Appendix A - FORTRAN Code for LakeVOC Model

[An electronic file with the FORTRAN code for LakeVOC model and the executable file are available on the World Wide Web at URL http://water.usgs.gov/nawqa/vocs or http://pubs.water.usgs.gov/ofr03212]

PROGRAM LAKEVOC

```
  use msflib
  use dialogm
  use mtbecom
  use tser_com
  implicit none
  integer(kind=4)i4, i, iwin
  TYPE (QWINFO) winfo
  TYPE (windowconfig) wcinit
           iwin = getactiveqq()
  winfo.H    = 28
  winfo.W    = 80
  winfo.TYPE = QWIN$SET
  i = SETWSIZEQQ(iwin, winfo)
 ! Set the x & y pixels to 800X600 and font size to 8x12
           i = GETWINDOWCONFIG(wcinit)
 wcinit%numxpixels  = -1
 wcinit%numypixels  = -1
 wcinit%numtextcols = -1
 wcinit%numtextrows = -1
 wcinit%numcolors   = -1
 wcinit%title= "Time Series of VOC Concentration"C
 wcinit%fontsize = -1
 i = SETWINDOWCONFIG(wcinit)  ! attempt to set configuration with above values
! call Initialize_TimeSeries to set allocatable array sizes at 12 initially
! both subroutines are located in interp_f.f90
  do i = 1, 365
    year_time(i) = dble(i)
  end do
  call Initialize_TimeSeries
! call spline_data to set up the needed data interpolations for running the model
  do i = 1, numtimeseries
    call spline_data(i)
  end do
  call lake_volume_calc
  i4 = aboutboxqq('LakeVOC Version 2.85:  October 24, 2002 &

                   Includes gas exchange, mixing, biochemical degradation &

                   Inflow/Outflow used to estimate inter-layer exchange, VOC loss &

                   Accounts for Volume Lost due to Evaporation &

                   William Asher'

!* This is the main loop of the program.  It does nothing but
!* cycle endlessly, allowing the menus to be used.
  do while(.true.)
    call yieldqq
  end do
end


logical(kind=4) function InitialSettings
  use msflib
  use dialogm
  use mt
  use mtbecom
  implicit none
  type (qwinfo) qwi
  character(len=50)mname
  integer(kind=4) mnum, i
  external parfilin, parfilout, datfilout, mtbe_params, meteor_params, &
           hydrog_params, model_params, start_model, halt_model, &
           restore_default, exitprog, pause_model, continue_model, &
           timeseries_setup
 ! Set window frame size.
  qwi%x = 50
  qwi%y = 50
  qwi%w = 700
  qwi%h = 600
  qwi%type = QWIN$SET
  i = SetWSizeQQ( QWIN$FRAMEWINDOW, qwi )
  mnum = 1
  mname = '&File'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,nul)) then
    initialsettings = .false.
    return
  end if

  mname = '&Read VOC Parameter File'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,parfilin)) then
    initialsettings = .false.
    return
  end if

  mname = '&Write VOC Parameter File'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,parfilout)) then
    initialsettings = .false.
    return
  end if

  mname = '&Save VOC Model results'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,datfilout)) then
    initialsettings = .false.
    return
  end if

  mname = '&Print...'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,winprint)) then
    initialsettings = .false.
    return
  end if

  mname = 'E&xit'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,exitprog)) then
    initialsettings = .false.
    return
  end if

  mnum = 2
  mname = '&Setup'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,nul)) then
    initialsettings = .false.
    return
  end if

  mname = '&Time Series Setup'c
```

```
  if (.not.appendmenuqq(mnum,$menuenabled,mname,timeseries_setup)) then
    initialsettings = .false.
    return
  end if


  mname = '&VOC Parameters'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,mtbe_params)) then
    initialsettings = .false.
    return
  end if

  mname = 'Me&teorological Parameters'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,meteor_params)) then
    initialsettings = .false.
    return
  end if

  mname = '&Hydrographical Parameters'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,hydrog_params)) then
    initialsettings = .false.
    return
  end if

  mname = '&Runtime Parameters'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,model_params)) then
    initialsettings = .false.
    return
  end if

  mname = 'Restore &Default Parameters'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,restore_default)) then
    initialsettings = .false.
    return
  end if

!  mname = '&clear all parameters'c
!  if (.not.appendmenuqq(mnum,$menuenabled,mname,clear_params)) then
!    initialsettings = .false.
!    return
!  end if

  mnum = 3
  mname = '&Run'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,nul)) then
    initialsettings = .false.
    return
  end if

  mname = '&Start Model'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,start_model)) then
    initialsettings = .false.
    return
  end if

  mname = '&Pause Model'c
  if (.not.appendmenuqq(mnum,$menuenabled,mname,pause_model)) then
    initialsettings = .false.
    return
  end if
```

```
mname = '&Continue After Pause'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,continue_model)) then
  initialsettings = .false.
  return
end if

mname = 'S&top Model'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,halt_model)) then
  initialsettings = .false.
  return
end if

mnum = 4
mname = '&Window'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,nul)) then
  initialsettings = .false.
  return
end if

mname = '&Full Screen'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,winfullscreen)) then
  initialsettings = .false.
  return
end if

mname = '&Size to Fit'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,winsizetofit)) then
  initialsettings = .false.
  return
end if

mname = '&Cascade'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,wincascade)) then
  initialsettings = .false.
  return
end if

mname = '&Tile'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,wintile)) then
  initialsettings = .false.
  return
end if

mname = '&Arrange Icons'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,winarrange)) then
  initialsettings = .false.
  return
end if

mnum = 5
mname = '&Help'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,nul)) then
  initialsettings = .false.
  return
end if

mname = '&About VOC Model'c
if (.not.appendmenuqq(mnum,$menuenabled,mname,winabout)) then
  initialsettings = .false.
  return
  end if

!  if (.not.setwindowmenuqq(mnum)) then
!    initialsettings = .false.
!    return
!  end if

  initialsettings = .true.
  return



subroutine cfunc (t, c, d)
  use mtbecom
  use modelcom
  use tser_com
  implicit none
! num_eqs set in modelcom
! local real*8 variables, some are shorthand notation
  real*8 t, c(num_eqs), d(num_eqs), chyp, la, de, dh, depth, MLHeight, &
         deltavol, cs, el, hl, evap, evap_vol, kl_local
! external real*8 functions
  real*8 Calc_Inflow, Calc_Outflow, ch, csat, ii, la_func, ld, kl, mld, mldp,&
         EpiLoss, HypLoss,interpolate, depivol_dt, dhypvol_dt, epi_vol, hyp_vol,&
         Calc_InHeight, Calc_OutHeight,flux_h2o
! external function declarations
  external Calc_Inflow, Calc_Outflow, ch, csat, ii, la_func, ld, EpiLoss,&
           HypLoss,kl, mld, mldp, interpolate, depivol_dt, dhypvol_dt,&
           epi_vol, hyp_vol,Calc_InHeight, Calc_OutHeight, flux_h2o
   ! conc(1) = volume of epilimnion
   ! conc(2) = volume of hyplimnion
   ! conc(3) = Epilimnion concentration (mol m^-3)
   ! conc(4) = Hypolimnion concentration (mol m^-3)
   ! conc(5) = total mass in lake, unused outside of RKINTOUT
   ! d(1) = dVE/dt
   ! d(2) = dVH/dt
   ! d(3) = dC(Epi)/dt
   ! d(4) = dC(Hyp)/dt
   ! d(5) = dM(total)/dt
  kl_local = kl(t)
  el = EpiLoss(t)
  hl = HypLoss(t)
  cs = csat(t)
  depth = ld(t)
  la = la_func(t)
! thickness of the epilimnion
  de = mld(t)
! if epi depth = 0.0d0 then no mixed layer and epi depth is lake depth
! bug corrected 10/28/2002 - wea
      if (de .eq. 0.0d0) de = depth
! thickness of the hypolimnion
  dh = depth - de
! height of the mixed-layer above bottom = thickness of the hypolimnion
  MLHeight = dh
  chyp = 0.0
  InH = calc_InHeight(t)
  OutH = calc_OutHeight(t)
  if ((InH .gt. MLHeight) .and. (OutH .gt. MLHeight)) then
    case_flow = 1    ! inflow/outflow in the epilimnion
  elseif ((InH .gt. MLHeight) .and. (OutH .le. MLHeight)) then
```

```
    case_flow = 2    ! inflow in epilimnion, outflow in hyplimnion
  elseif ((InH .le. MLHeight) .and. (OutH .gt. MLHeight)) then
    case_flow = 3    ! inflow in hyplimnion, outflow in epilimnion
  elseif ((InH .le. MLHeight) .and. (OutH .le. MLHeight)) then
    case_flow = 4    ! inflow/outflow in hypolimnion
  endif
! lake volumes, c(1) = epilimnion, c(2) = hypolimnion
  c(1) = epi_vol(t)
  c(2) = hyp_vol(t)
! change in lake volumes
  d(1) = depivol_dt(t)
  d(2) = dhypvol_dt(t)
  deltavol = d(1) + d(2)
! calculate inflows and outflows, layer exchanges
  In = Calc_Inflow(t)
  Out = Calc_Outflow(t)
! calculations for conservation of volume using inflow and outflow.
! first calculate evaporation rate based on 50% RH
  evap = 0.001 * flux_h2o(t)
! flux_h2o returns evaporation in mm/day-m^2, to change to m/day-m^2
  evap_vol = la * evap
! for testing purposes, next two lines shut off evaporation and gas exchange
!  evap_vol = 0.0
!  kl_local = 0.0
! comment out previous two lines to run program
  select case (case_flow)
    case (1)
      iexchange = d(2)
      makeup = d(1) + iexchange - in + out + evap_vol
    case(2)
      iexchange = d(2) + out
      makeup = d(1) + iexchange - in + evap_vol
    case(3)
      iexchange = d(2) - in
!     d(2)=in+iexchange, iexchange<0:epi gaining, iexchange>0:epi losing
      makeup = d(1) + iexchange + out + evap_vol
    case (4)
      iexchange = d(2) - in + out
      makeup = d(1) + iexchange + evap_vol
  end select
  if (makeup .gt. 0.0) then
    mu_mass = makeup*cs
!   assumes added water equilibrated with atm. VOC conc.
  else
    mu_mass = makeup*c(3)     ! lost water through surface outlet
  endif
  if (c(2) .gt. 0.0) then   ! two-layer system with epilimnion and hypolimnion
    if (deltavol .ge. 0.0) then    ! lake gaining volume
      select case (case_flow)
        case (1) ! inflow/outflow in upper layer
          if (d(2) .gt. 0.0) then  ! epi gains, hyp gains
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-d(2)*c(3)-Out*c(3)+ &
                    mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (d(2)*c(3))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          elseif ((d(1) .gt. 0.0) .and. (d(2) .le. 0.0)) then
!           volume gained by epi > volume lost by hyp
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-d(2)*c(4)-Out*c(3)+ &
                    mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (d(2)*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          elseif ((d(1) .eq. 0.0) .and. (d(2) .eq. 0.0)) then
```

```
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-Out*c(3)+mu_mass)/c(1)-&
                    el*c(3)
            d(4) = -hl*c(4)
          endif
          d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)*c(1)-hl*c(4)*c(2)-&
                  Out*c(3)+mu_mass

        case (2) ! inflow in upper layer, outflow in lower layer
          if (d(2) .gt. 0.0) then
!           epi gains, hyp gains, iexchange must be > 0
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(3) +&
                    mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (iexchange*c(3)-Out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          elseif ((d(1) .gt. 0.0) .and. (d(2) .le. 0.0)) then
!           volume gained by epi > volume lost by hyp
            if (iexchange .le. 0.0) then
              d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(4)+ &
                      mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
              d(4) = (iexchange*c(4)-Out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            else
              d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(3)+ &
                      mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
              d(4) = (iexchange*c(3)-Out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            endif
          elseif ((d(1) .eq. 0.0) .and. (d(2) .eq. 0.0)) then
            if (iexchange .le. 0.0) then
              d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(4)+ &
                      mu_mass)/c(1) - el*c(3)
              d(4) = (iexchange*c(4)-Out*c(4))/c(2) - hl*c(4)
            else
              d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(3)+ &
                      mu_mass)/c(1) - el*c(3)
              d(4) = (iexchange*c(3)-Out*c(4))/c(2) - hl*c(4)
            endif
          endif
          d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)*c(1)-hl*c(4)*c(2)- &
                  Out*c(4)+mu_mass

        case (3)
!         inflow in lower layer, outfow in upper layer
!         assume no VOC in lower layer inflow
          if (d(2) .gt. 0.0) then  ! epi does whatever, hyp gains
            if (iexchange .le. 0.0) then
              d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)-Out*c(3)+ &
                      mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
              d(4) = iexchange*c(4)/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            else
              d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)-Out*c(3)+ &
                      mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
              d(4) = iexchange*c(3)/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            endif
          elseif ((d(1) .gt. 0.0) .and. (d(2) .le. 0.0)) then
!           iexchange must be < 0
            d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)-Out*c(3)+ &
                    mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (iexchange*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          elseif ((d(1) .eq. 0.0) .and. (d(2) .eq. 0.0)) then
            if (iexchange .le. 0.0) then
              d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)-Out*c(3)+&
                      mu_mass)/c(1) - el*c(3)
              d(4) = iexchange*c(4)/c(2) - hl*c(4)
            else
              d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)-Out*c(3)+ &
                      mu_mass)/c(1) - el*c(3)
              d(4) = iexchange*c(3)/c(2) - hl*c(4)
            endif
          endif
          d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)*c(1)-hl*c(4)*c(2)- &
                  out*c(3)+mu_mass

        case (4)
!         inflow in lower, outflow in lower
          if ((d(1) .ne. 0.0) .or. (d(2) .ne. 0.0)) then
!           epi gains, hyp gains
            if (iexchange .le. 0.0) then
              d(3)=(kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)+mu_mass)/c(1)-&
                    el*c(3) - c(3)*d(1)/c(1)
              d(4) = (iexchange*c(4)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            else
              d(3)=(kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)+mu_mass)/c(1)-&
                    el*c(3) - c(3)*d(1)/c(1)
              d(4) = (iexchange*c(3)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
            endif
          elseif ((d(1) .eq. 0.0) .and. (d(2) .eq. 0.0)) then
            if (iexchange .le. 0.0) then
              d(3)=(kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)+mu_mass)/c(1)-&
                    el*c(3)
              d(4) = (iexchange*c(4)-Out*c(4))/c(2) - hl*c(4)
            else
              d(3)=(kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)+mu_mass)/c(1)-&
                    el*c(3)
              d(4) = (iexchange*c(3)-Out*c(4))/c(2) - hl*c(4)
            endif
          endif
          d(5) = kl_local*la*(cs-c(3)) + ii(t) - el*c(3)*c(1) - hl*c(4)*c(2)-&
                  out*c(4) + mu_mass
      end select
!          end select for delta-volume >= 0
!          (lake gaining volume or volume constant)

    elseif (deltavol .lt. 0.0) then    ! lake losing volume
      select case (case_flow)
        case (1) ! inflow/outflow in upper layer
          if (d(2) .lt. 0.0) then  ! epi loses or constant, hyp loses
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-d(2)*c(4)-&
                    out*c(3)+mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (d(2)*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          elseif ((d(1) .lt. 0.0) .and. (d(2) .ge. 0.0)) then
!           epi loses, hyp gains
            d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-d(2)*c(3)-out*c(3)+ &
                    mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
            d(4) = (d(2)*c(3))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
          endif
          d(5) = kl_local*la*(cs-c(3))+ii(t)+In*cs-el*c(3)*c(1)- &
                  hl*c(4)*c(2)-out*c(3)+mu_mass
        case (2) ! inflow in upper layer, outflow in lower layer
          if (d(2) .lt. 0.0) then
!           hyp loses, epi gains or loses
            if (iexchange .le. 0.0) then
              d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(4)+&
           mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
       d(4) = (iexchange*c(4)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
      else
       d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(3)+ &
           mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
       d(4) = (iexchange*c(3)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
      endif
     elseif (d(2) .ge. 0.0) then
!     epi loses, hyp gains, iexchange must be > 0
      d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-iexchange*c(3)+ &
          mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
      d(4) = (iexchange*c(3)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
     endif
     d(5) = kl_local*la*(cs-c(3))+ii(t)+In*cs-el*c(3)*c(1)-hl*c(4)*c(2)-&
         out*c(4)+mu_mass
    case (3) ! inflow in lower layer, outfow in upper layer
     if (d(1) .lt. 0.0) then  ! epi loses, hyp loses
      if (iexchange .le. 0.0) then
       d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)-out*c(3)+&
           mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
       d(4) = (iexchange*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
      else
       d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)-out*c(3)+ &
           mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
       d(4) = (iexchange*c(3))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
      endif
     elseif (d(1) .ge. 0.0) then
!     epi same/gains, hyp loses, iexchange must be > 0
      d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)-out*c(3)+ &
          mu_mass)/c(1) - el*c(3) - c(3)*d(1)/c(1)
      d(4) = (iexchange*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
     endif
     d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)*c(1)-hl*c(4)*c(2)- &
         out*c(3)+mu_mass
    case (4)  ! inflow in lower, outflow in lower
     if (iexchange .le. 0.0) then
      d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(4)+mu_mass)/c(1)-&
          el*c(3) - c(3)*d(1)/c(1)
      d(4) = (iexchange*c(4)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
     else
      d(3) = (kl_local*la*(cs-c(3))+ii(t)-iexchange*c(3)+mu_mass)/c(1)-&
          el*c(3) - c(3)*d(1)/c(1)
      d(4) = (iexchange*c(3)-out*c(4))/c(2) - hl*c(4) - c(4)*d(2)/c(2)
     endif
     d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)*c(1)-hl*c(4)*c(2)-&
         out*c(4)+mu_mass
   end select ! end select for delta-volume < 0 (lake losing volume)
  endif   ! endif for deltavol if statement

 elseif (c(2) .le. 0.0) then  ! no mixed layer, unstratified lake
  if (c(2) .lt. 0.0) c(2) = 0.0
   if (deltavol .ge. 0.0) then
    d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-out*c(3)+mu_mass)/c(1) - &
        el*c(3) - c(3)*d(1)/c(1)
    d(4) = d(3)
    d(5) = kl_local*la*(cs-c(3))+ii(t)-el*c(3)+In*cs-Out*c(3)+mu_mass
    c(4) = c(3)
   elseif (deltavol .lt. 0.0) then
    d(3) = (kl_local*la*(cs-c(3))+ii(t)+In*cs-Out*c(3)+mu_mass)/c(1) - &
        el*c(3) - c(3)*d(1)/c(1)
        d(4) = d(3)
        d(5) = kl_local*la*(cs-c(3))+ii(t)+In*cs-Out*c(3)-el*c(3)+mu_mass
        c(4) = c(3)
      endif
    endif   ! endif for c(2) .gt. 0.0 if statement
  return
end subroutine cfunc


subroutine gout (time, conc)
  use msflib, setpixel0=>setpixel
  use dfmt
  use mtbecom
  use modelcom
  use errorcom
  use scigraph
  implicit none
  integer ipnts,numpts,ierr,i,j,iret
  real*8 time, conc(num_eqs), x(imaxpnts), y(isets,imaxpnts), u, degc, &
         airconc, yrtime, yr, convunits, maxtemp, DelHyp,&
         HypDepth, depth, lastdepth, tin, tout1, tout2, tout3, evap, la
  real*8 ii, kl, ld, mld, mldp, interpolate, csat, sol_calc, ch, flux_h2o, &
         la_func
  external ii, kl, ld, mld, mldp, interpolate, csat, sol_calc, ch, flux_h2o, &
           la_func
  ! conc(1) = Volume of epilimnion
  ! conc(2) = Volume of hypolimnion
  ! conc(3) = Epilimnion concentration (mol m^-3)
  ! conc(4) = Hypolimnion concentration (mol m^-3)
  ! conc(5) = total mass in lake, unused outside of RKINTOUT
    tin = time
    delHyp = -1.0*mldp(time)*OutputTimestep
    lastdepth = mld(time-outputtimestep)
    depth = mld(time)
    HypDepth = ld(time) - depth
    convunits = 1000.0*molweight
    numpts = splinepnts
    ipnts = 2
    if (plotpnts .le. 10000) plotpnts = plotpnts + 1
    x(1) = lasttime
    x(2) = time
    y(1,1) = HypConc_Last
    y(2,1) = lastplotconc
    y(3,1) = csat_last
    y(3,2) = scalefactor*convunits*csat(time)
    y(1,2) = conc(4)*convunits*scalefactor
!   note: scalefactor changes units to ug/L
    y(2,2) = conc(3)*convunits*scalefactor
    tout1 = time
    do i = 1,3
!     put data in save array in case rescaling of plot is required
      data_save(2,plotpnts,i) = y(i,2)
    end do
    if ((maxconc .lt. y(1,2)).or.&
        (maxconc .lt. y(2,2)).or.&
        (maxconc .lt. y(3,2))) then
      maxtemp = 0.0
      do while ((maxtemp .lt. y(1,2)).or.&
                (maxtemp .lt. y(2,2)).or.&
                (maxtemp .lt. y(3,2)))
        maxconc = 1.5*max(y(1,2),y(2,2),y(3,2),maxtemp)
        maxtemp = maxconc
      end do
      if (maxconc .ge. 10.0) then
        scalefactor = 0.1*scalefactor
        maxconc = 0.1*maxconc
        do i = 1, ipnts
          do j = 1, isets
            y(j,i) = 0.1 * y(j,i)
          end do
        end do
        do i = 1, isets
          do j = 1, plotpnts
            data_save(2,j,i) = 0.1 * data_save(2,j,i)
          enddo
        end do
      endif
      plotInit = .false.
