) return
      iopt=2
      jopt=1

* Print properties of vapor along pressure saturation curve
      call toolbx(MOVETO,5,13)
      write(*,101) name(iopt),nunit(iopt)
      call toolbx(MOVETO,95,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 83        ! One dependent variable
```

```
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 83        ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
83    continue
      dh1 = 13
      pp=p1-pi
      dl=0.
      dv=0.
84    pp=pp+pi
      if(pp.gt.p2) return
      p=pp/(ps*fp)
      if (p.ge..9972d0) return
      call tcorr(t,p,dl,dv)

* Calculate table of liquid properties along pressure saturation curve
      call calc(dl,t,props)
      props(2)=pp
      props(1)=tti(t)
      call qtz(dl,props(1),props(2),sio2)
      sio2p=10**sio2*60*1000
      props(19) = sio2p
      line = line + 1
      if(line.gt.5) then
      dialogID = 261
      dStorage = 0
      dialogptr = toolbx(GETNEWDIALOG,dialogID,dStorage,behind)
      call toolbx(GETPORT, grafptr)
      call toolbx(SETPORT, dialogptr)
      call toolbx(GETDITEM,dialogptr,1,itemType,Handle,Rect)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,Rect,-4,-4)
      call toolbx(FRAMEROUNDRECT,Rect,16,16)

      do                        ! Print option loop
      call toolbx(MODALDIALOG,0,itemHit)
      select case (itemHit)

         case(1)          ! Continue,resume table output
           call toolbx(CLOSEDIALOG,dialogptr)
           call toolbx(SETPORT, grafptr)
           exit

         case(2) ! print
           call toolbx(CLOSEDIALOG,dialogptr)
           call toolbx(SETPORT, grafptr)
           err1 = gpdump(grafptr,1)
           exit

          case default
       end select       ! End print option selection
      repeat                    ! End loop for print option

* Print table headings
      call toolbx(ERASERECT,windrect)
      call toolbx(MOVETO,5,13)
      write(*,101) name(iopt),nunit(iopt)
```

```
      call toolbx(MOVETO,95,13)
      write(*,101) name(jopt),nunit(jopt)
      call toolbx(MOVETO,180,13)
      write(*,101) name(ip3),nunit(ip3)
      if(itot.eq.1) go to 85        ! One dependent variable
      call toolbx(MOVETO,285,13)
      write(*,101) name(ip4),nunit(ip4)
      if(itot.eq.2) go to 85        ! Two dependent variables
      call toolbx(MOVETO,390,13)
      write(*,101) name(ip5),nunit(ip5)
85    continue
      dh1 = 13
      line = 1
      endif
      dh1 = dh1 + 13
      dv1 = 20

* Print table of liquid properties along saturation curve
      write(*,102) props(iopt)
      call toolbx(MOVETO,100,dh1)
      write(*,102) props(jopt)
      call toolbx(MOVETO,185,dh1)
      write(*,102) props(ip3)
      if(itot.eq.1) go to 86        ! One dependent variable
      call toolbx(MOVETO,290,dh1)
      write(*,102) props(ip4)
      if(itot.eq.2) go to 86        ! Two dependent variables
      call toolbx(MOVETO,395,dh1)
      write(*,102) props(ip5)
86    continue

* Calculate & print out table of vapor properties along saturation curve
      call calc(dv,t,props)
      call qtz(dv,props(1),props(2),sio2)
      sio2p=10**sio2*60*1000
      props(19) = sio2p
      dh1 = dh1 + 13
      dv1 = 20
      write(*,102) props(iopt)
      call toolbx(MOVETO,100,dh1)
      write(*,102) props(jopt)
      call toolbx(MOVETO,185,dh1)
      write(*,102) props(ip3)
      if(itot.eq.1) go to 87        ! One dependent variable
      call toolbx(MOVETO,290,dh1)
      write(*,102) props(ip4)
      if(itot.eq.2) go to 87        ! Two dependent variables
      call toolbx(MOVETO,395,dh1)
      write(*,102) props(ip5)
87    continue
      dh1 = dh1 + 10
      call toolbx(MOVETO,3,dh1)
      write (*,104)
      go to 84
88    stop
      end