    endif
    tout2 = time
    if (time .gt. 0.0) then
      call xyplot(x, y, ipnts)
      itotalsets = itotalsets - 1
    endif
    tout3 = time
   if (DatOut) then
      yrtime = 12.0*(time/365.0 - float(int(time/365.0)))
      yr = time/365.0
      u = interpolate(spl_WindSpeed, yrtime, splinepnts)
      degc = interpolate(spl_SurfaceTemp, yrtime, splinepnts)
      airconc = interpolate(spl_AtmMTBEConc, yrtime, splinepnts)
!     airconc in ppbv
      solubility = sol_calc(degc,SolParam)
!     solubility in mol/m^3-atm
      la = la_func(time)
      evap = la*flux_h2o(time)
      write (datfilunit,'(f8.2,4e14.4,2f8.2,f9.4,f6.1,3e11.3,e12.3,e14.4,i5,&
            8e14.4)',iostat=ierr) time, conc(3)*convunits, conc(4)*convunits,&
            conc(1), conc(2), u, degc, kl(time), mld(time),&
            mldp(time),ii(time),convunits*csat(time),airconc,evap,case_flow,&
            InH, OutH, in, out, iexchange, makeup, mu_mass, la
    endif
    HypConc_Last = y(1,2)
    csat_last = y(3,2)
    lastplotconc = y(2,2)
    lasttime = time
!   update the time for the next graph point and output point
    time = time + OutputTimeStep
!   PAUSE loop for modifying VOC inputs..
    do while (pause_mod)
      continue
    end do
!    Check lrunning to seeif we want to halt the model run
    if (.not. lrunning) then
      iret = messageboxqq('Run stopped by user'C, 'Model Status'C, mb$ok)
      menuactive = .false.
      if (DatOut) then
        close (DatFilUnit)
        DatOut = .false.
!        error here in character output 3/2/99 wea
```

```
        write (msg0, '(a,a)') 'Closed data output file'C
        msg1 = ' FILE I/O STATUS UPDATE 'C
        iret = messageboxqq(msg0, msg1, mb$iconexclamation .or. mb$ok)
      endif
!      This is the call that actually terminates the thread
      call exitthread(0)    !exit code is 0
    endif
  return
end subroutine gout


subroutine TimeSeriesEntry(dlg_parent, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  character*72 Local_title, local_units
  type(dialog) dlg_parent
  type (dialog) ts_dlg
  logical(kind=4) err
  integer(kind=4) id, callbacktype, iret, ierr, iloop
  external TSEntry_OK, TSFileEntry_OK
  logical(kind=4)checked
  call unusedqq(checked)
  ierr = callbacktype
  msg0 = ''c
  msg1 = ''c
  dlg_save = dlg_parent
! ts_entry_id defined in MTBECOM:  used to check the values in the time series
! and to reset the parent dialog box.
  ts_entry_id = id
! first need to close the parent dialog and save the temporary data
  call shutdown_parent (dlg_parent, id)
! main loop is used to check errors in input strings
  err = .true.
  errorwindow = .false.
  iloop = 1
  do while (err)
!   Select the case for the correct set of units and default data for the time
!   series data for each month
    select case (id)
!     note:  CTEMP passed to TimeSeriesEntrySetup through MTBECOM.F90
      case (IDC_SurfaceTemp)
        if (TSSetup(indexTW) .eq. 1) then
          write(ctemp,*) 'deg-C'C
          Local_Title = 'Epilimnion Temperature'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, SurfaceTemp, local_title)
        else
          Local_Title = 'Epilimnion Temperature Time Series File Entry'C
          local_units = 'degrees Celsius'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_MixedLayer)
        if (TSSetup(indexMLD) .eq. 1) then
          write(ctemp,*) 'm'C
          Local_Title = 'Epilimnion Depth'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, MixedLayer, local_title)
        else
          Local_Title = 'Epilimnion Depth Time Series File Entry'
          local_units = 'meters'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_LakeDepth)
        if (TSSetup(indexLD) .eq. 1) then
          write(ctemp,*) 'm'C
          Local_Title = 'Lake Depth'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, LakeDepth, local_title)
        else
          Local_Title = 'Lake Depth Time Series File Entry'
          local_units = 'meters'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_Inflow)
        if (TSSetup(indexIN) .eq. 1) then
          write(ctemp,*) 'm^3/day'C
          Local_Title = 'Lake Inflow Volume'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, Inflow, local_title)
        else
          Local_Title = 'Lake Inflow Volume Time Series File Entry'
          local_units = 'cubic meters per day'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_Outflow)
        if (TSSetup(indexOUT) .eq. 1) then
          write(ctemp,*) 'm^3/day'C
          Local_Title = 'Lake Outflow Volume'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, Outflow, local_title)
        else
          Local_Title = 'Lake Outflow Volume Time Series File Entry'
          local_units = 'cubic meters per day'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_InflowHeight)
        if (TSSetup(indexINHe) .eq. 1) then
          write(ctemp,*) 'm'C
          Local_Title = 'Lake Inflow Height (From Lake Bottom)'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, InflowHeight, local_title)
        else
          Local_Title = 'Lake Inflow Height Time Series File Entry'
          local_units = 'meters'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_OutflowHeight)
        if (TSSetup(indexOUTHe) .eq. 1) then
          write(ctemp,*) 'm'C
          Local_Title = 'Lake Outflow Height (From Lake Bottom)'C
          call TimeSeriesEntrySetup (ts_dlg,iloop,OutflowHeight,local_title)
        else
          Local_Title = 'Lake Outflow Height Time Series File Entry'
          local_units = 'meters'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_WindSpeed)
        if (TSSetup(indexU) .eq. 1) then
          write(ctemp,*) 'm/s'C
          Local_Title = 'Wind Speed'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, WindSpeed, local_title)
```

```
        else
          Local_Title = 'Wind Speed Time Series File Entry'
          local_units = 'meters per second'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_AirTemp)
        if (TSSetup(indexTA) .eq. 1) then
          write(ctemp,*) 'deg-C'C
          Local_Title = 'Air Temperature'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, AirTemp, local_title)
        else
          Local_Title = 'Air Temperature Time Series File Entry'
          local_units = 'degrees Celsius'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_AtmPressure)
        if (TSSetup(indexPA) .eq. 1) then
          write(ctemp,*) 'Atm.'C
          Local_Title = 'Atmospheric Pressure'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, AtmosPress, local_title)
        else
          Local_Title = 'Atm. Pressure Time Series File Entry'
          local_units = 'Atmospheres'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_MTBEInputSeries, IDC_RuntimeMTBEInputSeries)
        if (TSSetup(indexMTBE) .eq. 1) then
          write (ctemp, *) 'kg/month'C
          Local_Title = 'Direct VOC Input to Epilimnion'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, MTBEInput, local_title)
        else
          Local_Title = 'VOC Input Time Series File Entry'
          if (TSSetup(indexMTBE) .eq. 2) local_units = 'kilograms per week'C
          if (TSSetup(indexMTBE) .eq. 3) local_units = 'kilograms per day'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_AtmMTBEConc, IDC_RuntimeAtmMTBEConc)
        if (TSSetup(indexAirMTBE) .eq. 1) then
          write (ctemp, *) 'ppbv'C
          Local_Title = 'Atmospheric VOC Concentration'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, AtmMTBEConc, local_title)
        else
          Local_Title = 'Atm. VOC Conc. Time Series File Entry'
          local_units = 'Part-per-billion by volume'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_EpiLossRate)
        if (TSSetup(indexEpiL) .eq. 1) then
          write(ctemp,*) '1/days'C
          Local_Title = 'Epilimnion Biochemical Degradation Rate'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, EpiLossRate, local_title)
        else
          Local_Title = 'Epilimnion Loss Rate Time Series File Entry'
          local_units = 'inverse days (days^-1)'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
      case (IDC_HypLossRate)
        if (TSSetup(indexHypL) .eq. 1) then
          write(ctemp,*) '1/days'C
          Local_Title = 'Hypolimnion Biochemical Degradation Rate'C
          call TimeSeriesEntrySetup (ts_dlg, iloop, HypLossRate, local_title)
        else
          Local_Title = 'Hypolimnion Loss Rate Time Series File Entry'
          local_units = 'inverse days (days^-1)'C
          call TSFileEntry(ts_dlg, id, Local_Title, local_units)
        endif
    end select
! Set units for time series
!   bring up the dialog box
    iret = dlgmodal(ts_dlg)
!* destroy and release the dialog resources
    call dlguninit(ts_dlg)
    err = .false.
    if ((err_dlg(1)).or.(err_dlg(2)).or.(err_dlg(3)).or.(err_dlg(4)).or.&
        (err_dlg(5)).or.(err_dlg(6)).or.(err_dlg(7)).or.(err_dlg(8)).or.&
        (err_dlg(9)).or.(err_dlg(10)).or.(err_dlg(11)).or.(err_dlg(12)).or.&
        (err_dlg(13))) then
      err = .true.
      call dialog_error_display(id)
!     id is the identifier that tells who called
    endif
    iloop = iloop + 1
  enddo
  if (errorwindow) close (ErrWinUnit)
  call reset_parentdialog (dlg_parent)
  dlg_save = dlg_parent
  return
end subroutine TimeSeriesEntry

subroutine TimeSeriesEntrySetup (dlg, iloop, dat_array, local_title)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog)dlg
  logical lret
  character*72 local_title
  character*25 cmonth(12)
  real*8 dat_array(12)
  integer i, iloop
  external TSEntry_OK
  lret = dlginit(IDD_TimeSeriesEntry, dlg)
  lret = dlgset(dlg, IDC_MonthlyTitle, local_title)
  lret = dlgsetsub(dlg, idok, TSEntry_OK)
  do i = 1, 12
    if (iloop .eq. 1) write (err_str(i),'(f12.2)') dat_array(i)
    cmonth(i) = err_str(i)
!   Set the units in the text box (note: ctemp set in TimeSeriesEntry and
!   passed through MTBECOM.F90
    lret = dlgset(dlg, IDCMonthUnits(i), ctemp)
!   Set default values
    lret = dlgset(dlg, IDCMonthValues(i), cmonth(i))
  end do
  return
end subroutine TimeSeriesEntrySetup


subroutine TSEntry_OK(DLG, ID, CALLBACKTYPE)
```

```
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret, lerr
  integer(kind=4) i, id, callbacktype, ierr(12), index
  real*8 time, depth, mixed, dd_save, dayspermonth
  real*8 ld, mld
  external ld, mld
  dayspermonth = 365.0/12.0
  call unusedqq(dlgparent, id, callbacktype)
  lerr = .false.
!* get the new information
  do i = 1, 12
    ret = dlgget(dlg, IDCMonthValues(i), err_str(i))
  end do
  err_dlg(13) = .false.
  do i = 1, 12
    err_dlg(i) = .false.
    read(err_str(i),*,iostat=ierr(i)) dummy_dat(i)
    if (ierr(i) .eq. 0) then
      select case (ts_entry_id)
!       ts_entry_id is defined in MTBECOM and is used to check the values
!       in the time series and to reset the parent dialog box.  It is used
!       in the three subroutines below.
        case (IDC_SurfaceTemp)
          index = indexTW
        case (IDC_MixedLayer)
          index = indexMLD
          if (dummy_dat(i) .ge. 0.0) then
            time = float(i)*dayspermonth - 0.5*dayspermonth
            depth = ld(time)
            if (dummy_dat(i) .gt. depth) err_dlg(i) = .true.
!           test to see if MLD > LD
          else
            err_dlg(i) = .true.
          endif
        case (IDC_LakeDepth)
          index = indexLD
          if ((dummy_dat(i).gt.0.0).and.(dummy_dat(i).le.LakeArea(1,1))) then
            index = indexLD
          else
            err_dlg(i) = .true.
          endif
        case (IDC_Inflow)
          index = indexIN
          if (dummy_dat(i) .ge. 0.0) then
            index = indexIN
          else
            err_dlg(i) = .true.
          endif
        case (IDC_Outflow)
          index = indexOUT
          if (dummy_dat(i) .ge. 0.0) then
            index = indexOUT
          else
            err_dlg(i) = .true.
          endif
        case (IDC_InflowHeight)
          index = indexIN
          if (dummy_dat(i) .ge. 0.0) then
            index = indexINHe
          else
            err_dlg(i) = .true.
          endif
        case (IDC_OutflowHeight)
          index = indexOUTHe
          if (dummy_dat(i) .ge. 0.0) then
            index = indexOUTHe
          else
            err_dlg(i) = .true.
          endif
        case (IDC_AirTemp)
          index = indexTA
        case (IDC_WindSpeed)
          index = indexU
          if (dummy_dat(i) .ge. 0.0) then
            index = indexU
          else
            err_dlg(i) = .true.
          endif
        case (IDC_AtmPressure)
          index = indexPA
          if (dummy_dat(i) .gt. 0.0) then
            index = indexPA
          else
            err_dlg(i) = .true.
          endif
        case (IDC_MTBEInputSeries, IDC_RuntimeMTBEInputSeries)
          index = indexMTBE
          if (dummy_dat(i) .ge. 0.0) then
            index = indexMTBE
          else
            err_dlg(i) = .true.
          endif
        case (IDC_AtmMTBEConc, IDC_RuntimeAtmMTBEConc)
          index = indexAirMTBE
          if (dummy_dat(i) .ge. 0.0) then
            index = indexAirMTBE
          else
            err_dlg(i) = .true.
          endif
        case (IDC_EpiLossRate)
          index = indexEpiL
          if (dummy_dat(i) .ge. 0.0) then
            index = indexEpiL
          else
            err_dlg(i) = .true.
          endif
        case (IDC_HypLossRate)
          index = indexHypL
          if (dummy_dat(i) .ge. 0.0) then
            index = indexHypL
          else
            err_dlg(i) = .true.
          endif
```

```
      end select
    else
!  else for the (if (ierr(i).eq.0) statement (format error on input variable)
      err_dlg(i) = .true.
    endif   !  endif for the (if (ierr(i) .eq. 0) statement
  end do
!  write (*, '(12f6.0)') dummy_dat
! spline the new data and test it if there are no dialog errors
  if ((.not. err_dlg(1)).and.(.not. err_dlg(2)).and.(.not. err_dlg(3)).and.&
      (.not. err_dlg(4)).and.(.not. err_dlg(5)).and.(.not. err_dlg(6)).and.&
      (.not. err_dlg(7)).and.(.not. err_dlg(8)).and.(.not. err_dlg(9)).and.&
      (.not. err_dlg(10)).and.(.not. err_dlg(11)).and.(.not. err_dlg(12))) &
  then
    call spline_dummy(12)  ! spline_dummy located in par_tser.f90
    call data_test(12, ts_entry_id, lerr)
!   if LERR .eq. TRUE then data is OK
    if (lerr) then
      TSDatLen(index) = 12
      call spline_data(index)
      if (index .eq. indexLD) call spline_data(indexMLD)
!     reset MLD to new Lake Depth series
!     TSError is the array telling whether length of series is correct
!     set in TS_SETUP.F90, this resets it so that data is OK in the dialog
      if (TSError(index)) TSError(index) = .false.
    else
      err_dlg(13) = .true.
    endif
  endif
  call dlgsetreturn(dlg, idok)
  call dlgexit(dlg)
  return
end subroutine TSEntry_OK


subroutine shutdown_parent (dlg, id)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  logical ret
  integer id, iret
  type(dialog) dlg
  iret = id
  ! ts_entry_id defined in MTBECOM: used to check the values in the time series
  ! and to reset the parent dialog box.
    select case (id)
      case (IDC_MixedLayer, IDC_SurfaceTemp, IDC_LakeDepth, IDC_Inflow,  &
            IDC_Outflow, IDC_InflowHeight, IDC_OutflowHeight, IDC_LakeArea, &
            IDC_LakeArea2)
        ret = dlgget(dlg, IDC_ProfilePoints, ctemp)
        temp_dlg(1) = ctemp
      case (IDC_WindSpeed, IDC_AirTemp, IDC_AtmPressure)
        ret = dlgget(dlg, IDC_RelativeHumidity, ctemp)
        temp_dlg(1) = ctemp
      case (IDC_MTBEInputSeries, IDC_AtmMTBEConc, IDC_CallDiffParam, &
            IDC_CallSolParam, IDC_EpiLossRate,IDC_HypLossRate)
        ret = dlgget(dlg, IDC_MolWeight, ctemp)
        temp_dlg(1) = ctemp
        ret = dlgget(dlg, IDC_InitialConc, ctemp)
        temp_dlg(2) = ctemp
      case (IDC_RuntimeMTBEInputSeries, IDC_RuntimeAtmMTBEConc)
        continue
    end select
    call dlgexit(dlg)
    call dlguninit(dlg)
  return
end subroutine shutdown_parent


subroutine reset_parentdialog (dlg)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  logical ret
  integer iret
  type(dialog) dlg
  external timeseriesentry, mtbeparam_ok, meteor_ok, hydrog_ok, &
           mtbeparam_cancel, meteor_cancel, DiffParamEntry,&
           SolParamEntry, runtimeMTBEParam_OK, enterDepthProfile, &
           ViewDepthProfile
  ctemp=temp_dlg(1)
  ! ts_entry_id defined in MTBECOM:  used to check the values in the time series
  ! and to reset the parent dialog box.
  ! it must be set in the calling program
    select case (ts_entry_id)
      case (IDC_MixedLayer,IDC_SurfaceTemp, IDC_LakeDepth, IDC_Inflow, &
            IDC_Outflow, IDC_InflowHeight, IDC_OutflowHeight, &
            IDC_LakeArea, IDC_LakeArea2)
        ret = dlginit(IDD_HydrogParams, dlg)
        ret = dlgsetsub(dlg, IDOK, Hydrog_OK)
        ret = dlgsetsub(dlg, IDC_MixedLayer, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_SurfaceTemp, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_LakeDepth, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_Inflow, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_Outflow, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_InflowHeight, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_OutflowHeight, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_LakeArea, EnterDepthProfile)
        ret = dlgsetsub(dlg, IDC_LakeArea2, ViewDepthProfile)
        ret = dlgset(dlg, IDC_ProfilePoints, temp_dlg(1))
        call Set_buttons(startHydro, endHydro, dlg)
        iret = dlgmodal(dlg)
      case (IDC_WindSpeed, IDC_AirTemp, IDC_AtmPressure)
        ret = dlginit(IDD_MeteorParams, dlg)
        ret = dlgsetsub(dlg, IDOK, Meteor_OK)
        ret = dlgsetsub(dlg, IDCANCEL, Meteor_Cancel)
        ret = dlgsetsub(dlg, IDC_AirTemp, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_WindSpeed, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_AtmPressure, TimeSeriesEntry)
        ret = dlgset(dlg, IDC_RelativeHumidity, temp_dlg(1))
        call set_buttons(startAtm, endAtm, dlg)
        iret = dlgmodal(dlg)
      case (IDC_MTBEInputSeries, IDC_AtmMTBEConc, IDC_CallDiffParam, &
            IDC_CallSolParam, IDC_EpiLossRate, IDC_HypLossRate)
        ret = dlginit(IDD_MTBEParams, dlg)
        ret = dlgsetsub(dlg, IDOK, MTBEParam_OK)
        ret = dlgsetsub(dlg, IDCANCEL, MTBEParam_Cancel)
        ret = dlgsetsub(dlg, IDC_MTBEInputSeries, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_AtmMTBEConc, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_CallDiffParam, DiffParamEntry)
        ret = dlgsetsub(dlg, IDC_CallSolParam, SolParamEntry)
        ret = dlgsetsub(dlg, IDC_EpiLossRate, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_HypLossRate, TimeSeriesEntry)
        ret = dlgset(dlg, IDC_MolWeight, temp_dlg(1))
        ret = dlgset(dlg, IDC_InitialConc, temp_dlg(2))
        call set_buttons(startVOC, endVOC, dlg)
        iret = dlgmodal(dlg)
      case (IDC_RuntimeMTBEInputSeries, IDC_RuntimeAtmMTBEConc)
        ret = dlginit(IDD_RuntimeMTBEParams,dlg)
        ret = dlgsetsub(dlg, IDOK, RuntimeMTBEParam_OK)
        ret = dlgsetsub(dlg, IDC_RuntimeMTBEInputSeries, TimeSeriesEntry)
        ret = dlgsetsub(dlg, IDC_RuntimeAtmMTBEConc, TimeSeriesEntry)
        call set_buttons(startVOC, endVOC, dlg)
        iret = dlgmodal(dlg)
    end select
  return
end subroutine reset_parentdialog


subroutine TSFileEntry (dlg, id, Local_Title, local_units)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  use graphcom
  implicit none
  character*72 Local_Title, local_units
  character*25 Local_FileName
  character*65 Local_FileStatus
  character*65 Local_FileStatus2
  character($MAXPATH) tempdir, tDataDir
  logical ret
  integer id, length, ts_ident
  type(dialog) dlg
  external TSFileEntry_OK, GraphData
    call unusedqq(dlg, id, length)
    graph_id = id
    tempdir = FILE$CURDRIVE
    length = getdrivedirqq(tempdir)
  ! ts_entry_id defined in TimeSeriesEntry which is the main calling routine
  ! ts_entry_id is located in MTBECOM.F90
    call select_ts_id(ts_entry_id, ts_ident)
    if (TSError(ts_ident)) FileLoaded(ts_ident) = .false.
    if (DataDir(ts_ident) .eq. '') then
      tDataDir = tempdir
    else
      tDataDir = DataDir(ts_ident)
    endif
    local_filename = FileName(ts_ident)
    local_filestatus = File_status(ts_ident)
    local_filestatus2 = File_status2(ts_ident)
    ret = dlginit(IDD_TimeSeriesFileEntry, dlg)
    ret = dlgset(dlg, IDC_LocalTitle, Local_Title)
    ret = dlgset(dlg, IDC_DataUnits, local_units)
    ret = dlgset(dlg, IDC_CurrentWorkDir, tempdir)
    ret = dlgset(dlg, IDC_DataDirectory, tDataDir)
    ret = dlgset(dlg, IDC_TimeSeriesFilename, local_filename)
    ret = dlgset(dlg, IDC_FileStatus, local_FileStatus)
    ret = dlgset(dlg, IDC_FileStatus2, local_filestatus2)
    ret = dlgsetsub(dlg, IDC_GraphData, GraphData)
    ret = dlgsetsub(dlg, IDC_LoadData, TSFileEntry_OK)
    ret = dlgsetsub(dlg, IDOK, TSFileEntry_OK)
    return
end subroutine TSFileEntry


subroutine TSFileEntry_OK(DLG, ID, CALLBACKTYPE)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  use graphcom
  implicit none
  logical ret, file_exist, data_ok
  integer i, id, callbacktype, lengthdir, lengthfile, ts_ident, file_iostat, idum,&
          ipnts
  type(dialog) dlg
  character($MAXPATH) tdir, dir_file, dir_save
  character*25 tfile
  character*235 local_filestatus
    call unusedqq(dlg, id, callbacktype)
    ret = dlgget(dlg, IDC_DataDirectory, tdir)
    if (tdir .eq. '') then
      local_filestatus = 'Enter Directory!'
      ret = dlgset(dlg, IDC_FileStatus, local_filestatus)
      return
    endif
    ret = dlgget(dlg, IDC_TimeSeriesFilename, tfile)
    if (tfile .eq. '') then
      local_filestatus = 'Enter Filename!'
      ret = dlgset(dlg, IDC_FileStatus, local_filestatus)
      return
    endif
    dir_save = FILE$CURDRIVE
    lengthdir = getdrivedirqq(dir_save)
    if (.not. changedirqq(tdir)) then
      local_filestatus = 'Directory Does Not Exist!'
      ret = dlgset(dlg, IDC_FileStatus, local_filestatus)
      return
    endif
    ret = changedirqq(dir_save)
    lengthdir = len_trim(tdir)
    lengthfile = len_trim(tfile)
    if ((tdir(lengthdir:lengthdir) .ne. '\') .and. (tfile(1:1) .ne. '\')) then
      tdir(lengthdir+1:lengthdir+1) = '\'
      lengthdir = lengthdir + 1
    endif
  ! if both have \'s in them, reset lengthdir so trailing slash is overwritten
    if ((tdir(lengthdir:lengthdir) .eq. '\') .and. (tfile(1:1) .eq. '\')) &
         lengthdir = lengthdir - 1
    dir_file = tdir(1:lengthdir)//tfile(1:lengthfile)
  ! ts_entry_id defined in TimeSeriesEntry
  ! ts_entry_id is located in MTBECOM.F90
    call select_ts_id(ts_entry_id, ts_ident)
  ! now check to see if file exists
    inquire (FILE=dir_file, EXIST=file_exist)
    if (.not. file_exist) then
      local_filestatus = 'File Not Found!'
      ret = dlgset(dlg, IDC_FileStatus, local_filestatus)
      file_status(ts_ident) = local_filestatus
      return
    endif
  ! file was found, say so in dialog box and set status parameter
    File_Status(ts_ident) = 'File Found'
    ret = dlgset(dlg, IDC_FileStatus, File_Status(ts_ident))
  ! open up the data file and read the first line of header information
    open (11, file=dir_file, iostat=file_iostat)
    file_iostat = 0
    read (11, *, iostat=file_iostat)
    i = 0
    do while (file_iostat .ge. 0)
      i = i + 1
      read (11, *, iostat=file_iostat) idum, dummy_dat(i)
    end do
    close (11)
    i = i - 1
    select case (TSSetup(ts_ident))
      case (2)
        if (i .ne. 52) then
          if (i .lt. 52) local_filestatus = &
             'Error Reading File!  Not enough data points.'
          if (i .gt. 52) local_filestatus = &
             'Error Reading File!  Too many data points.'
          ret = dlgset(dlg, IDC_FileStatus2, local_filestatus)
          return
        endif
        ipnts = i
        data_ok = .false.
        call spline_dummy(52)  ! spline_dummy located in par_tser.f90
        call data_test(ipnts, ts_entry_id, data_ok)
      case (3)
        if (i .ne. 365) then
          if (i .lt. 365) local_filestatus = &
              'Error Reading File!  Not enough data points.'
          if (i .gt. 365) local_filestatus = &
              'Error Reading File!  Too many data points.'
          ret = dlgset(dlg, IDC_FileStatus2, local_filestatus)
          return
        endif
        ipnts = i
        data_ok = .false.
        call spline_dummy(365)  ! spline_dummy located in par_tser.f90
        call data_test(ipnts, ts_entry_id, data_ok)
    end select
    if (data_ok) then
      FileLoaded = .true.
      write(local_filestatus, '(a,a25)') 'Data Loaded from File: ', tfile
      ret = dlgset(dlg, IDC_FileStatus2, local_filestatus)
      file_status2(ts_ident) = local_filestatus
      TSDatLen(ts_ident) = ipnts
      call spline_data(ts_ident)
      if (TSError(ts_ident)) TSError(ts_ident) = .false.
      DataDir(ts_ident) = tdir
      FileName(ts_ident) = tfile
      gfile = tfile
  !   next line resets MLD_data to the data in MixedLayer if LakeDepth changes
      if (ts_ident .eq. IndexLD) call spline_data(indexMLD)
      call lake_volume_calc
    else
      select case (ts_ident)
        case (indexLD)
          write(local_filestatus, '(a,a25,a,a,a)') &
            'Data Input Error, Check File: ', tfile, &
            '\nPossible errors are LD<MLD, LD<0, non-numeric data\n', &
            'It is possible you need to reset MLD before LD'C
        case (indexMLD)
          write(local_filestatus, '(a,a25,a)') &
          'Data Input Error, Check File: ', tfile, &
          '\nPossible errors are MLD<0, LD<MLD, non-numeric data'C
        case (indexTW)
          write(local_filestatus, '(a,a25,a)') &
          'Data Input Error, Check File: ', tfile, &
            '\nPossible errors are non-numeric data'C
        case (indexIN)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are Inflow<0, non-numeric data'C
        case (indexOUT)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are Outflow<0, non-numeric data'C
        case (indexINHe)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are Inflow Height<0, non-numeric data'C
        case (indexOUTHe)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are Outflow Height<0, non-numeric data'C
        case (indexPA)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are Atm. Pressure<0, non-numeric data'C
        case (indexTA)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are non-numeric data'C
        case (indexU)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are wind speed<0, non-numeric data'C
        case (indexMTBE)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are VOC Input<0, non-numeric data'C
        case (indexAirMTBE)
          write(local_filestatus,'(a,a25,a)') 'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are [VOC]-atmos. <0, non-numeric data'C
        case (indexEpiL)
          write(local_filestatus,'(a,a25,2a)')'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are epilimnion degradation rate<0',&
          'non-numeric data'C
        case (indexHypL)
          write(local_filestatus,'(a,a25,2a)')'Data Input Error, Check File: ',&
          tfile, '\nPossible errors are hypolimnion degradation rate<0\n',&
          'non-numeric data'C
      end select
      ret = dlgset(dlg, IDC_FileStatus2, local_filestatus)
      file_status2(ts_ident) = local_filestatus
      return
    endif
```

```
    if (id .eq. idok) then
      call dlgsetreturn(dlg, idok)
      call dlgexit(dlg)
    endif
    return
end subroutine TSFileEntry_OK

subroutine data_test (ipnts, id, data_ok)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  logical data_ok
  integer ipnts, id, i, iret, LDpnts
  real*8 ld_save(splinepnts), dd_save(splinepnts)
  data_ok = .true.
    select case (id)
      case (IDC_SurfaceTemp)
        if (allocated(SurfaceTemp)) deallocate(SurfaceTemp)
        allocate(SurfaceTemp(ipnts))
        do i = 1, ipnts
          SurfaceTemp(i) = dummy_dat(i)
        end do
        data_ok = .true.
      case (IDC_MixedLayer)
        do i = 1, 365
          if ((spl_dummy(i) .lt. 0.0) .or. &
          (spl_dummy(i) .gt. spl_LakeDepth(i))) then
            data_ok = .false.
           endif
        end do
        if (data_ok) then
          if (allocated(MixedLayer)) deallocate(MixedLayer)
          allocate(MixedLayer(ipnts))
          do i = 1, ipnts
            MixedLayer(i) = dummy_dat(i)
          end do
        endif
      case (IDC_LakeDepth)
   !    first we need to put test LakeDepth spine in spl_LakeDepth
   !    so we can make spl_dummy = raw MLD_data to test lake depth profile
        do i = 1, 365
          LD_save(i) = spl_dummy(i)
  !       saved spl_dummy (new spl_LakeDepth) in LD_save
        end do
        do i = 1, ipnts
          dd_save(i) = dummy_dat(i)
  !       saved dummy_dat (new lakeDepth series) in dd_save
        end do
        do i = 1, TSDatLen(indexMLD)
          dummy_dat(i) = MixedLayer(i)  ! copy MixedLayer into dummy_dat
        end do
        call spline_dummy(ipnts)
  !     now spl_dummy contains the raw splined MLD time series
  !     now test LD_save (trial LD time series) against raw current
  !     MLD time series in spl_dummy
        do i = 1, 365
          if (LD_save(i) .le. 0.0) data_ok = .false.
          if (spl_dummy(i) .gt. LD_save(i)) data_ok = .false.
          if (LD_save(i) .gt. LakeArea(1,1)) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(LakeDepth)) deallocate(LakeDepth)
          allocate(LakeDepth(ipnts))
          do i = 1, ipnts
            LakeDepth(i) = dd_save(i)
  !         LakeDepth time series is in DD_save, not dummy_dat
          end do
        endif
      case (IDC_Inflow)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(Inflow)) deallocate(Inflow)
          allocate(Inflow(ipnts))
          do i = 1, ipnts
            Inflow(i) = dummy_dat(i)
          end do
        endif
      case (IDC_Outflow)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(Outflow)) deallocate(Outflow)
          allocate(Outflow(ipnts))
          do i = 1, ipnts
            Outflow(i) = dummy_dat(i)
          end do
        endif
      case (IDC_InflowHeight)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(InflowHeight)) deallocate(InflowHeight)
          allocate(InflowHeight(ipnts))
          do i = 1, ipnts
            InflowHeight(i) = dummy_dat(i)
          end do
        endif
      case (IDC_OutflowHeight)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(OutflowHeight)) deallocate(OutflowHeight)
          allocate(OutflowHeight(ipnts))
          do i = 1, ipnts
            OutflowHeight(i) = dummy_dat(i)
          end do
        endif
      case (IDC_AirTemp)
        if (allocated(AirTemp)) deallocate(AirTemp)
        allocate(AirTemp(ipnts))
        do i = 1, ipnts
          AirTemp(i) = dummy_dat(i)
        end do
        data_ok = .true.
      case (IDC_WindSpeed)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(WindSpeed)) deallocate(WindSpeed)
          allocate(WindSpeed(ipnts))
          do i = 1, ipnts
            WindSpeed(i) = dummy_dat(i)
          end do
        endif
      case (IDC_AtmPressure)
        do i = 1, 365
          if (spl_dummy(i) .le. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(AtmosPress)) deallocate(AtmosPress)
          allocate(AtmosPress(ipnts))
          do i = 1, ipnts
            AtmosPress(i) = dummy_dat(i)
          end do
        endif
      case (IDC_MTBEInputSeries, IDC_RuntimeMTBEInputSeries)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(MTBEInput)) deallocate(MTBEInput)
          allocate(MTBEInput(ipnts))
          do i = 1, ipnts
            MTBEInput(i) = dummy_dat(i)
          end do
        endif
      case (IDC_AtmMTBEConc, IDC_RuntimeAtmMTBEConc)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(AtmMTBEConc)) deallocate(AtmMTBEConc)
          allocate(AtmMTBEConc(ipnts))
          do i = 1, ipnts
            AtmMTBEConc(i) = dummy_dat(i)
          end do
        endif
      case (IDC_EpiLossRate)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(EpiLossRate)) deallocate(EpiLossRate)
          allocate(EpiLossRate(ipnts))
          do i = 1, ipnts
            EpiLossRate(i) = dummy_dat(i)
          end do
        endif
      case (IDC_HypLossRate)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
        if (data_ok) then
          if (allocated(HypLossRate)) deallocate(HypLossRate)
          allocate(HypLossRate(ipnts))
          do i = 1, ipnts
            HypLossRate(i) = dummy_dat(i)
          end do
        endif
    end select
  return
end subroutine data_test

subroutine spline_dummy (ipnts)
  use tser_com
  implicit none
  integer ipnts
  logical test
  integer i, j, numpts
  parameter (numpts=3)
  real*8 time, ti, tf, timefrac, monthconv, time_month, time_week, weekconv
  !begin subroutine
    monthconv = 12.0/365.0
    weekconv = 52.0/365.0
    timefrac = 1.0
    if (ipnts .eq. 365) then
      do i = 1, 365
        spl_dummy(i) = dummy_dat(i)
      end do
  ! begin section for weekly data
    elseif (ipnts .eq. 52) then
      do j = 1, 365
        time = float(j)*timefrac
        test = .false.
        i = 1
        do while ((i .le. 51) .and. (.not. test))
          ti = float(i) - 0.5
          tf = ti + 1.0
          time_week = time*weekconv
          if ((time_week .ge. ti) .and. (time_week .lt. tf)) then
            spl_dummy(j) = &
              dummy_dat(i) + (time_week-ti)*(dummy_dat(i+1)-dummy_dat(i))
            test = .true.
          endif
          i = i + 1
        end do
        if (.not. test) then
          if (time_week .ge. 51.5) then
            spl_dummy(j) = &
              dummy_dat(52) + (time_week-51.5)*(dummy_dat(1)-dummy_dat(52))
            test = .true.
          elseif (time_week .lt. 0.5) then
            spl_dummy(j) = &
              dummy_dat(52) + (time_week+0.5)*(dummy_dat(1)-dummy_dat(52))
            test = .true.
          endif
        endif
      enddo
  ! begin section for monthly data
    elseif (ipnts .eq. 12) then
      do j = 1, 365
```

```
        time = float(j)*timefrac
        test = .false.
        i = 1
        do while ((i .le. 11) .and. (.not. test))
          ti = float(i) - 0.5
          tf = ti + 1.0
          time_month = time*monthconv
          if ((time_month .ge. ti) .and. (time_month .lt. tf)) then
            spl_dummy(j) = &
              dummy_dat(i) + (time_month-ti)*(dummy_dat(i+1)-dummy_dat(i))
            test = .true.
          endif
          i = i + 1
        end do
        if (.not. test) then
          if (time_month .ge. 11.5) then
            spl_dummy(j) = &
              dummy_dat(12) + (time_month-11.5)*(dummy_dat(1)-dummy_dat(12))
            test = .true.
          elseif (time_month .lt. 0.5) then
            spl_dummy(j) = &
              dummy_dat(12) + (time_month+0.5)*(dummy_dat(1)-dummy_dat(12))
           test = .true.
          endif
        endif
      enddo
    endif
  !  write (*, '(12f6.0)') spl_dummy
    return
end subroutine spline_dummy


subroutine select_ts_id (id, ts_ident)
  use mtbecom
  use Graphcom
  implicit none
  include 'resource.fd'
  integer id, ts_ident
    select case (id)
      case (IDC_SurfaceTemp)
        ts_ident = indexTW
        GraphSeries = spl_SurfaceTemp
      case (IDC_MixedLayer)
        ts_ident = indexMLD
        GraphSeries = MLD_Data
      case (IDC_LakeDepth)
        ts_ident = indexLD
        GraphSeries = spl_LakeDepth
      case (IDC_Inflow)
        ts_ident = indexIN
        GraphSeries = spl_Inflow
      case (IDC_Outflow)
        ts_ident = indexOUT
        GraphSeries = spl_Outflow
      case (IDC_InflowHeight)
        ts_ident = indexINHe
        GraphSeries = spl_InflowHeight
      case (IDC_OutflowHeight)
        ts_ident = indexOUTHe
        GraphSeries = spl_OutflowHeight
      case (IDC_AirTemp)
        ts_ident = indexTA
        GraphSeries = spl_AirTemp
      case (IDC_WindSpeed)
        ts_ident = indexU
        GraphSeries = spl_WindSpeed
      case (IDC_AtmPressure)
        ts_ident = indexPA
        GraphSeries = spl_AtmosPress
      case (IDC_MTBEInputSeries, IDC_RuntimeMTBEInputSeries)
        ts_ident = indexMTBE
        GraphSeries = spl_MTBEInput
      case (IDC_AtmMTBEConc, IDC_RuntimeAtmMTBEConc)
        ts_ident = indexAirMTBE
        GraphSeries = spl_AtmMTBEConc
      case (IDC_EpiLossRate)
        ts_ident = indexEpiL
        GraphSeries = spl_EpiLossRate
      case (IDC_HypLossRate)
        ts_ident = indexHypL
        GraphSeries = spl_HypLossRate
    end select
    return
end subroutine select_ts_id


subroutine set_buttons (j, k, dlg)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  implicit none
  integer i, j, k
  type (dialog) dlg
  logical ret
    do i = j, k
      select case (TSDatLen(i))
        case (12)
          ret = dlgset(dlg, TSOK3(i), 'Monthly')
          if (TSSetup(i) .eq. 1) then
            ret = dlgset(dlg, TSID_Needed(i), 'Monthly')
            ret = dlgset(dlg, TSID_ModReq(i), 'NO')
          else
            ret = dlgset(dlg, TSID_ModReq(i), 'YES')
            select case (TSSetup(i))
              case (2)
                ret = dlgset(dlg, TSID_Needed(i), 'Weekly')
              case (3)
                ret = dlgset(dlg, TSID_Needed(i), 'Daily')
            end select
          endif
        case (52)
          ret = dlgset(dlg, TSOK3(i), 'Weekly')
          if (TSSetup(i) .eq. 2) then
            ret = dlgset(dlg, TSID_Needed(i), 'Weekly')
            ret = dlgset(dlg, TSID_ModReq(i), 'NO')
          else
            ret = dlgset(dlg, TSID_ModReq(i), 'YES')
            select case (TSSetup(i))
              case (1)
```

```
                ret = dlgset(dlg, TSID_Needed(i), 'Monthly')
              case (3)
                ret = dlgset(dlg, TSID_Needed(i), 'Daily')
            end select
          endif
        case (365)
          ret = dlgset(dlg, TSOK3(i), 'Daily')
          if (TSSetup(i) .eq. 3) then
            ret = dlgset(dlg, TSID_Needed(i), 'Daily')
            ret = dlgset(dlg, TSID_ModReq(i), 'NO')
          else
            ret = dlgset(dlg, TSID_ModReq(i), 'YES')
            select case (TSSetup(i))
              case (2)
                ret = dlgset(dlg, TSID_Needed(i), 'Weekly')
              case (1)
                ret = dlgset(dlg, TSID_Needed(i), 'Monthly')
            end select
          endif
      end select
    end do
  return
end subroutine set_buttons


subroutine GraphData (DLG, ID, CALLBACKTYPE)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  use graphcom
  implicit none
  character*5 xyDataLegend
  logical ret, file_exist, data_ok
  integer i, id, callbacktype, lengthdir, lengthfile, ts_ident, file_iostat, &
          idum, ipnts, retcode
  real*8 maxts, mints, maxminfunc, dummyxy(2,splinepnts)
  external maxminfunc
  type(dialog) dlg
  call select_ts_id(graph_id, ts_ident)
  if (.not. FileLoaded(ts_ident)) then
    write(gstatus, '(a,a25)') 'Graph Failed: Load data from file: ', gfile
    ret = dlgset(dlg, IDC_FileStatus2, gstatus)
    file_status2(ts_ident) = gstatus
    return
  endif
  xyDataLegend = '11'
  ipnts = splinepnts
  do i = 1, ipnts
    dummyxy(1,i) = i
    dummyxy(2,i) = GraphSeries(i)
  end do
  call clearscreen($GCLEARSCREEN)
  write (*,'(a)') ' Time Series Data Display'
  openwindow = .true.