*************************************************************************
```

```
* Subroutine 'unit' converts unit-option flags to character names      *
************************************************************************
      subroutine unit

      implicit double precision(a-h,o-z)
      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh

      dimension ffh(6),ffd(6),ffp(6)
      integer it,id,ip,ih
      character*8 np,nt,nd,nh,nnt(4),nnp(4),nnd(4),nnh(6)

      data nnt/'°K      ','°C      ','°R      ','°F      '/
      data nnd/'kg/m3   ','g/cm3   ','mol/l   ','lb/ft3  '/
      data nnp/'mpa     ','bar     ','atm     ','psia    '/
      data nnh/'kj/kg   ','j/g     ','j/mol   ','cal/g   ','cal/mol '
     1,'btu/lb '/
      data ffh/2*1.d0,18.0152d0,.2388459d0,4.3028566d0,.42992262d0/
      data ffp/1.d0,10.d0,9.8692327d0,145.03d0,2*.1d0/
      data ffd/1.d0,1000.d0,18.0152d0,16.01846d0,2*0.d0/

      np=nnp(ip)
      fp=ffp(ip)
      nt=nnt(it)
      nd=nnd(id)
      fd=ffd(id)
      nh=nnh(ih)
      fh=ffh(ih)
      return
   99 stop
      end

************************************************************************
* Function 'ttt' converts input temperatures to reduced temperatures.  *
* It and the following 4 functions and subroutines are only slightly   *
* modified from those provided in the latest updated version of Haar,  *
* Gallagher, and Kell (1984).                                          *
************************************************************************
      function ttt(t)
      implicit double precision(a-h,o-z)

      common /params/ tr,dr,pr,z0,ass,sss,rss,alpha(4),beta(4)
      common /units/ it,id,ip,ih,ft,fd,fp,fh,/unitc/nt,nd,np,nh

      character*8 nd,nh,np,nt
      integer it,id,ip,ih

************************************************************************
* it = temperature units; ft converts to local temperature units       *
*       it = 1, °K;      it = 2, °C (default)                          *
*       it = 3, °R;      it = 4, °F                                    *
************************************************************************
      go to (1,2,3,4),it
    1 ttt=t/tr
      ft=1.
      return
    2 ttt=t+273.15
      ft=1.
      ttt=ttt/tr
```

```
      return
   3  ttt=t/1.8
      ft=5./9.
      ttt=ttt/tr
      return
   4  ttt=(t+459.67)/1.8
      ft=5./9.
      ttt=ttt/tr
      return
      end

************************************************************************
* Function 'tti' converts internal reduced temperatures to external    *
* units.                                                               *
************************************************************************
      function tti(t)

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

      common /params/ tr,dr,pr,z0,ass,sss,rss,alpha(4),beta(4)
      common /units/ it,id,ip,ih,ft,fd,fp,fh

      character*8 nd,nh,np,nt
      integer it,id,ip,ih

************************************************************************
* it = temperature units; ft converts to local temperature units       *
*       it = 1, °K;      it = 2, °C (default)                          *
*       it = 3, °R;      it = 4, °F                                    *
************************************************************************
      go to (5,6,7,8),it
   5  tti = t*tr
      return
   6  tti = t*tr-273.15
      return
   7  tti = t*tr*1.8
      return
   8  tti = t*tr*1.8d0 - 459.67
      return
      end


************************************************************************
* function 'pv' calculates an approxmation to the vapor pressure       *
* based on the input temperature (t)                                   *
************************************************************************
      function pv(t)

      double precision pv,t,a,w,b,z,q

      dimension a(8)
      data a/-7.8889166,2.5514255,-6.716169,33.239495,
     & -105.38479,174.35319,-148.39348,48.631602/

      pv=0.
      if(t.gt..9997775d0) return   ! above critical point
      w=dabs(1.d0-t*.9997d0)
      b=0.
```

```
      do 50 i=1,8
      z=i
50    b=b+a(i)*w**((z+1.)/2.)
      q=b/t
      pv=1.001*exp(q)
      return
      end