    if (Openwindow) then
      if( .not. GetWindowConfig(wc) ) stop 'Window Not Open'
      OpenWindow = .false.
    endif
    retcode=GetGraphDefaults($GTXY,xyGraph)
    xyGraph.setGraphMode=.FALSE.
    xyGraph.graphbgcolor = $CIBLACK
    xyGraph.x1 = 20
    xyGraph.y1 = 30
    xyGraph.x2 = 620
    xyGraph.y2 = 430
    xyGraph.title=graphtitle(ts_ident)
    retcode = GetDataDefaults (xyGraph, ipnts, GraphSeries, xyTimeSeries)
      xyTimeSeries.title=xyDataLegend
      xyTimeSeries.markertype = $MKNONE
      xyTimeSeries.numPoints = splinepnts
      xyTimeSeries.TitleFont = xyGraph.TitleFont
      DataSetColor(1) = xyDataSets(1).linecolor
    retcode = GetAxisDefaults(xyGraph, xyTimeSeries, $ATX, $AFLINEAR, xyAxes(1))
      xyAxes(1).title = 'Days'
      xyAxes(1).lowVal = 0.0
      xyAxes(1).highVal = 365.0
      xyAxes(1).tickColor = 15  !bright white
      xyAxes(1).increment = 40
      xyAxes(1).tickratio = 4
      xyAxes(1).numdigits = 0
      xyAxes(1).gridStyle=$GSNONE
      xyAxes(1).gridLineType=$LTNONE
      xyAxes(1).ticktype = $TTOUTSIDE
      xyAxes(1).axisfont = xyAxes(1).titlefont
    retcode = GetAxisDefaults(xyGraph, xyTimeSeries, $ATY, $AFLINEAR, xyAxes(2))
      mints = maxminfunc(GraphSeries, -1, 365)
      if ((mints .gt. 0.0d0) .and. (mints - 0.2d0*mints .lt. 0.0d0)) then
        mints = 0.0d0
      else
        mints = mints - 0.2d0*mints
      endif
      maxts = maxminfunc(GraphSeries, 1, 365)
      if ((maxts .lt. 0.0d0) .and. (maxts + 0.2d0*maxts .gt. 0.0d0)) then
        maxts = 0.0d0
      else
        maxts = maxts + 0.2d0*maxts
      endif
      xyAxes(2).lowVal = mints
      xyAxes(2).highVal = maxts
      xyAxes(2).increment = 0.1*(xyAxes(2).highVal-xyAxes(2).lowVal)
      xyAxes(2).title=axistitle(ts_ident)
      xyAxes(2).gridStyle=$GSNONE
      xyAxes(2).gridLineType=$LTNONE
      xyAxes(2).ticktype = $TTOUTSIDE
      xyAxes(2).numdigits = 1
      xyAxes(2).tickratio = 1
      xyAxes(2).axisfont = xyAxes(2).titlefont
    retcode=PlotGraph(xyGraph, 2, xyAxes, 1)
    retcode=PlotData(xyGraph, dummyxy, xyTimeSeries, xyAxes(1), xyAxes(2))
    msg0 = 'Press OK to Continue'C
    msg1 = 'Pause'C
    retcode = messageboxqq(msg0, msg1, MB$OK)
    call clearscreen($GCLEARSCREEN)
    write (*, '(1x,a72)') title
    return
  end subroutine GraphData
```

```
subroutine Model_Params (checked)
  use msflib
  use dialogm
  use mtbecom
  use modelcom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  logical(kind=4)lret, err
  integer(kind=4)iret, ierr, iloop
  external ModelPar_OK, ModelPar_Cancel
  logical(kind=4)checked
  call unusedqq(checked)
  ierr = 0
  msg0 = ''c
  msg1 = ''c
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  err = .true.
  if (errorwindow) close (errwinunit)
  errorwindow = .false.
  iloop = 1
  TempTR = TotalRuntime
  TempOT = OutputTimestep
  TempTol = Tolerance
  SaveTR = TotalRuntime
  SaveOT = OutputTimestep
  SaveTol = Tolerance
  do while ((.not. lrunning) .and. (err))
    menuactive = .true.
!   Initialize the dialog box
    lret = dlginit(IDD_RuntimeParams, DLG)
    lret = dlgsetsub(dlg, IDOK, ModelPar_OK)
    if (iloop .eq. 1) then
      write(err_str(1),'(f12.3)') TempTR
      write(err_str(2),'(f12.4)') TempOT
      write(err_str(3),'(e15.4)') TempTol
      err_str(4) = Title
      err_str(5) = Comment(1)
      err_str(6) = Comment(2)
    endif
    lret = dlgset(dlg, IDC_TotalTime, err_str(1))
    lret = dlgset(dlg, IDC_OutputTimestep, err_str(2))
    lret = dlgset(dlg, IDC_Tolerance, err_str(3))
    lret = dlgset(dlg, IDC_Title, err_str(4))
    lret = dlgset(dlg, IDC_Comment1, err_str(5))
    lret = dlgset(dlg, IDC_Comment2, err_str(6))
!   bring up the dialog box
    iret = dlgmodal(dlg)
!   destroy and release the dialog resources
    call dlguninit(dlg)
    menuactive = .false.
    err = .false.
    if (err_dlg(1) .or. err_dlg(2) .or. err_dlg(3) .or. err_dlg(4)) then
      err = .true.
      call dialog_error_display(IDD_RuntimeParams)
    endif
    iloop = iloop + 1
  enddo
  if (lrunning) then
    msg0 = ' Model running: cannot change parameters\nPress <Run\\Stop> to&
             terminate'C
    msg1 = ' PARAMETER SETUP ERROR'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  if (errorwindow) close (ErrWinUnit)
  return
end subroutine Model_Params


subroutine ModelPar_OK (dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use modelcom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret
  integer(kind=4)id, callbacktype, ierr1, ierr2, ierr3
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
  err_dlg(1) = .FALSE.
! get the new information
  ret = dlgget(dlg, IDC_TotalTime, err_str(1))
  read(err_str(1),*,iostat=ierr1) TempTR
  if ((ierr1 .ne. 0) .or. (TempTR .le. 0.0)) then
    err_dlg(1) = .TRUE.
    ierr1 = 0
  else
    TotalRuntime = TempTR
  endif
  err_dlg(2) = .FALSE.
! get the new information
  ret = dlgget(dlg, IDC_OutputTimestep, err_str(2))
  read(err_str(2),*,iostat=ierr1) TempOT
  if ((ierr1 .ne. 0) .or. (TempOT .le. 0.0)) then
    err_dlg(2) = .TRUE.
    ierr1 = 0
  else
    OutputTimestep = TempOT
  endif
  err_dlg(3) = .FALSE.
! get the new information
  ret = dlgget(dlg, IDC_Tolerance, err_str(3))
  read(err_str(3),*,iostat=ierr1) TempTol
  if ((ierr1 .ne. 0) .or. (TempTol .le. 0.0)) then
    err_dlg(3) = .TRUE.
    ierr1 = 0
  else
    Tolerance = TempTol
  endif
  err_dlg(4) = .false.
  if ((TotalRuntime*365.0)/OutputTimestep .gt. 10000.0) err_dlg(4) = .true.
  if (.not. err_dlg(1) .and. .not. err_dlg(2) .and. .not. err_dlg(3) .and.&
      .not. err_dlg(4)) then
    ret = dlgget(dlg, IDC_Title, err_str(4))
    ret = dlgget(dlg, IDC_Comment1, err_str(5))
    ret = dlgget(dlg, IDC_Comment2, err_str(6))
    Title = err_str(4)
    comment(1) = err_str(5)
    comment(2) = err_str(6)
    call punct_b_gone(title)
    call punct_b_gone(comment(1))
    call punct_b_gone(comment(2))
  endif
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine ModelPar_OK


subroutine ModelPar_Cancel (dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use modelcom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret
  integer(kind=4)id, callbacktype, ierr1, ierr2, ierr3
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
  err_dlg(1) = .FALSE.
  TotalRuntime = SaveTR
  err_dlg(2) = .FALSE.
  OutputTimestep = SaveOT
  err_dlg(3) = .FALSE.
  Tolerance = SaveTol
  err_dlg(4) = .false.
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine ModelPar_Cancel


subroutine punct_b_gone(tline)
  implicit none
  character*72 tline
  integer length, i
  length = len_trim(tline)
  do i = 1, length
    if (tline(i:i) .eq. ',') tline(i:i) = ';'
  end do
 return
end subroutine punct_b_gone


subroutine TimeSeries_Setup (checked)
  use msflib
  use dialogm
  use mtbecom
  use tser_com
  use errorcom
  implicit none
! include 'resource.fd'
  type(dialog) dlg
  logical(kind=4) retlog
  integer(kind=4) iret, ierr, iloop, i, j
  external TS_Paramset, TSSetup_OK
  logical(kind=4)checked, err
  call unusedqq(checked)
  ierr = 0
  msg0 = ''c
  msg1 = ''c
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  err = .true.
  if (errorwindow) close (errwinunit)
  errorwindow = .false.
  iloop = 1
  do while ((.not. lrunning) .and. (err))
!   initialize the dialog box
    retlog = dlginit(IDD_TimeSeriesSetup, dlg)
    menuactive = .true.
    retlog = dlgsetsub(dlg, IDOK, TSSetup_OK)
    do i = 1, NumTimeSeries
      select case (TSSetup(i))
        case (1)
          retlog = dlgset(dlg, RadButton(i,1), .true.)
          retlog = dlgset(dlg, RadButton(i,2), .false.)
          retlog = dlgset(dlg, RadButton(i,3), .false.)
          if (TSDatLen(i) .eq. 12) then
            retlog = dlgset(dlg, tsok(i), 'OK')
          else
            retlog = dlgset(dlg, tsok(i), 'ERR')
          endif
        case (2)
          retlog = dlgset(dlg, RadButton(i,1), .false.)
          retlog = dlgset(dlg, RadButton(i,2), .true.)
          retlog = dlgset(dlg, RadButton(i,3), .false.)
          if (TSDatLen(i) .eq. 52) then
            retlog = dlgset(dlg, tsok(i), 'OK')
          else
            retlog = dlgset(dlg, tsok(i), 'ERR')
          endif
        case (3)
          retlog = dlgset(dlg, RadButton(i,1), .false.)
          retlog = dlgset(dlg, RadButton(i,2), .false.)
          retlog = dlgset(dlg, RadButton(i,3), .true.)
          if (TSDatLen(i) .eq. 365) then
            retlog = dlgset(dlg, tsok(i), 'OK')
          else
            retlog = dlgset(dlg, tsok(i), 'ERR')
          endif
      end select
      select case (TSDatlen(i))
        case (12)
          retlog = dlgset(dlg, tsok2(i), 'Monthly')
        case (52)
          retlog = dlgset(dlg, tsok2(i), 'Weekly')
        case (365)
          retlog = dlgset(dlg, tsok2(i), 'Daily')
      end select
      do j = 1, 3
        retlog = dlgsetsub(dlg, RadButton(i,j), TS_ParamSet)
      end do
    end do
!   bring up the dialog box
    iret = dlgmodal(dlg)
!   destroy and release the dialog resources
    call dlguninit(dlg)
    menuactive = .false.
    err = .false.
    if ((err_dlg(1)) .or. (err_dlg(2)))then
      err = .true.
      call dialog_error_display(IDD_TimeSeriesSetup)
    endif
    iloop = iloop + 1
  enddo
  if (lrunning) then
     msg0 = ' Model running: cannot change parameters\nPress <Run\\Stop> &
              to terminate'C
     msg1 = ' PARAMETER SETUP ERROR'C
     iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  if (errorwindow) close (ErrWinUnit)
  return
end subroutine TimeSeries_Setup


subroutine TSSetup_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  integer(kind=4) id, callbacktype
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
  call dlgsetreturn(dlg, idok)
  call dlgexit(dlg)
  return
end subroutine TSSetup_OK


subroutine ts_paramset (dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
```

```
type(dialog) dlg
type(dialog) dlgparent
logical(kind=4) retlog, buttonfound
integer(kind=4) id, callbacktype, i
call unusedqq(dlgparent, id, callbacktype)
ButtonFound = .false.
do while (.not. ButtonFound)
  select case (id)
    case  IDC_TWMonthly, IDC_MLDMonthly, IDC_LDMonthly, IDC_InflowMonthly,&
          IDC_OutflowMonthly, IDC_TAMonthly, IDC_UMonthly, IDC_PAMonthly,&
          IDC_MTBEMonthly, IDC_AtmMTBEMonthly, IDC_InflowHeightMonthly,&
          IDC_OutflowHeightMonthly, IDC_EpiLossMonthly, IDC_HypLossMonthly)
      i = 1
      do while ((i .le. NumTimeSeries) .and. (.not. ButtonFound))
        if (MonthlyButtons(i) .eq. id) then
          TSSetup(i) = 1
          retlog = dlgset(dlg, RadButton(i,1), .true.)
          retlog = dlgset(dlg, RadButton(i,2), .false.)
          retlog = dlgset(dlg, RadButton(i,3), .false.)
          ButtonFound = .true.
          if (TSDatLen(i) .eq. 12) then
            retlog = DLGSET(dlg, TSOK(i), 'OK')
            TSError(i) = .false.
          else
            retlog = DLGSET(dlg, TSOK(i), 'ERR')
            TSError(i) = .true.
          endif
        endif
        i = i + 1
      end do
    case (IDC_TWWeekly, IDC_MLDWeekly, IDC_LDWeekly, IDC_InflowWeekly,&
          IDC_OutflowWeekly, IDC_TAWeekly, IDC_UWeekly, IDC_PAWeekly,&
          IDC_MTBEWeekly, IDC_AtmMTBEWeekly, IDC_InflowHeightWeekly, &
          IDC_OutflowHeightWeekly, IDC_EpiLossWeekly, IDC_HypLossWeekly)
      i = 1
      do while ((i .le. NumTimeSeries) .and. (.not. ButtonFound))
        if (WeeklyButtons(i) .eq. id) then
          TSSetup(i) = 2
          retlog = dlgset(dlg, RadButton(i,1), .false.)
          retlog = dlgset(dlg, RadButton(i,2), .true.)
          retlog = dlgset(dlg, RadButton(i,3), .false.)
          ButtonFound = .true.
          if (TSDatLen(i) .eq. 52) then
            retlog = DLGSET(dlg, TSOK(i), 'OK')
            TSError(i) = .false.
          else
            retlog = DLGSET(dlg, TSOK(i), 'ERR')
            TSError(i) = .true.
          endif
        endif
        i = i + 1
      end do
    case (IDC_TWDaily, IDC_MLDDaily, IDC_LDDaily, IDC_InflowDaily,&
          IDC_OutflowDaily, IDC_TADaily, IDC_UDaily, IDC_PADaily,&
          IDC_MTBEDaily, IDC_AtmMTBEDaily, IDC_InflowHeightDaily, &
          IDC_OutflowHeightDaily, IDC_EpiLossDaily, IDC_HypLossDaily)
      i = 1
      do while ((i .le. NumTimeSeries) .and. (.not. ButtonFound))
        if (DailyButtons(i) .eq. id) then
          TSSetup(i) = 3
```

```
            retlog = dlgset(dlg, RadButton(i,1), .false.)
            retlog = dlgset(dlg, RadButton(i,2), .false.)
            retlog = dlgset(dlg, RadButton(i,3), .true.)
            ButtonFound = .true.
            if (TSDatLen(i) .eq. 365) then
              retlog = DLGSET(dlg, TSOK(i), 'OK')
              TSError(i) = .false.
            else
              retlog = DLGSET(dlg, TSOK(i), 'ERR')
              TSError(i) = .true.
            endif
          endif
          i = i + 1
        end do
    end select
  end do
  return
  end subroutine ts_paramset


subroutine MTBE_Params (checked)
!***************************************************************************
!*  Subroutine to configure parameters associated with the physicochemical *
!*  variables for MTBE.  Also sets the motorboat inputs for MTBE.          *
!***************************************************************************
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use modelcom
  use tser_com
  use diffsolcom
  implicit none
  type(dialog)dlg
  logical(kind=4) lret, err
  integer(kind=4) iret, ierr, iloop
  external MTBEParam_OK, TimeSeriesEntry, DiffParamEntry, SolParamEntry,&
           RuntimeMTBEParam_OK, MTBEParam_Cancel
  logical(kind=4)checked
  call unusedqq(checked)
  ierr = 0
  msg0 = ''c
  msg1 = ''c
  if (menuactive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  err = .true.
  if (errorwindow) close (errwinunit)
  errorwindow = .false.
  iloop = 1
  do while ((.not. lrunning) .and. (err))
    menuactive = .true.
!   set temporary variables
    TempMW = MolWeight
    TempIC = Initial_MTBEConc
!   set the save variables
    SaveMW = MolWeight
    SaveIC = Initial_MTBEConc
!   initialize the dialog box
    lret = dlginit(IDD_MTBEParams, dlg)
    dlg_save = dlg
!   set the callback routines
    lret = dlgsetsub(dlg, IDOK, MTBEParam_OK)
    lret = dlgsetsub(dlg, IDCANCEL, MTBEParam_Cancel)
    lret = dlgsetsub(dlg, IDC_MTBEInputSeries, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_AtmMTBEConc, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_CallDiffParam, DiffParamEntry)
    lret = dlgsetsub(dlg, IDC_CallSolParam, SolParamEntry)
    lret = dlgsetsub(dlg, IDC_EpiLossRate, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_HypLossRate, TimeSeriesEntry)
!   write the current molecular weight into the dialog box edit field
    if (iloop .eq. 1) then
      write(err_str(1),'(f9.3)') TempMW
      write(err_str(2),'(f8.3)') TempIC
    endif
    lret = dlgset(dlg, IDC_MolWeight, err_str(1))
    lret = dlgset(dlg, IDC_InitialConc, err_str(2))
    call set_buttons(startVOC, endVOC, dlg)
!   bring up the dialog box
    iret = dlgmodal(dlg)
!   destroy and release the dialog resources
    call dlguninit(dlg)
    menuactive = .false.
    err = .false.
    if ((err_dlg(1)).or.(err_dlg(2)).or.(err_dlg(3)).or.(err_dlg(4)))then
      err = .true.
      call dialog_error_display(IDD_MTBEParams)
    endif
    iloop = iloop + 1
  enddo
  if ((lrunning) .and. (.not. pause_mod)) then
    msg0 = ' Model running: cannot change parameters\nPress <Run\\Stop> to&
             terminate'C
    msg1 = ' PARAMETER setUP ERROR'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  elseif (lrunning .and. pause_mod) then
    err = .true.
    if (errorwindow) close (errwinunit)
    errorwindow = .false.
    iloop = 1
    do while (err)
      menuactive = .true.