*************************************************************************
* function 'tsat' calculates the saturation temperature based on the   *
* input pressure (p)                                                    *
*************************************************************************
      function tsat(p)

      double precision p,tsat,tg,pv,tdpsdt,pl,pp,dp

      tsat=0.
      if(p.gt.1.) return
      k=0
      pl=7.1226152+2.302585*dlog(p)
      tg=.57602+pl*(.042887+pl*(.00368+pl*(3.837e-4+pl*3.e-5)))
   1  if(tg.lt..422d0) tg=.422d0
      if(tg.gt..9996d0) tg=.9996d0
      if(k.lt.8) go to 2
      go to 8
   2  k=k+1
      pp=pv(tg)
      dp=tdpsdt(tg)
      if(dabs(1.-pp/p).lt..00001) go to 8
      tg = tg*(1.+(p-pp)/dp)
      go to 1
   8  tsat=tg
      return
      end

*************************************************************************
* function 'tdsdt' calculates t*(dps/dt) for use by tsat               *
*************************************************************************
      function tdpsdt(t)

      double precision t,tdpsdt,a,w,b,c,y,q

      dimension a(8)
      data a/-7.8889166,2.5514255,-6.716169
     &,33.239495,-105.38479,174.35319,-148.39348
     &,48.631602/

      if(t.ge.1.d0) t=.99999d0
      w=1.d0-t
      b=0.d0
      c=0.d0
      do 50 i=1,8
      z=i
      y=a(i)*w**((z+1.d0)/2.d0)
      c=c+y/w*(.5d0-.5d0*z-1.d0/t)
50    b=b+y
      q=b/t
      tdpsdt=exp(q)*c
```

```
      return
      end

*************************************************************************
* subroutine 'therm' is a dimensionless version of the Haar, Gallagher,*
* and Kell equation of state by Kestin, Sengers, Kamgar-Parsi, and     *
* Levelt Sengers.                                                      *
*************************************************************************
      subroutine therm(d,t)

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

      common /props/ p,f,g,h,u,s,cv,cp,dpdd,dpdt,dvdt,w,cjtt,cjth
      common/coefs/a0(18),a1(5),a20,y(4),a3(36),rho(4),tee(4),a4(4)
     &,k(36),l(36),m(4),n(4)
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)

      call pdp(d,t,pr,dpddr)
      ti=1./t
      tln=dlog(t)
      f0=(a0(1)+a0(2)*t)*tln
      s0=-a0(1)*ti-a0(2)*tln-a0(2)
      c0=-a0(2)+a0(1)*ti
      do 50 i=3,18
        term=a0(i)*t**(i-6)
        f0=f0+term*t
        s0=s0-term*(i-5)
50    c0=c0-term*(i-5)*(i-6)
      f1=0.d0
      s1=0.d0
      c1=0.d0
      do 51 i=1,5
        term=d*a1(i)*ti**(i-2)
        f1=f1+term
        s1=s1+term*(i-2)*ti
51    c1=c1-term*(i-2)*(i-1)*ti
      dpt1=-s1/d
      yy=d*(y(1)+y(2)*tln+y(3)*ti**3+y(4)*ti**5)
      dydt=d*(y(2)*ti-3.*y(3)*ti**4-5.*y(4)*ti**6)
      d2ydt2=d*(-y(2)*ti**2+12.*y(3)*ti**5+30.*y(4)*ti**7)
      y1=1.-yy
      y2=130./(3.*y1)
      y3=169./(6.*y1*y1)
      p2=a20/(d*y1) * (1.-y2*yy+2.*y3*yy-14.*y1*yy)
      dpt2=p2+a20*t/y1/y1 *(-127./3.+(169.-260.*yy)/(3.*y1) +
     & 169.*yy/(y1**2) -14.*y1*y1) *dydt/d
      f2=a20*t*(dlog(d/y1)-y2+y3-14.*yy)
      s2=-f2/t-a20*t/y1*(1.-y2+2.*y3-14.*y1)*dydt
      c2=-a20*t/y1*(1.-y2+2.*y3-14.*y1)*(2.*dydt+t*d2ydt2)
     &   -a20*(t*dydt/y1)**2*(1.-2.*y2+6.*y3)
      z=1.-dexp(-z0*d)
      f3=0.d0
      s3=0.d0
      c3=0.d0
      dpt3=0.d0
      do 52 i=1,36
        term=a3(i)*ti**l(i)*z**(k(i)-1)
        f3=f3+term*z
```

```
        s3=s3+term*l(i)*ti*z
        dpt3=dpt3-l(i)*k(i)*term*ti
52      c3=c3-term*l(i)*(l(i)+1)*ti*z
        dpt3=dpt3*z0*(1.-z)
        f4=0.d0
        p4=0.d0
        dpd4=0.d0
        s4=0.d0
        c4=0.d0
        dpt4=0.d0
        do 53 i=1,4
          if(t.gt..6d0 .and. i.eq.4) goto 53
          if((t.lt..96d0 .or. t.gt.1.05d0) .and. i.le.3) goto 53
          del=(d-rho(i))/rho(i)
          if(dabs(del).lt.1.d-8) del=1.d-8
          tau=(t-tee(i))/tee(i)
          deln=del**n(i)
          term=a4(i)*dexp(-alpha(i)*del**m(i)-beta(i)*tau**2)
          f4=f4+term*deln
          p4term=term*deln/del*(n(i)-m(i)*alpha(i)*del**m(i))/rho(i)
          p4=p4+p4term
          dpd4=dpd4+term*deln/(rho(i)*del)**2*(n(i)*(n(i)-1)-
     1         alpha(i)*m(i)*(m(i)+2.*n(i)-1.)*del**m(i)+
     2         (alpha(i)*m(i)*del**m(i))**2)
         s4=s4+term*2.*beta(i)*tau*deln/tee(i)
         dpt4=dpt4-2.*beta(i)*tau*p4term/tee(i)
         c4=c4+term*2.*t*(beta(i)-2.*(beta(i)*tau)**2)*deln/tee(i)**2
53      continue
        dpd4=2.*d*p4+d*d*dpd4
        f=f0+f1+f2+f3+f4
        s=s0+s1+s2+s3+s4
        cv=c0+c1+c2+c3+c4
        s=s*sss
        f=f*ass
        u=f+t*ts*s
        cv=cv*sss
        p=pr*ps
        dpdd=dpddr*ps/ds
        h=u+1.e3*p/(d*ds)
        g=f+1.e3*p/(d*ds)
        dpdt=dpt1+dpt2+dpt3+dpt4
        dpdt=dpdt*d*d*ps/ts
        dvdt=0.
        cp=0.
        w=0.
        cjtt=0.
        cjth=0.
        if(dpdd.le.1.d-5) goto 54
        dvdt=dpdt/dpdd/d/d/ds/ds
        cp=cv+t*ts*dpdt**2*1.e3/(d*d*ds*ds*dpdd)
        w=dsqrt(dabs(1.d6*cp*dpdd/cv))
        cjtt=1./(d*ds)-t*ts*dvdt
        cjth=-cjtt/cp
54      return
        end

*************************************************************************
* Subroutine 'transp' calculates the transport properties: viscosity & *
```

```
* thermal conductivity                                                 *
************************************************************************
      subroutine transp(tt,dd,dpdt,dpdd,visc,thcond)

      double precision tt,dd,dpdt,dpdd,visc,thcond

      data tref,dref/647.27,317.763/

      ts=tt
      ts1=1./ts
      r1=dd
      dr=dd-1.
      dt1=ts1-1.
      chi=.0695959*dd/dpdd
      call visc85(ts1,r1,dt1,dr,chi,visc0,visc)
      dt=ts-1.
      delth=0.
      if(dd.lt.3.) call anathc(ts,r1,dt,dr,visc0,dpdt,chi,delth)
      call condbk(ts1,r1,dr,dt1,thc,delth)
      thcond=thc
      return
      end

************************************************************************
* Subroutine 'condbk' calculates background thermal conductivity       *
* using the 1985 iaps form                                             *
************************************************************************
      subroutine condbk(redt,r1,dr,dt,thcond,delth)

      dimension b(5,6)
      data b/1.3293046,1.7018363,5.2246158,8.7127675,-1.8525999,
     1    -0.40452437,-2.2156845,-10.124111,-9.5000611,.9340469,
     2     .24409490,1.6511057,4.9874687,4.3786606,0.,
     3       .018660751,-.76736002,-.27297694,-.91783782,0.,
     4     -.12961068,.37283344,-.43083393,0.,0.,
     5      .044809953,-.1120316,.13333849,0.,0./
      data a0,a1,a2,a3/1.0,6.978267,2.599096,-.998254/

      thcond = a0+a1*redt+a2*redt**2+a3*redt**3
      thcond = sqrt(1./redt)/thcond
      sum = 0.
      do 50 i=1,5
      do 50 j=1,6
50    sum = sum+b(i,j)*dr**(j-1)*dt**(i-1)
      thcond = thcond*exp(r1*sum)
      thcond = (thcond+delth)*.4945
      return
      end