!     initialize the dialog box
      lret = dlginit(IDD_RuntimeMTBEParams, dlg)
      dlg_save = dlg
!     set the callback routines
      lret = dlgsetsub(dlg, IDOK, RuntimeMTBEParam_OK)
      lret = dlgsetsub(dlg, IDC_RuntimeMTBEInputSeries, TimeSeriesEntry)
      lret = dlgsetsub(dlg, IDC_RuntimeAtmMTBEConc, TimeSeriesEntry)
!     bring up the dialog box
      iret = dlgmodal(dlg)
!     destroy and release the dialog resources
      call dlguninit(dlg)
      menuactive = .false.
      err = .false.
      iloop = iloop + 1
    enddo
  endif
  if (errorwindow) close (ErrWinUnit)
  return
end subroutine MTBE_Params


subroutine MTBEParam_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use tser_com
  use errorcom
  use diffsolcom
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4) ret
  integer(kind=4) id, callbacktype, ierr1, ierr2
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
! Get the molecular weight
  err_dlg(1) = .FALSE.
  ret = dlgget(dlg, IDC_MolWeight, err_str(1))
  read(err_str(1),*,iostat=ierr1) TempMW
  if (ierr1 .eq. 0) then
    if (TempMW .le. 0.0) then
      err_dlg(1) = .TRUE.
    else
      MolWeight = TempMW
    endif
  else
    err_dlg(1) = .TRUE.
  endif
! Get the initial concentration
  err_dlg(2) = .FALSE.
  ret = dlgget(dlg, IDC_InitialConc, err_str(2))
  read(err_str(2),*,iostat=ierr2) TempIC
  if (ierr2 .eq. 0) then
    if (TempIC .lt. 0.0) then
      err_dlg(2) = .TRUE.
    else
      Initial_MTBEConc = TempIC
    endif
  else
    err_dlg(2) = .TRUE.
  endif
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine MTBEParam_OK


subroutine MTBEParam_Cancel(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use tser_com
  use errorcom
  use diffsolcom
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4) ret
  integer(kind=4) id, callbacktype, ierr1, ierr2
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
! clear the error codes
  err_dlg(1) = .FALSE.
  err_dlg(2) = .FALSE.
! reset the save variables
  MolWeight = SaveMW
  Initial_MTBEConc = SaveIC
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine MTBEParam_Cancel


subroutine RuntimeMTBEParam_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  integer(kind=4) id, callbacktype
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine RuntimeMTBEParam_OK


subroutine DiffParamEntry(dlg_parent,id,cbtype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg, dlg_parent
  logical(kind=4)ret, retlog
  integer(kind=4)iret, id, cbtype
  logical(kind=4)checked
! SHUTDOWN_PARENT and REset_PARENTDIALOG are contained in PAR_TSER.F90
  external Diff_Param, shutdown_parent, reset_parentdialog, DiffParam_OK, &
           DiffParam_Cancel, CalcScNum
  ts_entry_id = id      ! required for correct operation of REset_PARENTDIALOG
  call unusedqq(checked)
  iret = cbtype
  dlg_save = dlg_parent
  call shutdown_parent (dlg_parent, id)
  ret = dlginit(IDD_DiffParam, dlg)
  TempDiffParam = DiffParam
  tempMV = MolarVolume
  tempWD0 = WD0
  tempWD1 = WD1
  tempWD2 = WD2
  tempWD3 = WD3
  SaveDiffParam = DiffParam
  SaveMV = MolarVolume
  SaveWD0 = WD0
  SaveWD1 = WD1
  SaveWD2 = WD2
  SaveWD3 = WD3
  write(ctemp,'(a)') '1.0'
  retlog = dlgset(dlg, IDC_ScDay, ctemp)
  call CalcScNum(dlg, IDC_CalcSc, cbtype)
  write(ctemp,'(g12.4)') TempMV
  retlog = dlgset(dlg, IDC_MolarVolume, ctemp)
  write(ctemp,'(g12.4)') TempWD0
  retlog = dlgset(dlg, IDC_Wank_a0, ctemp)
  write(ctemp,'(g12.4)') TempWD1
  retlog = dlgset(dlg, IDC_Wank_a1, ctemp)
  write(ctemp,'(g12.4)') TempWD2
  retlog = dlgset(dlg, IDC_Wank_a2, ctemp)
  write(ctemp,'(g12.4)') TempWD3
  retlog = dlgset(dlg, IDC_Wank_a3, ctemp)
  select case (TempDiffParam)
    case (1)
      retlog = dlgset(dlg, IDC_DiffButtWilk, .true.)
      retlog = dlgset(dlg, IDC_DiffButtWann, .false.)
    case (2)
      retlog = dlgset(dlg, IDC_DiffButtWilk, .false.)
      retlog = dlgset(dlg, IDC_DiffButtWann, .true.)
  end select
  retlog = dlgsetsub(dlg, IDC_DiffButtWilk, Diff_Param)
  retlog = dlgsetsub(dlg, IDC_DiffButtWann, Diff_Param)
  retlog = dlgsetsub(dlg, IDC_CalcSc, CalcScNum)
  retlog = dlgsetsub(dlg, IDOK, DiffParam_OK)
! bring up the dialog box
  iret = dlgmodal(dlg)
! destroy and release the dialog resources
  call dlguninit(dlg)
! restore calling dialog box
  call reset_parentdialog (dlg_parent)
  dlg_save = dlg_parent
  return
end subroutine DiffParamEntry


subroutine Diff_Param (dlg, id, cbtype)
! callback subroutine for the choice in diffusivity parameterization
! radio buttons
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg
  logical(kind=4) ret
  integer(kind=4) id, cbtype, idum, iret
  character*255 msg3
  idum = cbtype
  select case (id)
```

```
    case (IDC_DiffButtWilk)
      TempDiffParam = 1
      ret = dlgset(dlg, IDC_DiffButtWilk, .true.)
      ret = dlgset(dlg, IDC_DiffButtWann, .false.)
    case (IDC_DiffButtWann)
      TempDiffParam = 2
      ret = dlgset(dlg, IDC_DiffButtWilk, .false.)
      ret = dlgset(dlg, IDC_DiffButtWann, .true.)
      msg1 = 'Information Message for using the Wanninkhof parameterization'C
      msg3 = 'The coefficients in Wanninkhof (1992) give Schmidt numbers.\n&
              These will be converted to diffusivities internally by the program.\n&
              There is no need to modify the coefficients in Wanninkhof (1992).'C
      iret = messageboxqq(msg3, msg1, MB$OK)
  end select
  return
end subroutine Diff_Param


subroutine DiffParam_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret, lerr, str1, str2, str3, str4
  integer(kind=4)iret, id, callbacktype, ierr1, ierr2,ierr3,ierr4, i
  real*8 t,D,diff_calc
  external diff_calc
  msg1 = 'Error reading information'C
  call unusedqq(dlgparent, id, callbacktype)
  lerr = .FALSE.
  select case (TempDiffParam)
    case (1)      !Wilke-Chang diffusivity
      ret = dlgget(dlg, idc_molarvolume, ctemp)
      read(ctemp,*,iostat = ierr1) TempMV
      if ((ierr1 .ne. 0) .or. (TempMV .le. 0.0)) then
        msg0 = 'Error reading molar volume\nVolume must be >0 and numeric'C
        iret = messageboxqq(msg0, msg1, MB$OK)
        ret = dlgset(dlg, IDC_MolarVolume, ctemp)
        lerr = .TRUE.
      else
        MolarVolume = TempMV
      endif
    case(2)       !Wanninkhof polynomial expression
      iret = dlgget(dlg, IDC_Wank_a0, ctemp)
      read(ctemp,*,iostat=ierr1) TempWD0
      if (ierr1 .ne. 0) then
        msg0 = 'Error reading coefficient a0'C
        lerr = .TRUE.
      else
        WD0 = TempWD0
      endif
      iret = dlgget(dlg, IDC_Wank_a1, ctemp)
      read(ctemp,*,iostat=ierr2) TempWD1
      if (ierr2 .ne. 0) then
        msg0 = 'Error reading coefficient a1'C
        lerr = .TRUE.
      else
        WD1 = TempWD1
      endif
      iret = dlgget(dlg, IDC_Wank_a2, ctemp)
      read(ctemp,*,iostat=ierr3) TempWD2
      if (ierr3 .ne. 0) then
        msg0 = 'Error reading coefficient a2'C
        lerr = .TRUE.
      else
        WD2 = TempWD2
      endif
      iret = dlgget(dlg, IDC_Wank_a3, ctemp)
      read(ctemp,*,iostat=ierr4) TempWD3
      if (ierr4 .ne. 0) then
        msg0 = 'Error reading coefficient a3'C
        lerr = .TRUE.
      else
        WD3 = TempWD3
      endif
      if (lerr) iret = messageboxqq(msg0, msg1, MB$OK)
  end select
  i = 1
  do while ((.not. lerr) .and. (i .le. 40))
    t = float(i)
    D = diff_calc(t, tempDiffParam)
    if ((D .le. 0.0) .and. (.not. lerr)) then
      lerr = .true.
      msg0 = 'Calculated diffusivity <= 0\nCheck coefficients'C
      iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    endif
    i = i + 1
  end do
  if(.NOT. lerr) then
    DiffParam = TempDiffParam
    call dlgsetreturn(dlg, IDOK)
    call dlgexit(dlg)
  endif
  return
end subroutine DiffParam_OK


subroutine DiffParam_Cancel(dlg, id, callbacktype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret, lerr, str1, str2, str3, str4
  integer(kind=4)iret, id, callbacktype, ierr1, ierr2, ierr3, ierr4, i
  real*8 t,D,diff_calc
  external diff_calc
  call unusedqq(dlgparent, id, callbacktype)
  DiffParam = SaveDiffParam
  select case (DiffParam)
    case (1)      !Wilke-Chang diffusivity
```

```
      MolarVolume = SaveMV
    case(2)       !Wanninkhof polynomial expression
      WD0 = SaveWD0
      WD1 = SaveWD1
      WD2 = SaveWD2
      WD3 = SaveWD3
  end select
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine DiffParam_Cancel


subroutine CalcScNum (dlg, id, cbtype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg
  logical(kind=4) ret, checked
  integer(kind=4) iret, id, cbtype, ierr1
  real*8 TempDay, degc, nu, sc, diff
! external function declarations...
  real*8 interpolate, diff_calc
  external interpolate, diff_calc
! end external function declarations
  ret = dlgget(dlg, IDC_ScDay, ctemp)
  read(ctemp, *, iostat = ierr1) TempDay
  if ((ierr1 .ne. 0) .or. (TempDay .lt. 0.0).or. (TempDay .gt. 365.0)) then
    msg1 = 'Error reading information'C
    msg0 = 'Error reading calculation time\n365 > Time > 0 and numeric'C
    iret = messageboxqq(msg0, msg1, MB$OK)
    ret = dlgset(dlg, IDC_ScDay, ctemp)
    return
  else
    TempDay = 12.0*TempDay/365.0
    degc = interpolate(spl_SurfaceTemp, TempDay, splinepnts)
    diff = diff_calc(degc, DiffParam)
!   Kin. Visc. (nu) in cm^2/sec is calculated from temperature in deg-C.
!   The underlying data are from CRC 63rd edition.
!   The polynomial fit was done in the spreadsheet KINVISC.WB1 in QDATA
    nu = 0.017826598 - 5.76464E-04*degc +  1.12266E-05*degc**2 - &
         9.66507E-08*degc**3
    sc = nu/diff
    write (ctemp, '(f8.1)') sc
    ret = dlgset(dlg, IDC_ScVal, ctemp)
    return
  endif
  return
end subroutine CalcScNum


subroutine SolParamEntry(dlg_parent,id,cbtype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg, dlg_parent
  logical(kind=4)ret, retlog
  integer(kind=4)iret, id, cbtype
  logical(kind=4)checked
  external Sol_Param, shutdown_parent, reset_parentdialog, SolParam_OK,&
           SolParam_Cancel, CalcHSol
  ts_entry_id = id      ! required for correct operation of RESET_PARENTDIALOG
  call unusedqq(checked)
  iret = cbtype
  call shutdown_parent (dlg_parent, id)
  ret = dlginit(IDD_SolParam, dlg)
  TempSolParam = SolParam
  tempSolA = SolA
  tempSolB = SolB
  tempWA0 = WA0
  tempWA1 = WA1
  tempWA2 = WA2
  tempWB0 = WB0
  tempWB1 = WB1
  tempWB2 = WB2
  tempSal = Salinity
  SaveSolParam = SolParam
  SaveSolA = SolA
  SaveSolB = SolB
  SaveWA0 = WA0
  SaveWA1 = WA1
  SaveWA2 = WA2
  SaveWB0 = WB0
  SaveWB1 = WB1
  SaveWB2 = WB2
  SaveSal = Salinity
  write(ctemp,'(a)') '1.0'
  retlog = dlgset(dlg, IDC_HDay, ctemp)
  call CalcHSol(dlg, IDC_CalcH, cbtype)
  write(ctemp,'(g12.4)') TempSolA
  retlog = dlgset(dlg, IDC_RobbinsA, ctemp)
  write(ctemp,'(g12.4)') TempSolB
  retlog = dlgset(dlg, IDC_RobbinsB, ctemp)
  write(ctemp,'(g12.4)') TempWA0
  retlog = dlgset(dlg, IDC_Wank_a0_sol, ctemp)
  write(ctemp,'(g12.4)') TempWA1
  retlog = dlgset(dlg, IDC_Wank_a1_sol, ctemp)
  write(ctemp,'(g12.4)') TempWA2
  retlog = dlgset(dlg, IDC_Wank_a2_sol, ctemp)
  write(ctemp,'(g12.4)') TempWB0
  retlog = dlgset(dlg, IDC_Wank_b0_sol, ctemp)
  write(ctemp,'(g12.4)') TempWB1
  retlog = dlgset(dlg, IDC_Wank_b1_sol, ctemp)
  write(ctemp,'(g12.4)') TempWB2
  retlog = dlgset(dlg, IDC_Wank_b2_sol, ctemp)
  write(ctemp,'(g12.4)') TempSal
  retlog = dlgset(dlg, IDC_Wank_Salinity, ctemp)
  select case (TempSolParam)
    case (1)
      retlog = dlgset(dlg, IDC_SolButtRobbins, .true.)
      retlog = dlgset(dlg, IDC_SolButtWann, .false.)
    case (2)
```

```
      retlog = dlgset(dlg, IDC_SolButtRobbins, .false.)
      retlog = dlgset(dlg, IDC_SolButtWann, .true.)
  end select
  retlog = dlgsetsub(dlg, IDC_SolButtRobbins, Sol_Param)
  retlog = dlgsetsub(dlg, IDC_SolButtWann, Sol_Param)
  retlog = dlgsetsub(dlg, IDC_CalcH, CalcHSol)
  retlog = dlgsetsub(dlg, IDOK, SolParam_OK)
! bring up the dialog box
  iret = dlgmodal(dlg)
! destroy and release the dialog resources
  call dlguninit(dlg)
! restore calling dialog box
  call reset_parentdialog (dlg_parent)
  dlg_save = dlg_parent
  return
end subroutine SolParamEntry


subroutine Sol_Param (dlg, id, cbtype)
! callback subroutine for the choice in diffusivity parameterization radio buttons
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg
  logical(kind=4) ret
  integer(kind=4) id, cbtype, idum, iret
  character*255 msg3
  idum = cbtype
  select case (id)
    case (IDC_SolButtRobbins)
      TempSolParam = 1
      ret = dlgset(dlg, IDC_SolButtRobbins, .true.)
      ret = dlgset(dlg, IDC_SolButtWann, .false.)
    case (IDC_SolButtWann)
      TempSolParam = 2
      ret = dlgset(dlg, IDC_SolButtRobbins, .false.)
      ret = dlgset(dlg, IDC_SolButtWann, .true.)
      msg1 = 'Information message for using Wanninkhof parameterization'C
      msg3 = 'The program assumes the coefficients give solubility as the\n&
              dimensionless Ostwald number (converted to mol/m^3-atm internally)\n&
              You must modify the coefficients in Wanninkhof (1992) if they give\n&
              the solubility as a dimensioned number (e.g. CO2)'C
      iret = messageboxqq(msg3,msg1,MB$OK)
  end select
  return
end subroutine Sol_Param


subroutine SolParam_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
```

```
type(dialog)dlg
type(dialog)dlgparent
logical(kind=4)ret, lerr
integer(kind=4) iret, id, callbacktype, ierr1,ierr2,ierr3,ierr4,ierr5,&
                ierr6, ierr7, i
real*8 t,a,sol_calc, ssola, ssolb, swa0, swa1, swa2, swb0, swb1, swb2, ssal
external sol_calc
msg1 = 'error reading information'C
call unusedqq(dlgparent, id, callbacktype)
lerr = .FALSE.
select case (TempSolParam)
  case (1)      !Robbins solubility
    ssola = SolA
    ssolb = SolB
    ret = dlgget(dlg, IDC_RobbinsA, ctemp)
    read(ctemp, *, iostat = ierr1) TempSolA
    if ((.not. lerr) .and. (ierr1 .ne. 0)) then
      msg0 = 'Error reading coefficient A'C
      iret = messageboxqq(msg0, msg1,MB$OK)
      lerr = .TRUE.
    else
      SolA = TempSolA
    endif
    ret = dlgget(dlg, IDC_RobbinsB, ctemp)
    read(ctemp, *, iostat = ierr2) TempSolB
    if ((ierr2 .ne. 0) .and. (.not. lerr)) then
      msg0 = 'Error reading coefficient B'C
      iret = messageboxqq(msg0, msg1, MB$OK)
      lerr = .TRUE.
    else
      SolB = TempSolB
    endif
  case(2)       !Wanninkhof polynomial expression
    swa0 = wa0
    swa1 = wa1
    swa2 = wa2
    swb0 = wb0
    swb1 = wb1
    swb2 = wb2
    ssal = salinity
    iret = dlgget(dlg, IDC_Wank_a0_sol, ctemp)
    read(ctemp, *, iostat = ierr1) TempWA0
    if ((ierr1 .ne. 0) .and. (.not. lerr)) then
      msg0 = 'Error reading coefficient a0'C
      lerr = .TRUE.
    else
      WA0 = TempWA0
    endif
    iret = dlgget(dlg, IDC_Wank_a1_sol, ctemp)
    read(ctemp, *, iostat = ierr2) TempWA1
    if ((ierr2 .ne. 0) .and. (.not. lerr)) then
      msg0 = 'Error reading coefficient a1'C
      lerr = .TRUE.
    else
      WA1 = TempWA1
    endif
    iret = dlgget(dlg, IDC_Wank_a2_sol, ctemp)
    read(ctemp, *, iostat = ierr3) TempWA2
    if ((ierr3 .ne. 0) .and. (.not. lerr)) then
      msg0 = 'Error reading coefficient a2'C
```

```
        lerr = .TRUE.
      else
        WA2 = TempWA2
      endif
      iret = dlgget(dlg, IDC_Wank_b0_sol, ctemp)
      read(ctemp, *, iostat = ierr4) TempWB0
      if ((ierr4 .ne. 0) .and. (.not. lerr)) then
        msg0 = 'Error reading coefficient B1'C
        lerr = .TRUE.
      else
        WB0 = TempWB0
      endif
      iret = dlgget(dlg, IDC_Wank_b1_sol, ctemp)
      read(ctemp, *, iostat = ierr5) TempWB1
      if ((ierr5 .ne. 0) .and. (.not. lerr)) then
        msg0 = 'Error reading coefficient B2'C
        lerr = .TRUE.
      else
        WB1 = TempWB1
      endif
      iret = dlgget(dlg, IDC_Wank_b2_sol, ctemp)
      read(ctemp, *, iostat = ierr6) TempWB2
      if ((ierr6 .ne. 0) .and. (.not. lerr)) then
        msg0 = 'Error reading coefficient B3'C
        lerr = .TRUE.
      else
        WB2 = TempWB2
      endif
      if ((wa0 .eq. 0.) .and. (wa1 .eq. 0.) .and. (wa2 .eq. 0.) .and. &
          (wb0 .eq. 0.) .and. (wb1 .eq. 0.) .and. (wb2 .eq. 0.)) then
        msg0 = 'Error:  All coefficients cannot = 0.0'C
        lerr = .TRUE.
      endif
      iret = dlgget(dlg, IDC_Wank_Salinity, ctemp)
      read(ctemp, *, iostat = ierr7) TempSal
      if (((TempSal .lt. 0.0) .or. (ierr7 .ne. 0)) .and. (.not. lerr)) then
        msg0 = 'Error reading salinity'C
        lerr = .TRUE.
      else
        salinity = TempSal
      endif
  end select
  if (lerr) iret = messageboxqq(msg0, msg1,MB$OK)
  i = 1
  do while ((.not. lerr) .and. (i .le. 40))
    t = float(i)
    a = sol_calc(t, TempSolParam)
    if (a .le. 0.0) then
      lerr = .true.
      msg0 = 'Calculated solubility <0\nCheck coefficients'C
      iret = messageboxqq(msg0, msg1,MB$OK)
    endif
    i = i + 1
  end do
  if (.NOT. lerr) then
    SolParam = TempSolParam
    call dlgsetreturn(dlg, IDOK)
    call dlgexit(dlg)
  else
!   reset Solubility parameters to entering values
```

```
    select case (TempSolParam)
      case (1)
        SolA = ssola
        SolB = ssolb
      case (2)
        wa0 = swa0
        wa1 = swa1
        wa2 = swa2
        wb0 = swb0
        wb1 = swb1
        wb2 = swb2
        salinity = ssal
    end select
  endif
  return
end subroutine SolParam_OK


subroutine SolParam_Cancel(dlg, id, callbacktype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4)ret, lerr
  integer(kind=4)iret, id, callbacktype
  msg1 = 'Error reading information'C
  call unusedqq(dlgparent, id, callbacktype)
  lerr = .FALSE.
  SolParam = SaveSolParam
! reset Solubility parameters to entering values
  select case (SolParam)
    case (1)
      SolA = SaveSola
      SolB = SaveSolb
    case (2)
      wa0 = Savewa0
      wa1 = Savewa1
      wa2 = Savewa2
      wb0 = Savewb0
      wb1 = Savewb1
      wb2 = Savewb2
      salinity = SaveSal
  end select
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine SolParam_Cancel


subroutine CalcHSol (dlg, id, cbtype)
  use msflib
  use dialogm
  use tser_com
  use mtbecom
  use errorcom
  use diffsolcom
  implicit none
  type(dialog) dlg
  logical(kind=4) ret, checked
  integer(kind=4) iret, id, cbtype, ierr1
  real*8 TempDay, degc, sol
! external function declarations...
  real*8 interpolate, sol_calc
  external interpolate, sol_calc
! end external function declarations
  ret = dlgget(dlg, IDC_HDay, ctemp)
  read(ctemp, *, iostat = ierr1) TempDay
  if ((ierr1 .ne. 0) .or. (TempDay .lt. 0.0).or. (TempDay .gt. 365.0)) then
     msg1 = 'Error reading information'C
     msg0 = 'Error reading calculation time\n365 > Time > 0 and numeric'C
     iret = messageboxqq(msg0, msg1, MB$OK)
     ret = dlgset(dlg, IDC_HDay, ctemp)
     return
  else
     TempDay = 12.0*TempDay/365.0
     degc = interpolate(spl_SurfaceTemp, TempDay, splinepnts)
     sol = sol_calc(degc, SolParam)
     write (ctemp, '(e10.4)') sol
     ret = dlgset(dlg, IDC_HVal, ctemp)
     return
  endif
  return
end subroutine CalcHSol


subroutine Meteor_Params(checked)
!****************************************************************************
!*  this subroutine creates a dialog box that allows the user to specify   *
!*  the parameters associated with the meteorological conditions.          *
!****************************************************************************
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  logical(kind=4) lret
  integer(kind=4) iret, ierr, iloop
  external TimeSeriesEntry, Meteor_OK, Meteor_Cancel
  logical(kind=4)checked, err
  call unusedqq(checked)
  ierr = 0
  msg0 = ''c
  msg1 = ''c
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  err = .true.
  if (errorwindow) close (errwinunit)
  errorwindow = .false.
  iloop = 1
```

```
! initialize the temporary value of Rel_Hum
  tempRel_hum = 100.0* rel_hum
! initialize the save value of Rel_Hum
  saveRel_hum = 100.0* rel_hum
  do while ((.not. lrunning) .and. (err))
    menuactive = .true.
!   initialize the dialog box
    lret = dlginit(IDD_MeteorParams, DLG)
    dlg_save = dlg
!   write initial values and set subroutines
    lret = dlgsetsub(dlg, IDOK, Meteor_OK)
    lret = dlgsetsub(dlg, IDC_WindSpeed, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_AirTemp, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_AtmPressure, TimeSeriesEntry)
    if (iloop .eq. 1) then
      write(err_str(1), '(f7.2)') tempRel_Hum
    endif
    lret = dlgset(dlg, IDC_RelativeHumidity, err_str(1))
    call set_buttons(startAtm, endAtm, dlg)
!   initiate the dialog window
    iret = dlgmodal(dlg)
!   terminate the dialog resources
    call dlguninit(dlg)
    menuactive = .false.
    err = .false.
    err = .false.
    if (err_dlg(1))then
      err = .true.
      call dialog_error_display(IDD_MeteorParams)
    endif
    iloop = iloop + 1
  enddo
  if (lrunning) then
    msg0 = ' Model running: cannot change parameters\nPress <Run\\Stop> to&
             terminate'C
    msg1 = ' PARAMETER SETUP ERROR'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  if (errorwindow) close (ErrWinUnit)
  return
end subroutine Meteor_Params


subroutine Meteor_OK(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4) ret
  integer(kind=4)id, callbacktype, ierr1
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
! Get the relative humidity
  err_dlg(1) = .FALSE.
  ret = dlgget(dlg, IDC_RelativeHumidity, err_str(1))
  read(err_str(1),*,iostat=ierr1) TempRel_Hum
  if (ierr1 .eq. 0) then
    if ((TempRel_Hum .lt. 0.0) .or. (TempRel_Hum .gt. 100.0)) then
      err_dlg(1) = .TRUE.
    else
      rel_hum = 0.01 * TempRel_Hum
    endif
  else
    err_dlg(1) = .TRUE.
  endif
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine Meteor_OK


subroutine Meteor_Cancel(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  logical(kind=4) ret
  integer(kind=4)id, callbacktype, ierr1
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
! clear the error code
  err_dlg(1) = .FALSE.
! reset the value
  rel_hum = saveRel_Hum
! close the dialog
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  return
end subroutine Meteor_Cancel


subroutine Hydrog_Params(checked)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog)dlg
  logical(kind=4)lret, err
  integer(kind=4)iret, ierr, iloop
  external hydrog_ok, timeseriesentry, enterdepthprofile, viewdepthprofile
  logical(kind=4)checked
  call unusedqq(checked)
  ierr = 0
  msg0 = ''c
  msg1 = ''c
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  err = .true.
  if (errorwindow) close (errwinunit)
  errorwindow = .false.
  TempPP = ProfilePoints
  iloop = 1
  do while ((.not. lrunning) .and. (err))
    menuactive = .true.
!   initialize the dialog box
    lret = dlginit(IDD_HydrogParams, dlg)
    dlg_save = dlg
    if (iloop .eq. 1) then
      write(err_str(1),'(i5)') TempPP
    endif
    lret = dlgsetsub(dlg, IDC_MixedLayer, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_SurfaceTemp, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_LakeDepth, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_Inflow, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_Outflow, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_InflowHeight, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_OutflowHeight, TimeSeriesEntry)
    lret = dlgsetsub(dlg, IDC_LakeArea, EnterDepthProfile)
    lret = dlgsetsub(dlg, IDC_LakeArea2, ViewDepthProfile)
    lret = dlgset(dlg, IDC_ProfilePoints, err_str(1))
    lret = dlgsetsub(dlg, IDOK, Hydrog_OK)
    call set_buttons(startHydro, endHydro, dlg)
!   bring up the dialog box
    iret = dlgmodal(dlg)
!   destroy and release the dialog resources
    call dlguninit(dlg)
    menuactive = .false.
    err = .false.
    if (err_dlg(1)) then
      err = .true.
      call dialog_error_display(IDD_HydrogParams)
    endif
    iloop = iloop + 1
  enddo
  if (lrunning) then
    msg0 = ' Model running: cannot change parameters\nPress <Run\\Stop> &
             to terminate'C
    msg1 = ' PARAMETER SETUP ERROR'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  if (errorwindow) close (ErrWinUnit)
  ErrorWindow = .false.
  return
end subroutine Hydrog_Params


subroutine Hydrog_OK(dlg, id, callbacktype)
!****************************************************************************
!*  callback routine for when the ok button has been pushed                *
!*  it contains code to sense if an invalid entry in the edit box has been *
!*  made.  if an error occurs, an error message is printed and the edit box*
!*  is reset back to its original value.  if no error occurs, the dialog   *
!*  return value is the id for the ok button                               *
!****************************************************************************
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog)dlg
  type(dialog)dlgparent
  integer(kind=4)id, callbacktype, ierr1, ierr2
  logical ret
  call unusedqq(dlgparent, id, callbacktype)
  if (errorwindow) close (ErrWinUnit)
  call dlgsetreturn(dlg, IDOK)
  call dlgexit(dlg)
  call lake_volume_calc
  return
end subroutine Hydrog_OK


subroutine EnterDepthProfile(dlg_parent, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog) dlg_parent, dlg_child
  logical(kind=4)lret
  integer(kind=4)id, iret, ierr1, i, OldPoints, callbacktype
  character*29 DataString
  external Profile_OK, EnterPoint
  call unusedqq(dlg_parent, ID, callbacktype)
! save the old number of profile points
  OldPoints = ProfilePoints
  PointsChanged = 0
  lret = dlgget(dlg_parent, IDC_ProfilePoints, err_str(1))
  read(err_str(1),*,iostat=ierr1) TempPP
  if (ierr1 .eq. 0) then
    if (TempPP .le. 0) then
      err_dlg(1) = .TRUE.
    else
      ProfilePoints = TempPP
    endif
  endif
  if (err_dlg(1)) then
    msg1 = 'Parameter Input Error'C
    msg0 = 'Invalid Number of Profile Points'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    TempPP = OldPoints
    return
  endif
! get the temporary area array ready for the data
  if (allocated(TempArea)) deallocate(TempArea)
  allocate (TempArea(ProfilePoints,2))
  do i = 1, min(OldPoints, ProfilePoints)
    TempArea(i,1) = LakeArea(i,1)
    TempArea(i,2) = LakeArea(i,2)
  end do
! save the dialog info and the id of the calling routine....
  dlg_save = dlg_parent
! setting ts_entry_id is necessary for correct operation of reset_parentdialog
  ts_entry_id = id
  lret = dlgset(dlg_parent, IDC_ProfilePoints, err_str(1))
! first need to close the parent dialog and save the temporary data
  call shutdown_parent (dlg_parent, id)
! now initialize the new dialog
  lret = dlginit(IDD_LakeDepthProfileEntry, dlg_child)
  lret = dlgset(dlg_child, IDC_NumPointsChanged, '0')
  lret = dlgsetsub(dlg_child, IDC_EnterPoint, EnterPoint)
  lret = dlgsetsub(dlg_child, IDOK, Profile_OK)
  lret = dlgset(dlg_child, IDC_LakeDepthProfile, ProfilePoints)
  lret = dlgset(dlg_child, IDC_LAErrorMessage, ''C)
  do i = 1, ProfilePoints
    write (DataString, '(1x,i4,3x,f8.2,e12.3)') i, TempArea(i,1),&
           TempArea(i,2)
    lret = dlgset(dlg_child, IDC_LakeDepthProfile, DataString, i)
  end do
  iret = dlgmodal(dlg_child)
! release the dialog resources
  call dlguninit(dlg_child)
! Close the Lake Area profile if open
  if (ErrorWindow) then
    close (errwinunit)
    ErrorWindow = .false.
  endif
! reset the parent dialog (ParHydro)
  call reset_parentdialog (dlg_parent)
  dlg_save = dlg_parent
  return
end subroutine EnterDepthProfile


Subroutine EnterPoint(dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog) dlg
  logical(kind=4)lret, error
  integer(kind=4)id, iret, ierr1, i, ispace, ilength, callbacktype
  character*45 DataString, DummyString, istring, dstring, astring
  character*140 error_msg
  real*8 depth, area
  call unusedqq(dlg, id, callbacktype)
  iret = id
  error = .false.
  lret = dlgget(dlg, IDC_LakeDepthProfile, DataString)
! check to make sure there is something in the string just read
  if (DataString .eq. '') return
  if (DataString(1:1) .eq. 'm') DataString(1:1) = ' '
  DummyString = trim(adjustl(DataString))
  ilength = len_trim(DummyString)
  ispace = scan(DummyString, ' ')
  istring = ''
  istring(1:ispace-1) = DummyString(1:ispace-1)
  istring = trim(adjustl(istring))
  DummyString = DummyString(ispace:ilength)
  DummyString = trim(adjustl(DummyString))
  ilength = len_trim(DummyString)
  ispace = scan(DummyString, ' ')
  dstring = DummyString(1:ispace-1)
  astring = DummyString(ispace:ilength)
  dstring = trim(adjustl(dstring))
  astring = trim(adjustl(astring))
  read (istring, *, iostat = ierr1) i
  if (ierr1 .ne. 0) then
    error_msg = 'Error reading current data point\nvalid format is:\n&
                 pnt#   depth   area'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    error = .true.
    return
  endif
  if (i .gt. profilepoints) then
    error_msg = 'Error:  Point Number > Max Points\nPoint ignored'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    error = .true.
    return
  endif
  ilength = len_trim(dstring)
  read (dstring, *, iostat = ierr1) depth
  if (ierr1 .ne. 0) then
    error_msg = 'Error reading current data point\nvalid format is:\n  pnt#&
                 depth   area'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    error = .true.
    return
  endif
  ilength = len_trim(astring)
  read (astring, *, iostat = ierr1) area
  if (ierr1 .ne. 0) then
    error_msg = 'Error reading current data point\nvalid format is:\n  pnt#&
                 depth   area'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    error = .true.
    return
  endif
  if ((i .eq. 1) .and. (depth .lt. maxdepth)) then
    error_msg = 'Lake Area Depth > Current Max Depth\nPossible data error'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    error = .true.
  endif
  if (depth .ge. 0.0) TempArea(i,1) = depth
  if (area .ge. 0.0) TempArea(i,2) = area
  if (DataString(1:1) .eq. ' ') then
    DataString(1:1) = 'm'
  else
    ilength = len_trim(DataString)
    DataString(2:ilength+1) = DataString(1:ilength)
    DataString(1:1) = 'm'
  endif
  lret = dlgset(dlg, IDC_LakeDepthProfile, DataString, i)
  PointsChanged = PointsChanged + 1
  write (istring, '(i2)') PointsChanged
  lret = dlgset(dlg, IDC_NumPointsChanged, istring)
  if (.not. error) lret = dlgset(dlg, IDC_LAErrorMessage, 'Point OK'C)
  return
end subroutine EnterPoint


subroutine Profile_OK (dlg, id, callbacktype)
```

```
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  logical lret, error
  character*150 error_msg
  type(dialog)dlg
  integer(kind=4)id, iret, i, callbacktype
  call unusedqq(dlg, id, callbacktype)
  iret = id
  error = .false.
  do i = 1, 365
    if (TempArea(1,1) .lt. spl_LakeDepth(i)) then
      error = .true.
      error_msg = 'Error in Lake Area profile\n Max. Depth < Lake Depth'C
      lret = dlgset (dlg, IDC_LAErrorMessage, error_msg)
    endif
  end do
  do i = 1, profilepoints-1
    if (temparea(i,2) .lt. temparea(i+1,2)) then
      error = .true.
      error_msg = 'Error in Lake Area profile\n&
                   Lake Area(i) >= Lake Area(i+1)\n&
                   (i.e., areas decrease with depth)'C
      lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
    endif
  end do
  if (temparea(profilepoints,1) .ne. 0.0) then
    error = .true.
    error_msg = 'Error in Lake Area profile\nFinal Depth must equal zero'C
    lret = dlgset(dlg, IDC_LAErrorMessage, error_msg)
  endif
  if (.not. error) then
    deallocate (LakeArea)
    allocate (LakeArea(ProfilePoints, 2))
    do i = 1, ProfilePoints
      LakeArea(i,1) = TempArea(i,1)
      LakeArea(i,2) = TempArea(i,2)
    end do
    if (LakeArea(1,1) .gt. MaxDepth) MaxDepth = LakeArea(1,1)
    deallocate(TempArea)
    call dlgsetreturn(dlg, IDOK)
    call dlgexit(dlg)
    call lake_volume_calc
  endif
  return
end subroutine Profile_OK


subroutine ViewDepthProfile(dlg_parent, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type(dialog) dlg_parent
  logical lret
  integer id, callbacktype
  integer i, iret
  call unusedqq(dlg_parent, ID, callbacktype)
! save the dialog info and the id of the calling routine....
  dlg_save = dlg_parent
! setting ts_entry_id is necessary for correct operation of reset_parentdialog
  ts_entry_id = id
  lret = dlgset(dlg_parent, IDC_ProfilePoints, err_str(1))
! first need to close the parent dialog and save the temporary data
  call shutdown_parent (dlg_parent, id)
  open (ErrWinUnit, file='USER', title='Lake Area versus Depth Profile')
  ErrorWindow = .true.
  write (errwinunit, '(a)') ' Point Number     Depth (m)      Area (m^2)'
  do i = 1, profilepoints
     write (errwinunit, '(5x,i3,9x,f10.3,g16.4)') i, LakeArea(i,1),&
            LakeArea(i,2)
  end do
  msg0 = 'Press OK\nto Continue'C
  msg1 = 'Information'C
  iret = messageboxqq(msg0, msg1, MB$OK)
! reset the parent dialog (ParHydro)
  dlg_save = dlg_parent
  call reset_parentdialog (dlg_parent)
  return
end subroutine ViewDepthProfile


subroutine parfilin(checked)
  use msflib
  use inputinfo
  use msfwinty
  use msfwin
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type (t_openfilename)fred
  logical(kind=4)ret, ts_error
  integer(kind=4)ierror, i
  character(len=26)filter(7)
  character(len=60)dlgtitle
  logical(kind=4)checked
  external read_parfile
  call unusedqq(checked)
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    ierror = messageboxqq(msg0, msg1,MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  if (errorwindow) close (ErrWinUnit)
  errorwindow = .false.
  i = 1
  do while ((.not. ts_error) .and. (i .le. NumTimeSeries))
    if (TSError(i)) ts_error = .true.
    i = i + 1
  enddo
  filter(1) = 'Parameter File (*.PAR)  'C
  filter(2) = '*.par                   'C
  filter(3) = 'All Files (*.*)         'C
  filter(4) = '*.*                     'C
  filter(5) = ''C
  filter(6) = ''C
  filter(7) = ''C
  dlgtitle = 'Read Parameter File'C
  fred%lstructsize = (bit_size(fred%lstructsize) +        &
                      bit_size(fred%hwndowner) +        &
                      bit_size(fred%hinstance) +        &
                      bit_size(fred%lpstrfilter) +        &
                      bit_size(fred%lpstrcustomfilter) +        &
                      bit_size(fred%nmaxcustfilter) +        &
                      bit_size(fred%nfilterindex) +        &
                      bit_size(fred%lpstrfile) +        &
                      bit_size(fred%nmaxfile) +        &
                      bit_size(fred%lpstrfiletitle) +        &
                      bit_size(fred%nmaxfiletitle) +        &
                      bit_size(fred%lpstrinitialdir) +        &
                      bit_size(fred%lpstrtitle) +        &
                      bit_size(fred%flags) +        &
                      bit_size(fred%nfileoffset) +        &
                      bit_size(fred%nfileextension) +        &
                      bit_size(fred%lpstrdefext) +        &
                      bit_size(fred%lcustdata) +        &
                      bit_size(fred%lpfnhook) +        &
                      bit_size(fred%lptemplatename))/8
  fred%hwndowner = null
  fred%hinstance = null
  fred%lpstrfilter = loc(filter(1))
  fred%lpstrcustomfilter = null
  fred%nmaxcustfilter = null
  fred%nfilterindex = 1
  fred%lpstrfile = loc(parfile_in)
  fred%nmaxfile = len(parfile_in)
  fred%lpstrfiletitle = null
  fred%nmaxfiletitle = null
  fred%lpstrinitialdir = null
  fred%lpstrtitle = loc(dlgtitle)
  fred%flags = null
  fred%nfileoffset = null
  fred%nfileextension = null
  fred%lpstrdefext = null
  fred%lcustdata = null
  fred%lpfnhook = null
  fred%lptemplatename = null
  ret = getopenfilename(fred)
  call comdlger(ierror)
  Par_File_read = .FALSE.