************************************************************************
* Subroutine 'anathc' calculates theAnamolous part of thermal          *
* conductivity from the iaps 1985 formulation                          *
************************************************************************
      subroutine anathc(ts,r1,dt,dr,visc,dpdt,chi,delth)

      double precision dpdt

      data tref,rref,pref/647.27,317.763,22.115/
```

```
      data a,b,c,omega/18.66,1.00,1.3848e-3,0.4678/

      dpdtr=dpdt*tref/pref
      if(dpdtr.lt.0.) dpdtr=0.
      chir=abs(chi)**omega
      ex=-a*dt**2-b*dr**4
      fac=abs(ts/r1*dpdtr)**2*chir*sqrt(r1)
      delth=c*fac*exp(ex)/visc
      return
      end

*************************************************************************
* Subroutine 'visc85' calculates viscosity according to the iaps 1985  *
* formulation                                                           *
*************************************************************************
      subroutine visc85(ts1,r1,dt1,dr,chi,visc0,visc)

      double precision visc

      dimension a(4),b(6,7)
      data a/1.0,.978197,.579829,-.202354/
      data b/.5132047,.3205656,2*0.,-.7782567,.1885447,.2151778,
     & .7317883,1.241044,1.476783,2*0.,-.2818107,-1.070786,-1.263184,
     & 3*0.,.1778064,.4605040,.2340379,-.4924179,2*0.,-.0417661,2*0.,
     & .1600435,3*0.,-.01578386,7*0.,-.003629481,2*0./

      sum=0.
      do 50 k=1,4
50    sum=sum+a(k)*ts1**(k-1)
      visc=sqrt(1./ts1)/sum
      sum=0.
      do 51 i=1,6
        dt1i=dt1**(i-1)
      do 51 j=1,7
51    sum=sum+b(i,j)*dt1i*dr**(j-1)
      visc0=visc*exp(r1*sum)
      v2=1.
      t=1./ts1
      if(t.lt..997 .or. t.gt.1.0082 .or. dr.lt.-.245 .or. dr.gt..29
     1 .or. chi.lt.21.93) go to 52
      v2=.922*chi**(.0263)
52    visc = 55.071e-6*visc0*v2
      return
      end

*************************************************************************
* Subroutine 'corr' calculates liquid and vapor densities at the given *
* temperature and pressure. delg = (gl-gv)/rt is calculated for use in *
* correcting vapor pressures for delg = 0.                              *
*************************************************************************
      subroutine corr(t,p,dl,dv,delg)

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

      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)
      common /props/ pp,ad,gd,hd,ud,sd,cvd,cpd,dpdd,dpdt,dvdt,w,cjt,cjh

      if(t.lt..42d0) then
```

```
       write(*,100)

100    format(' temperature is below valid range for equation of state')
       p = 0.
       return
      endif
      if(t.gt..9985d0) go to 50     ! Temperature is above critical point
      tau=(1.d0-t)**(.333333d0)
      dliq=1.+1.99206d0*tau+1.10123d0*tau**2-.512506d0*tau**5
      call dfind(dl,p,dliq,t,dq)
      call therm(dl,t)
      gl=gd
      dvap=dexp(-2.02957d0*tau-2.68781d0*tau**2-5.38107d0*tau**4)
      call dfind(dv,p,dvap,t,dq)
      if(dv.lt.5.d-7) dv=5.d-7
      call therm(dv,t)
      gv=gd
      delg = (gl-gv)/ass
      return
50    p=0.
      if(t.gt..9997775d0) return
      delg=0.
      tau=2.06798d0*(1.-t/.9997775)**.325
      dl=1.012988d0+tau
      dv=1.012988d0-tau
      call pdp(dv,t,p,dpd)
      call therm(dv,t)
      return
      end

*************************************************************************
* Suroutine 'pcorr' calculates the saturated liquid and vapor densities *
* and the vapor pressure at the given saturation temperature.           *
*************************************************************************
      subroutine pcorr(t,p,dl,dv)