!* check to see if the ok button has been pressed
  if(ret .and. (ierror == 0))then
    call read_parfile
  endif
  return
end subroutine ParFilIn

subroutine read_parfile
  use msflib
  use mtbecom
  use modelcom
  use inputinfo
  use errorcom
  use tser_com
  use parcom
  implicit none
  integer(kind=4)iret, i, ierror, ifile, j
  logical LExist, error, f_error, data_ok, diffsol_error, la_error
  character*3 dbs
  character*72 dtitle, dcomment(2), header, error_msg
  character($maxpath) dDirName(NumTimeSeries)
  character*25 dFileName(NumTimeSeries)
  character*255 tempfile
  integer ipnts, isplit, dTSSetup(NumTimeSeries), dProfilePoints,&
          DataLength(3), idum
  real*8 dMixedLayer(:), dSurfaceTemp(:), dLakeDepth(:), dInflow(:),&
         dOutflow(:), dInflowHeight(:), dOutflowHeight(:), dAirTemp(:),&
         dWindSpeed(:), dAtmosPress(:), dMTBEInput(:), dAtmMTBEConc(:),&
         dLakeArea(:,:), dEpiLoss(:), dHypLoss(:)
  real*8 dTotalRuntime, dOutputTimestep, dinitconc, dMolWeight, dTol,&
         dMaxDepth
  real*8 ssola, ssolb, swa0, swa1, swa2, swb0, swb1, swb2, swd0, ssalinity,&
         swd1, swd2, swd3, sMV, ssolp, sdiffp, drel_hum, srel_hum
  real*8 T, S, D, diff_calc, sol_calc
  allocatable dMixedLayer, dSurfaceTemp, dLakeDepth, dLakeArea, dInflow,&
              dOutflow, dAirTemp, dWindSpeed, dAtmosPress, dMTBEInput,&
              dAtmMTBEConc, dInflowHeight, dOutflowHeight, dEpiLoss,&
              dHypLoss
  external diff_calc, sol_calc
  Data DataLength/12, 52, 365/
  data dbs/'$$$'/
! initialize dDirName and dFileName to null to reset these parameters
  data dDirName/' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '/
  data dFileName/' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '/
! open parameter file
  inquire (file=ParFile_In, exist=LExist)
  if (.not. LExist) then
    msg0 = ' Parameter file does not exist!\nRe-enter filename'C
    msg1 = ' ERROR OPENING FILE 'C
    iret = messageboxqq(msg0, msg1, MB$OK)
    return
  endif
  ipnts = 12
  sdiffp = DiffParam
  ssolp = SolParam
  sMV = MolarVolume
  ssola = sola
  ssolb = solb
  swa0 = wa0
  swa1 = wa1
  swa2 = wa2
  swb0 = wb0
  swb1 = wb1
  swb2 = wb2
  ssalinity = salinity
  swd0 = wd0
  swd1 = wd1
  swd2 = wd2
  swd3 = wd3
  sMV = molarvolume
  sRel_Hum = Rel_Hum
  if (allocated(dSurfaceTemp)) deallocate(dSurfaceTemp)
  if (allocated(dMixedLayer)) deallocate(dMixedLayer)
  if (allocated(dLakeDepth)) deallocate(dLakeDepth)
  if (allocated(dInflow)) deallocate(dInflow)
  if (allocated(dOutflow)) deallocate(dOutflow)
  if (allocated(dInflowHeight)) deallocate(dInflowHeight)
  if (allocated(dOutflowHeight)) deallocate(dOutflowHeight)
  if (allocated(dAirTemp)) deallocate(dAirTemp)
  if (allocated(dWindSpeed)) deallocate(dWindSpeed)
  if (allocated(dAtmosPress)) deallocate(dAtmosPress)
  if (allocated(dMTBEInput)) deallocate(dMTBEInput)
  if (allocated(dAtmMTBEConc)) deallocate(dAtmMTBEConc)
  if (allocated(dLakeArea)) deallocate(dLakeArea)
  if (allocated(dEpiLoss)) deallocate(dEpiLoss)
  if (allocated(dHypLoss)) deallocate(dHypLoss)
  msg1 = ' PARAMETER INPUT ERROR'C
  open (ParFilUnit, file=ParFile_In, status='OLD', action = 'READ')
  read (ParFilUnit, *) dTSSetup(indexTW), header
  allocate(dSurfaceTemp(DataLength(dTSSetup(IndexTW))))
  select case (dTSSetup(indexTW))
    case (1)
      read (ParFilUnit,*) (dSurfaceTemp(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexTW) = tempfile(1:isplit-1)
      dFileName(indexTW) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexMLD), header
  allocate(dMixedLayer(DataLength(dTSSetup(IndexMLD))))
  select case (dTSSetup(indexMLD))
    case (1)
      read (ParFilUnit, *) (dMixedLayer(i), i = 1, 12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexMLD) = tempfile(1:isplit-1)
      dFileName(indexMLD) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexLD), header
  allocate(dLakeDepth(DataLength(dTSSetup(IndexLD))))
  select case (dTSSetup(indexLD))
    case (1)
      read (ParFilUnit,*) (dLakeDepth(i), i=1,12)
      dMaxDepth = 0.0
      do i = 1, 12
        if (dMaxDepth .lt. dLakeDepth(i)) dMaxDepth = dLakeDepth(i)
      end do
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexLD) = tempfile(1:isplit-1)
      dFileName(indexLD) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexIN), header
  allocate(dInflow(DataLength(dTSSetup(IndexIN))))
  select case (dTSSetup(indexIN))
    case (1)
      read (ParFilUnit,*) (dInflow(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexIN) = tempfile(1:isplit-1)
      dFileName(indexIN) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexOUT), header
  allocate(dOutflow(DataLength(dTSSetup(IndexOUT))))
  select case (dTSSetup(indexOUT))
    case (1)
      read (ParFilUnit,*) (dOutflow(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexOUT) = tempfile(1:isplit-1)
      dFileName(indexOUT) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexTA), header
  allocate(dAirTemp(DataLength(dTSSetup(IndexTA))))
  select case (dTSSetup(indexTA))
    case (1)
      read (ParFilUnit,*) (dAirTemp(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexTA) = tempfile(1:isplit-1)
      dFileName(indexTA) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexU), header
  allocate(dWindSpeed(DataLength(dTSSetup(IndexU))))
  select case (dTSSetup(indexU))
    case (1)
      read (ParFilUnit,*) (dWindSpeed(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexU) = tempfile(1:isplit-1)
      dFileName(indexU) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexPA), header
  allocate(dAtmosPress(DataLength(dTSSetup(IndexPA))))
  select case (dTSSetup(indexPA))
    case (1)
      read (ParFilUnit,*) (dAtmosPress(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexPA) = tempfile(1:isplit-1)
      dFileName(indexPA) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexMTBE), header
  allocate(dMTBEInput(DataLength(dTSSetup(IndexMTBE))))
  select case (dTSSetup(indexMTBE))
    case (1)
      read (ParFilUnit,*) (dMTBEInput(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexMTBE) = tempfile(1:isplit-1)
      dFileName(indexMTBE) = tempfile(isplit+3:len_trim(tempfile))
  end select
  read (ParFilUnit, *) dTSSetup(indexAirMTBE), header
  allocate(dAtmMTBEConc(DataLength(dTSSetup(IndexAirMTBE))))
```

```
  select case (dTSSetup(indexAirMTBE))
    case (1)
      read (ParFilUnit,*) (dAtmMTBEConc(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexAirMTBE) = tempfile(1:isplit-1)
      dFileName(indexAirMTBE) = tempfile(isplit+3:len_trim(tempfile))
  end select
! CHECK THE TIME SERIES BEFORE PROCEEDING THROUGH THE PARAMETER FILE
! Have to do LAKEDEPTH first so that dMaxDepth will be set
  ierror = 0
  error = .false.
  f_error = .false.
  data_ok = .true.
  select case (dTSSetup(IndexLD))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dLakeDepth(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexLD), data_ok)
!     data_ok .eq. TRUE if data is good
      if (data_ok) then
        do i = 1, 365
          LD_temp(i) = spl_dummy(i)
        end do
      endif
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
      if (.not. f_error) dMaxDepth = md
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexLD), dTSSetup(IndexLD), &
                      dDirName(IndexLD), dFileName(IndexLD), &
                      error_msg, data_ok)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexLD))
          dLakeDepth(i) = dummy_dat(i)
        end do
        dMaxDepth = md
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in LakeDepth Series:  ', error_msg
    write (ErrWinUnit, '(/a)') ' Processing of the parameter file terminated'
    close (ParFilUnit)
    ErrorWindow = .true.
    return
  endif
  f_error = .false.
  select case (dTSSetup(IndexTW))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dSurfaceTemp(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexTW), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexTW), dTSSetup(IndexTW), &
                      dDirName(IndexTW), dFileName(IndexTW), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexTW))
          dSurfaceTemp(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1)&
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Surface Temperature Series:  ',&
                                 error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexMLD))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dMixedLayer(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexMLD), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexMLD), dTSSetup(IndexMLD), &
                      dDirName(IndexMLD), dFileName(IndexMLD), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexMLD))
          dMixedLayer(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Mixed-Layer Depth Series:  ',&
                                 error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexIN))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dInflow(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexIN), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
```

```
      call read_file (ParRead_IDCVals(IndexIN), dTSSetup(IndexIN), &
                      dDirName(IndexIN), dFileName(IndexIN), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexIN))
          dInflow(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Inflow Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexOUT))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dOutflow(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexOUT), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexOUT), dTSSetup(IndexOUT), &
                      dDirName(IndexOUT), dFileName(IndexOUT), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexOUT))
          dOutflow(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Outflow Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexTA))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dAirTemp(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexTA), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexTA), dTSSetup(IndexTA), &
                      dDirName(IndexTA), dFileName(IndexTA), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexTA))
          dAirTemp(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Air Temperature Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexU))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dWindSpeed(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexU), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexU), dTSSetup(IndexU), &
                      dDirName(IndexU), dFileName(IndexU), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexU))
          dWindSpeed(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Wind Speed Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexPA))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dAtmosPress(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexPA), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexPA), dTSSetup(IndexPA), &
                      dDirName(IndexPA), dFileName(IndexPA), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexPA))
          dAtmosPress(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Atm. Pressure Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexMTBE))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dMTBEInput(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexMTBE), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexMTBE), dTSSetup(IndexMTBE), &
                      dDirName(IndexMTBE), dFileName(IndexMTBE), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexMTBE))
          dMTBEInput(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in VOC Input Series:  ', error_msg
  endif
  f_error = .false.
  select case (dTSSetup(IndexAirMTBE))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dAtmMTBEConc(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexAirMTBE), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexAirMTBE),&
                      dTSSetup(IndexAirMTBE), dDirName(IndexAirMTBE),&
                      dFileName(IndexAirMTBE), error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexAirMTBE))
          dAtmMTBEConc(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Atm VOC Conc. Series:  ', error_msg
  endif
! end checking validity of time series data
  read (ParFilUnit, *)   ! Surface area of lake versus depth profile data
  read (ParFilUnit, *)     ! Number of points in profile
  read (ParFilUnit, *) dProfilePoints
  allocate (dLakeArea(dProfilePoints, 2))
  read (ParFilUnit, *)     ! Depth (m)   :   Lake Area (sq. meters)'
  do i = 1, dProfilePoints
    read (ParFilUnit, *) dLakeArea(i,1), dLakeArea(i,2)
!    write (*,'(i4,2g15.4)') i, dLakeArea(i,1), dLakeArea(i,2)
  end do
  la_error = .false.
  do i = 1, dProfilePoints - 1
    if (dLakeArea(i,1) .le. dLakeArea(i+1,1)) la_error = .true.
    if (dLakeArea(i,2) .lt. dLakeArea(i+1,2)) la_error = .true.
  end do
  if (dLakeArea(dProfilePoints,1) .ne. 0) then
    la_error = .true.
  endif
  if (la_error) then
    error = .true. !set main error flag to true, display local error message
    msg1 = 'Error in Lake Area Profile'C
    msg0 = 'Error in Lake Area versus Depth profile\n&
            Lake Depths must decrease to zero and \n&
            Lake Areas must stay constant or decrease with depth'C
    iret = messageboxqq(msg0,msg1,MB$ICONEXCLAMATION)
  endif
  read (ParFilUnit, *) dTSSetup(indexINHe), header     ! inflow height header
  allocate(dInflowHeight(DataLength(dTSSetup(IndexINHe))))
  select case (dTSSetup(indexINHe))
    case (1)
      read (ParFilUnit,*) (dInflowHeight(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexINHe) = tempfile(1:isplit-1)
      dFileName(indexINHe) = tempfile(isplit+3:len_trim(tempfile))
  end select
  f_error = .false.
  select case (dTSSetup(IndexINHe))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dInflowHeight(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexINHe), data_ok)
      f_error = .not. data_ok
!     f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexINHe), dTSSetup(IndexINHe), &
                      dDirName(IndexINHe), dFileName(IndexINHe), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexINHe))
          dInflowHeight(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Inflow Height Series:  ', error_msg
  endif
  read (ParFilUnit, *) dTSSetup(indexOUTHe), header  ! Outflow height header
  allocate(dOutflowHeight(DataLength(dTSSetup(IndexOUTHe))))
  select case (dTSSetup(indexOUTHe))
    case (1)
      read (ParFilUnit,*) (dOutflowHeight(i), i=1,12)
    case (2, 3)
      read (ParFilUnit, *) tempfile
      isplit = index(tempfile, dbs)
      dDirName(indexOUTHe) = tempfile(1:isplit-1)
      dFileName(indexOUTHe) = tempfile(isplit+3:len_trim(tempfile))
  end select
  f_error = .false.
  select case (dTSSetup(IndexOUTHe))
    case (1)
      do i = 1, 12
        dummy_dat(i) = dOUTflowHeight(i)
      end do
      call spline_dummy(12)  ! spline_dummy located in par_tser.f90
      call par_data_test(ParRead_IDCVals(IndexOUTHe), data_ok)
      f_error = .not. data_ok
!       f_error is TRUE if there was an error (data_ok .eq. FALSE)
    case (2, 3)
      call read_file (ParRead_IDCVals(IndexOUTHe), dTSSetup(IndexOUTHe), &
                      dDirName(IndexOUTHe), dFileName(IndexOUTHe), &
                      error_msg, f_error)
      if (.not. f_error) then
        do i = 1, DataLength(dTSSetup(IndexOUTHe))
          dOUTflowHeight(i) = dummy_dat(i)
        end do
      endif
  end select
  if (f_error) then
    error = .true.
    ierror = ierror + 1
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a,a)') ' Error in Outflow Height Series:  ',&
                                 error_msg
  endif
  read (ParFilUnit, *)
  read (ParFilUnit, *) dInitConc
  read (ParFilUnit, *)
  read (ParFilUnit, *) dMolWeight
  read (ParFilUnit, *)
  read (ParFilUnit, *) dTotalRuntime
  read (ParFilUnit, *)
  read (ParFilUnit, *) dOutputTimestep
  read (ParFilUnit, *)
  read (ParFilUnit, *) dTol
  read (ParFilUnit, *)
  read (ParFilUnit, *) DiffParam
  read (ParFilUnit, *)
  diffsol_error = .false.
  select case (DiffParam)
    case (1)
      read (ParFilUnit, *) MolarVolume
    case (2)
      read (ParFilUnit, *) wd0, wd1, wd2, wd3
```

```
  case default
    error = .true.
    msg0='Diffusivity param. index invalid\nParameter file not read'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
end select
i = 1
do while ((.not. diffsol_error) .and. (i .le. 40))
  t = float(i)
  D = diff_calc(t, DiffParam)
  if (D .le. 0.0) then
    error = .true.
    diffsol_error = .true.
    msg0='Calc. diffus.<=0.\nCheck coefficients\nParameter file not read'C
    IRET = MESSAGEBOXQQ(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  i = i + 1
end do
read (ParFilUnit, *)
read (ParFilUnit, *) SolParam
read (ParFilUnit, *)
select case (SolParam)
  case (1)
    read (ParFilUnit, *) SolA, SolB
  case (2)
    read (ParFilUnit, *) wa0, wa1, wa2, wb0, wb1, wb2, salinity
  case default
    error = .true.
    msg0 = ' Solubility param. index invalid\nParameter file not read'C
    iret = messageboxqq (msg0, msg1,MB$ICONEXCLAMATION .OR. MB$OK)
end select
i = 1
do while ((.not. diffsol_error) .and. (i .le. 40))
  t = float(i)
  S = sol_calc(t, SolParam)
  if (S .le. 0.0) then
    diffsol_error = .true.
    error = .true.
    msg0='Calc. Solub.<=0\nCheck coefficients\nParameter file not read'C
    iret = messageboxqq (msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
  endif
  i = i + 1
end do
ifile = 0
read (parfilunit, *, iostat=ifile)
title = '     '
if (ifile .eq. 0) then
  read (parfilunit, *, iostat=ifile) dtitle
  call dot2space(dtitle)
  if (ifile .eq. 0) then
    read (parfilunit, *, iostat=ifile) dcomment(1)
    call dot2space(dcomment(1))
    if (ifile .eq. 0) then
      read (parfilunit, *, iostat=ifile) dcomment(2)
      call dot2space(dcomment(2))
      if (ifile .eq. 0) then
        read (ParFilUnit, *, iostat=ifile) dTSSetup(indexEpiL), header
        if (ifile .eq. 0) then
          allocate(dEpiLoss(DataLength(dTSSetup(IndexEpiL))))
          select case (dTSSetup(indexEpiL))
            case (1)
```

```
                read (ParFilUnit,*, iostat=ifile) (dEpiLoss(i), i=1,12)
              case (2, 3)
                read (ParFilUnit, *, iostat=ifile) tempfile
                isplit = index(tempfile, dbs)
                dDirName(indexEpiL) = tempfile(1:isplit-1)
                dFileName(indexEpiL) = tempfile(isplit+3:len_trim(tempfile))
              end select
              read (ParFilUnit, *, iostat=ifile) dTSSetup(indexHypL), header
              allocate(dHypLoss(DataLength(dTSSetup(IndexHypL))))
              select case (dTSSetup(indexHypL))
              case (1)
                read (ParFilUnit,*, iostat=ifile) (dHypLoss(i), i=1,12)
              case (2, 3)
                read (ParFilUnit, *, iostat=ifile) tempfile
                isplit = index(tempfile, dbs)
                dDirName(indexHypL) = tempfile(1:isplit-1)
                dFileName(indexHypL) = tempfile(isplit+3:len_trim(tempfile))
            end select
          endif
        endif
      endif
    endif
  endif
! End of degradation rate input.
  if (ifile .ne. 0) then
    dTSSetup(indexEpiL) = 1
    dTSSetup(indexHypL) = 1
    allocate(dEpiLoss(DataLength(dTSSetup(IndexEpiL))))
    allocate(dHypLoss(DataLength(dTSSetup(IndexHypL))))
    do i = 1, 12
      dEpiLoss(i) = 0.0
      dHypLoss(i) = 0.0
    end do
    dRel_Hum = 0.7
  else
!   read in relative humidity
    read (ParFilUnit,*, iostat=ifile)
    read (ParFilUnit,*, iostat=ifile) dRel_Hum
    if (ifile .eq. 0) then
      dRel_Hum = 0.01*dRel_Hum   ! convert back from % to fraction
    else
      dRel_Hum = 0.7
    endif
  endif
  close (parfilunit)   ! close the parameter file, end of data input.