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

      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)

      tol=5.d-7
      if(t.gt..5d0) tol=1.d-6
      p = 0.d0
      if(t.gt..9997775d0) return
      p = pv(t)
      if(t.lt..997d0) goto 50
      tau=2.06798*(1.-t/.9997775)**.325
      dl=1.015+tau
      dv=1.010-tau
50    call corr(t,p,dl,dv,delg)
      if(dl.lt..95) dl=1.01299+tau
      if(dv.gt..93) dv=1.01299-tau
      dp=delg/(1./dv-1./dl)
      if(t.gt..997d0) dp=dp*.5
      if(dabs(dp).gt.1.d-2) dp=1.d-2*dabs(dp)/dp
      p = p+dp
      if(dabs(dp/p).lt.tol) return
      goto 50
```

```
      end

*************************************************************************
* Subroutine 'tcorr' calculates liquid and vapor densities along the   *
* pressure saturation curve and the saturation temperature at the      *
* saturation pressure                                                  *
*************************************************************************
      subroutine tcorr(t,p,dl,dv)

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

      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)

      t = tsat(p)
      if(t.eq.0.d0) return
      if(t.gt..9985d0) go to 51
50    call corr(t,p,dl,dv,delg)
        dp=delg/(1./dv-1./dl)
        t = t*(1.-dp/tdpsdt(t))
        if(dabs(dp/p).lt.5.d-7) return
      go to 50
51    call pcorr(t,pp,dl,dv)
        dp=pp-p
        t = t*(1.-dp/tdpsdt(t))
        if(dabs(dp).lt.1.d-6) return
      go to 51
      end



*************************************************************************
* suroutine 'dfind' calculates the density (dout) in the single phase  *
* based on the input temperature (t) and pressure (p) from an initial  *
* density guess (d). Dp/drho (dpd) is also calculated                  *
*************************************************************************
      subroutine dfind(dout,p,d,t,dpd)

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

      tol=1.0d-7
      if(t.lt..7) tol=1.0d-6
      dd=d
      l=0
50    l=l+1
      if(dd.le.1.d-8) dd=1.d-8
      if(dd.gt.5.5d0) dd=5.5
      call pdp(dd,t,pp,dpd)
      if(dpd.gt.0.) go to 51
      if(d.ge..9337d0) dd=dd*1.02
      if(d.lt..9337d0) dd=dd*.98
      if(l.le.10) goto 50
51    dpdx=dpd*1.1
      if(dpd.lt..01d0) dpdx=.01d0
      if(dabs(1.d0-pp/p).lt.tol) go to 52
      x=(p-pp)/dpdx
      if(dabs(x).gt..3d0) x=x*.3/dabs(x)
      dd=dd+x
      if(dd.le.1.d-8) dd=1.d-8
```

```
      if(l.le.30) go to 50
      dd=dd-x
      if(dabs(1.d0-pp/p).gt.1.d-3) write(*,100) p,pp,d,dd,t,x,dpd
52    continue
      dout=dd
100   format(' 30 iterations in dfind: pin pcalc din dcalc t frac
     &dpd',4e13.6/3e13.6)
      return
      end

*****************************************************************************
* Subroutine 'pdp' calculates pressure (p) and dp/dd (dpdd) from            *
* temperature (t) and density (d) input                                     *
*****************************************************************************
      subroutine pdp(d,t,p,dpdd)

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

      common/coefs/a0(18),a1(5),a20,y(4),a3(36),rho(4),tee(4),a4(4)
     &,k(36),l(36),m(4),n(4)
      common /params/ ts,ds,ps,z0,ass,sss,rss,alpha(4),beta(4)

      ti=1./t
      tln=dlog(t)
      f1=0.d0
      do 50 i=1,5
        term=d*a1(i)*ti**(i-2)
50    f1=f1+term
      p1=f1/d
      dpd1=2.*d*p1
      yy=d*(y(1)+y(2)*tln+y(3)*ti**3+y(4)*ti**5)
      y1=1.-yy
      y2=130./(3.*y1)
      y3=169./(6.*y1*y1)
      p2=a20*t/(d*y1)*(1.-y2*yy+2.*y3*yy-14.*y1*yy)
      dpd2=d*p2+a20*t*yy*(-127./(3.*y1*y1)-260.*yy/(3.*y1**3)
     &     +169./(3.*y1**3)+169.*yy/y1**4-14.)
      z=1.-dexp(-z0*d)
      p3=0.
      dpd3=0.
      do 51 i=1,36
        term=a3(i)*ti**l(i)*z**(k(i)-1)
        p3=p3+term*z0*(1.-z)*k(i)
51    dpd3=dpd3-term*k(i)*z0*z0*(1.-z)*(k(i)-(k(i)-1.)/z)
      dpd3=2.*d*p3+d*d*dpd3
      p4=0.
      dpd4=0.
      do 52 i=1,4
        if(t.gt..6d0 .and. i.eq.4) goto 52
        if((t.lt..96d0 .or. t.gt.1.05d0) .and. i.le.3) goto 52
        del=(d-rho(i))/rho(i)
        if(dabs(del).lt.1.d-8) del=1.d-8
        tau=(t-tee(i))/tee(i)
        deln=del**n(i)
        term=a4(i)*dexp(-alpha(i)*del**m(i)-beta(i)*tau**2)
        p4=p4+term*deln/del*(n(i)-m(i)*alpha(i)*del**m(i))/rho(i)
        dpd4=dpd4+term*deln/(rho(i)*del)**2*(n(i)*(n(i)-1)-
     &       alpha(i)*m(i)*(m(i)+2.*n(i)-1.)*del**m(i)+
```

```
     &       (alpha(i)*m(i)*del**m(i))**2)
52    continue
      dpd4=2.*d*p4+d*d*dpd4
      p=p1+p2+p3+p4
      dpdd=dpd1+dpd2+dpd3+dpd4
      p=p*d**2
      return
      end


*************************************************************************
*  Function 'gpdump' prints a grafport to the printer.                  *
*       error = gpdump(grafptr,lc)                                      *
* where lc is specifies either square pixels (lc = 1) or the            *
* Imagewriter's default rectangular pixels (lc = 0).  This should be 1  *
* for the LaserWriter.  Error will contain a zero if the print was      *
* successful. This function was slightly modified after one provided    *
* by Absoft for the 020 FORTRAN compilier.                              *
*************************************************************************
        integer function gpdump(grafptr,lc)

        implicit none
        integer grafptr
        integer lc
        integer toolbx
        integer line1,ct

        include :include files:file.inc
        include :include files:misc.inc
        include :include files:params.inc
        include :include files:grafport.inc

* Constant print driver reference number.
        integer iprdrvrref
        parameter (iprdrvrref = -3)

* Equivalence control parameters specific to PrCtlCall control calls
* over the generic control parameter byte array.
        integer iprdevctl
        equivalence (csparam(1),iprdevctl)

* Eqivalence control parameters specific to the bitmap print control call
* over the generic control parmeter byte array.
        integer pbitmap,pportrect,lcontrol
        equivalence (csparam(1),pbitmap)
        equivalence (csparam(5),pportrect)
        equivalence (csparam(9),lcontrol)

        character*7 filename            ! Used to build print driver name.

        integer err                     ! Error code for file calls.
        integer i                       ! scratch
        integer paramptr                ! Pointer to the parameter block.

* Clear the file parameter block.
        do i = 1, 80
          params(i) = 0
        repeat
```

```
        paramptr = toolbx(PTR, params)  ! Pointer to parameter block.

* Open the printer port for output.
        filename = char(6) // '.PRINT'
        ionameptr = toolbx(PTR, filename)
        err = toolbx(PBOPEN, paramptr)
        if (err .ne. 0) goto 50

* Reset the printer.
        cscode = 7
        iprdevctl = z'00010000'
        err = toolbx(PBCONTROL, paramptr)
        if (err .ne. 0) goto 50


* Initialize the printer for a new page
*        if(line1.eq.1.or.mod(line1,50).eq.0) then
        cscode = 7
        iprdevctl = z'00040000'
        err = toolbx(PBCONTROL, paramptr)
        if (err .ne. 0) goto 50
*       endif

* Print the grafport's bitmap.
        cscode = 4
        iprdevctl = z'00000000'
        pbitmap = grafptr + portbits
        pportrect = grafptr + portrect
        lcontrol = lc
        err = toolbx(PBCONTROL, paramptr)
        if (err .ne. 0) goto 50

* End the page
*        if(line1.eq.1.or.mod(line1,50).eq.0) then
        cscode = 7
        iprdevctl = z'00020000'
        err = toolbx(PBCONTROL, paramptr)
        if (err .ne. 0) goto 50
*       endif

* Close the printer port.
        err = toolbx(PBCLOSE, paramptr)

50      gpdump = err

        return
        end
```