  ierror = 0
  if (dLakeArea(1,1) .lt. dMaxDepth) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in surface area profile.  Maximum &
                               profile depth must be >= maximum lake depth '
  endif
  if (dTotalRuntime .le. 0.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in TOTAL RUNTIME.  Runtime must be >0'
  endif
  if (dOutputTimestep .le. 0.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in OUTPUT TIMESTEP.  Timestep <= 0'
  endif
  if ((TotalRuntime*365.0)/OutputTimestep .gt. 10000.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit, '(a,a)') &
        'TOTAL TIME/TIME STEP exceeds 10,000 data points.  ',&
        'Modify TOTAL TIME and TIME STEP accordingly'
  endif
  if (dMolWeight .le. 0.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in MOLECULAR WEIGHT.  Weight must be >0'
  endif
  if (dinitconc .lt. 0.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in INITIAL VOC CONC.  Concentration<0'
  endif
  if (dTol .le. 0.0) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in TOLERANCE.  TOLERANCE must be >0'
  endif
  if ((dRel_Hum .le. 0.0) .or. (dRel_Hum .gt. 100.0)) then
    ierror = ierror + 1
    error = .true.
    if (ierror .eq. 1) &
      open (ErrWinUnit, file='USER', title='PARAMETER FILE INPUT ERRORS')
    write (ErrWinUnit,'(a)') ' Error in RELATIVE HUMIDITY.  100>=R.H.>=0'
  endif
  if (.not. error) then
!   Surface Temperature
    deallocate(SurfaceTemp)
    if (dTSSetup(indexTW) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexTW) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexTW) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexTW) = idum
    allocate (SurfaceTemp(idum))
    SurfaceTemp = dSurfaceTemp
    TSSetup(IndexTW) = dTSSetup(indexTW)
    FileName(IndexTW) = dFileName(IndexTW)
    DataDir(IndexTW) = dDirName(IndexTW)
!    Mixed Layer Depth
    deallocate(MixedLayer)
    if (dTSSetup(indexMLD) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexMLD) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexMLD) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexMLD) = idum
    allocate (MixedLayer(idum))
    MixedLayer = dMixedLayer
    TSSetup(IndexMLD) = dTSSetup(indexMLD)
    FileName(IndexMLD) = dFileName(IndexMLD)
    DataDir(IndexMLD) = dDirName(IndexMLD)
!    Lake Depth
    deallocate(LakeDepth)
    if (dTSSetup(indexLD) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexLD) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexLD) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexLD) = idum
    allocate (LakeDepth(idum))
    LakeDepth = dLakeDepth
    TSSetup(IndexLD) = dTSSetup(indexLD)
    FileName(IndexLD) = dFileName(IndexLD)
    DataDir(IndexLD) = dDirName(IndexLD)
!    Inflow Volume
    deallocate(Inflow)
    if (dTSSetup(indexIN) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexIN) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexIN) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexIN) = idum
    allocate (Inflow(idum))
    Inflow = dInflow
    TSSetup(IndexIN) = dTSSetup(indexIN)
    FileName(IndexIN) = dFileName(IndexIN)
    DataDir(IndexIN) = dDirName(IndexIN)
!    Inflow Height
    deallocate(InflowHeight)
    if (dTSSetup(indexINHe) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexINHe) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexINHe) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexINHe) = idum
    allocate (InflowHeight(idum))
    InflowHeight = dInflowHeight
    TSSetup(IndexINHe) = dTSSetup(indexINHe)
    FileName(IndexINHe) = dFileName(IndexINHe)
    DataDir(IndexINHe) = dDirName(IndexINHe)
!    Outflow Volume
    deallocate(Outflow)
    if (dTSSetup(indexOUT) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexOUT) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexOUT) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexOUT) = idum
    allocate (Outflow(idum))
    Outflow = dOutflow
    TSSetup(IndexOUT) = dTSSetup(indexOUT)
    FileName(IndexOUT) = dFileName(IndexOUT)
    DataDir(IndexOUT) = dDirName(IndexOUT)
!    Outflow Height
    deallocate(OutflowHeight)
    if (dTSSetup(indexOUTHe) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexOUTHe) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexOUTHe) .eq. 3) then
      idum = 365
    endif
    TSDatlen(IndexOUTHe) = idum
    allocate (OutflowHeight(idum))
    OutflowHeight = dOutflowHeight
    TSSetup(IndexOUT) = dTSSetup(indexOUT)
    FileName(IndexOUT) = dFileName(IndexOUT)
    DataDir(IndexOUT) = dDirName(IndexOUT)
!    Air Temperature
    deallocate(AirTemp)
    if (dTSSetup(indexTA) .eq. 1) then
      idum = 12
    elseif (dTSSetup(indexTA) .eq. 2) then
      idum = 52
    elseif (dTSSetup(indexTA) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexTA) = idum
    allocate (AirTemp(idum))
    AirTemp = dAirTemp
    TSSetup(IndexTA) = dTSSetup(indexTA)
    FileName(IndexTA) = dFileName(IndexTA)
    DataDir(IndexTA) = dDirName(IndexTA)
!    Wind Speed
    deallocate(WindSpeed)
    if (dTSSetup(IndexU) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexU) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexU) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexU) = idum
    allocate (WindSpeed(idum))
    WindSpeed = dWindSpeed
    TSSetup(IndexU) = dTSSetup(IndexU)
    FileName(IndexU) = dFileName(IndexU)
    DataDir(IndexU) = dDirName(IndexU)
!    Atmospheric Pressure
    deallocate(AtmosPress)
    if (dTSSetup(IndexPA) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexPA) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexPA) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexPA) = idum
    allocate (AtmosPress(idum))
    AtmosPress = dAtmosPress
    TSSetup(IndexPA) = dTSSetup(IndexPA)
    FileName(IndexPA) = dFileName(IndexPA)
    DataDir(IndexPA) = dDirName(IndexPA)
!    VOC input to hypolimnion (Assumed to be MTBE in original model)
    deallocate(MTBEInput)
    if (dTSSetup(IndexMTBE) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexMTBE) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexMTBE) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexMTBE) = idum
    allocate (MTBEInput(idum))
    MTBEInput = dMTBEInput
    TSSetup(IndexMTBE) = dTSSetup(IndexMTBE)
    FileName(IndexMTBE) = dFileName(IndexMTBE)
    DataDir(IndexMTBE) = dDirName(IndexMTBE)
!    Atmospheric VOC Concentration
    deallocate(AtmMTBEConc)
    if (dTSSetup(IndexAirMTBE) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexAirMTBE) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexAirMTBE) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexAirMTBE) = idum
    allocate (AtmMTBEConc(idum))
    AtmMTBEConc = dAtmMTBEConc
    TSSetup(IndexAirMTBE) = dTSSetup(IndexAirMTBE)
    FileName(IndexAirMTBE) = dFileName(IndexAirMTBE)
    DataDir(IndexAirMTBE) = dDirName(IndexAirMTBE)
!    Biochemical VOC Loss Rate in epilimnion
    deallocate(EpiLossRate)
    if (dTSSetup(IndexEpiL) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexEpiL) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexEpiL) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexEpiL) = idum
    allocate (EpiLossRate(idum))
    EpiLossRate = dEpiLoss
    TSSetup(IndexEpiL) = dTSSetup(IndexEpiL)
    FileName(IndexEpiL) = dFileName(IndexEpiL)
    DataDir(IndexEpiL) = dDirName(IndexPA)
!    Biochemical VOC Loss Rate in hypolimnion
    deallocate(HypLossRate)
    if (dTSSetup(IndexHypL) .eq. 1) then
      idum = 12
    elseif (dTSSetup(IndexHypL) .eq. 2) then
      idum = 52
    elseif (dTSSetup(IndexHypL) .eq. 3) then
    idum = 365
    endif
    TSDatlen(IndexHypL) = idum
    allocate (HypLossRate(idum))
    HypLossRate = dHypLoss
    TSSetup(IndexHypL) = dTSSetup(IndexHypL)
    FileName(IndexHypL) = dFileName(IndexHypL)
    DataDir(IndexHypL) = dDirName(IndexHypL)
!    Finished copying dummy time series to those used by model
!    Now must spline the data to a grid of 365 days/year
    do i = 1, NumTimeSeries
      call spline_data(i)
      TSError(i) = .false.
      FileLoaded(i) = .false.
      File_Status(i) = 'Unknown'C
      File_Status2(i) = 'Not Loaded'C
      if ((TSSetup(i) .eq. 2) .or. (TSSetup(i) .eq. 3)) then
        FileLoaded(i) = .true.
        File_Status(i) = 'File Loaded'C
        File_Status2(i) = 'Data OK'C
      endif
    end do
    MaxDepth = dMaxDepth
    ProfilePoints = dProfilePoints
    if (allocated(LakeArea)) deallocate(LakeArea)
    allocate(LakeArea(ProfilePoints,2))
    do i = 1, ProfilePoints
      do j = 1, 2
        LakeArea(i,j) = dLakeArea(i,j)
      end do
    end do
    call lake_volume_calc
    TotalRuntime = dTotalRuntime
    OutputTimestep = dOutputTimestep
    MolWeight = dMolWeight
    Initial_MTBEConc = dinitconc
    Rel_Hum = dRel_Hum
    Tolerance = dTol
    title = dtitle
    comment(1) = dcomment(1)
    comment(2) = dcomment(2)
    Par_File_Read = .TRUE.
    msg1 = 'Operation Status Report'C
    msg0 = 'Parameter file read successfully'C
    ParFile_In(1:len_trim(ParFile_in)), '  read'
    iret = messageboxqq (msg0, msg1, MB$OK)
    return
  else
    msg0 = &
'Errors in parameter file\nFILE NOT READ!!!!\nPlease check parameter file'C
    iret = messageboxqq (msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    errorwindow = .true.
```

```
!   NEED TO RESET THE DIFF AND SOLUB COEFFICIENTS BEFORE RETURNING
    MolarVolume = sMV
    sola = ssola
    solb = ssolb
    wa0 = swa0
    wa1 = swa1
    wa2 = swa2
    wb0 = swb0
    wb1 = swb1
    wb2 = swb2
    salinity = ssalinity
    wd0 = swd0
    wd1 = swd1
    wd2 = swd2
    wd3 = swd3
    DiffParam = sdiffp
    SolParam = ssolp
    rel_hum = sRel_Hum
    return
  endif
end subroutine read_parfile


subroutine dot2space(tline)
  implicit none
  character*72 tline
  integer length, i
  length = len_trim(tline)
  do i = 1, length
    if (tline(i:i) .eq. '~') tline(i:i) = ' '
  end do
  return
end subroutine dot2space


! id is the IDC value of the time series data file (e.g. IDC_MixedLayer)
! Dirname is the temp directory name
! Filename is the temp filename
! Err_msg is the error message returned to the calling subroutine
! File_error is the logical value returned, .true. if there was an error
! index tells whether reading 52 or 365 data points
subroutine Read_File(id, index, Dname, Fname, err_msg, File_error)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use tser_com
  use parcom
  implicit none
  logical ret, file_exist, File_error, data_ok
  integer i, j, id, lengthdir, lengthfile, ts_ident, file_iostat, idum, &
          ipnts, index
  character($MAXPATH) Dname, dir_file, dir_save
  character*72 err_msg
  character*12 Fname
    File_error = .false.
    dir_save = FILE$CURDRIVE
    lengthdir = getdrivedirqq(dir_save)
    if (.not. changedirqq(Dname)) then
      err_msg = 'Directory does not exist'
      File_error = .true.
      return
    endif
    ret = changedirqq(dir_save)
    lengthdir = len_trim(Dname)
    lengthfile = len_trim(Fname)
    if ((Dname(lengthdir:lengthdir) .ne. '\') .and. (Fname(1:1) .ne. '\')) then
         Dname(lengthdir+1:lengthdir+1) = '\'
       lengthdir = lengthdir + 1
    endif
  ! if both have \'s, reset lengthdir so that trailing slash gets overwritten!
    if ((Dname(lengthdir:lengthdir) .eq. '\') .and. (Fname(1:1) .eq. '\')) &
         lengthdir = lengthdir - 1
    dir_file = Dname(1:lengthdir)//Fname(1:lengthfile)
    inquire (FILE=dir_file, EXIST=file_exist)
    if (.not. file_exist) then
      err_msg = 'Data file not found!'
      File_error = .true.
      return
    endif
  ! SELECT_TS_ID is located in TSER_COM.F90, it sets the value of index
    call select_ts_id(id, ts_ident)
    File_Status(ts_ident) = 'File Found'
  ! open up the data file and read the first line of header information
    open (11, file=dir_file, iostat=file_iostat)
    file_iostat = 0
    read (11, *, iostat=file_iostat)
    i = 0
    do while (file_iostat .ge. 0)
      i = i + 1
      read (11, *, iostat=file_iostat) idum, dummy_dat(i)
    end do
    close (11)
    i = i - 1
    data_ok = .false.
    select case (index)
      case (2)
        if (i .ne. 52) then
          if (i .lt. 52) err_msg = 'Not enough data points in data file'
          if (i .gt. 52) err_msg = 'Too many data points in file'
          File_error = .true.
          return
        endif
        ipnts = i
        file_error = .false.
        call spline_dummy(52)  ! spline_dummy located in par_tser.f90
        call par_data_test(id, data_ok)
      case (3)
        if (i .ne. 365) then
          if (i .lt. 365) err_msg = 'Not enough data points in data file'
          if (i .gt. 365) err_msg = 'Too many data points in file'
          File_error = .true.
          return
        endif
        ipnts = i
        file_Error = .false.
        call spline_dummy(365)  ! spline_dummy located in par_tser.f90
        call par_data_test(id, data_ok)
    end select
    if (data_ok) then
      if (id .eq. IDC_LakeDepth) then
```

```
        do i = 1, 365
          LD_temp(i) = spl_dummy(i)
        end do
      endif
      file_error = .false.
    else
      file_error = .true.
      err_msg = 'Inconsistent numerical value in data file'
    endif
    return
end subroutine Read_File


subroutine par_data_test (id, data_ok)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  use parcom
  use tser_com
  implicit none
  logical data_ok
  integer id, i
  data_ok = .true.
    select case (id)
      case (IDC_LakeDepth)
        do i = 1, 365
          if (spl_dummy(i) .le. 0.0) data_ok = .false.
        end do
      case (IDC_SurfaceTemp)
        do i = 1, 365
          if ((spl_dummy(i) .lt. -15.0) .or. (spl_dummy(i) .gt. 100.0)) &
            data_ok = .false.
        end do
      case (IDC_MixedLayer)
        do i = 1, 365
          if ((spl_dummy(i) .lt. 0.0) .or. (spl_dummy(i) .gt. LD_temp(i))) &
            data_ok = .false.
        end do
      case (IDC_Inflow)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_Outflow)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_InflowHeight)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_OutflowHeight)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_AirTemp)
        do i = 1, 365
          if ((spl_dummy(i) .lt. -100.0) .or. (spl_dummy(i) .gt. 100.0)) &
            data_ok = .false.
        end do
      case (IDC_WindSpeed)
        do i = 1, 365
          if ((spl_dummy(i) .lt. 0.0) .or. (spl_dummy(i) .gt. 100.0)) &
            data_ok = .false.
        end do
      case (IDC_AtmPressure)
        do i = 1, 365
          if ((spl_dummy(i) .le. 0.0) .or. (spl_dummy(i) .gt. 2.0)) &
            data_ok = .false.
        end do
      case (IDC_MTBEInputSeries)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_AtmMTBEConc)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_EpiLossRate)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
      case (IDC_HypLossRate)
        do i = 1, 365
          if (spl_dummy(i) .lt. 0.0) data_ok = .false.
        end do
    end select
  return
end subroutine par_data_test


subroutine ParFilOut(checked)
! Windows API common interface filename input subroutine for param file
! Note that for correct operation with threads need to check existence
! of files using INQUIRE rather than OPEN.  I know not why.
  use msflib
  use inputinfo
  use msfwinty
  use msfwin
  use mtbecom
  use errorcom
  use tser_com
  implicit none
  type (t_openfilename)fred
  logical(kind=4)ret
  integer(kind=4)ierror, i
  character(len=26)filter(7)
  character(len=60)dlgtitle
  logical(kind=4)checked, ts_error
  external write_parfile
  call unusedqq(checked)
  if (MenuActive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'C
    ierror = messageboxqq(msg0, msg1,MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
  i = 1
  do while ((.not. ts_error) .and. (i .le. NumTimeSeries))
    if (TSError(i)) ts_error = .true.
    i = i + 1
  enddo
  if (ts_error) then
    msg0 = 'Correct errors in time series before writing parameter file\n&
            Check time Series Setup Menu'C
    msg1 = 'Time Series Setup Error'C
    ierror = messageboxqq(msg0, msg1,MB$ICONEXCLAMATION .OR. MB$OK)
    return
  endif
! SET UP FILE SEARCH FILTERS
  filter(1) = 'Parameter File (*.PAR)  'C
  filter(2) = '*.par                   'C
  filter(3) = 'All Files (*.*)         'C
  filter(4) = '*.*                     'C
  filter(5) = ''C
  filter(6) = ''C
  filter(7) = ''C
! DIALOG TITLE
  dlgtitle = 'Write Parameter File'C
! SET UP STRUCTURE USED BY COMMON DIALOGS
! SEE WIN32 API HELP FOR EXPLANATION
  fred%lstructsize = (bit_size(fred%lstructsize) +        &
                      bit_size(fred%hwndowner) +        &
                      bit_size(fred%hinstance) +        &
                      bit_size(fred%lpstrfilter) +        &
                      bit_size(fred%lpstrcustomfilter) +        &
                      bit_size(fred%nmaxcustfilter) +        &
                      bit_size(fred%nfilterindex) +        &
                      bit_size(fred%lpstrfile) +        &
                      bit_size(fred%nmaxfile) +        &
                      bit_size(fred%lpstrfiletitle) +        &
                      bit_size(fred%nmaxfiletitle) +        &
                      bit_size(fred%lpstrinitialdir) +        &
                      bit_size(fred%lpstrtitle) +        &
                      bit_size(fred%flags) +        &
                      bit_size(fred%nfileoffset) +        &
                      bit_size(fred%nfileextension) +        &
                      bit_size(fred%lpstrdefext) +        &
                      bit_size(fred%lcustdata) +        &
                      bit_size(fred%lpfnhook) +        &
                      bit_size(fred%lptemplatename))/8
  fred%hwndowner = null
  fred%hinstance = null
  fred%lpstrfilter = loc(filter(1))
  fred%lpstrcustomfilter = null
  fred%nmaxcustfilter = null
  fred%nfilterindex = 1
  fred%lpstrfile = loc(parfile_out)
  fred%nmaxfile = len(parfile_out)
  fred%lpstrfiletitle = null
  fred%nmaxfiletitle = null
  fred%lpstrinitialdir = null
  fred%lpstrtitle = loc(dlgtitle)
  fred%flags = null
  fred%nfileoffset = null
  fred%nfileextension = null
  fred%lpstrdefext = null
  fred%lcustdata = null
  fred%lpfnhook = null
  fred%lptemplatename = null
```

```
! create dialog
  ret = getsavefilename(fred)
! check for error
  call comdlger(ierror)
! check to see if the ok button has been pressed
  if(ret .and. (ierror == 0))then
    call write_parfile
  endif
  return
end subroutine ParFilOut


subroutine write_parfile
  use msflib
  use mtbecom
  use modelcom
  use inputinfo
  use errorcom
  use tser_com
  implicit none
  integer(kind=4)iret, ierr, i
  logical LExist
  integer*2 delval
  character*10 weekly
  character*9 daily
  character*3 dbs
  weekly = ' 2 Weekly '
  daily = ' 3 Daily '
  dbs = '$$$'
! open data output file
  inquire (file=ParFile_Out, exist=LExist)
  if (LExist) then
    msg0 = ' Parameter file already exists!\nOverwrite and continue?'C
    msg1 = ' FILE CREATION WARNING'C
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$YESNO)
    if (iret == MB$IDYES) then
      delval = delfilesqq(ParFile_Out)
      ierr = 0
      open(ParFilUnit, file=ParFile_Out, status='NEW', iostat=ierr)
      if (ierr .ne. 0) write (*,'(a,i3)') ' IERR =', ierr
    endif
  else
    open(ParFilUnit, file=ParFile_Out, status='NEW', iostat=ierr)
  endif
  select case (TSsetup(indexTW))
    case (1)
      write (ParFilUnit, '(a)') &
              ' 1 Monthly Averaged Mixed Layer Temperatures in deg-C'
      write (ParFilUnit, '(12f8.2)') (SurfaceTemp(i), i=1,12)
    case (2, 3)
      if (TSSetup(indexTW) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexTW) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') &
              'Averaged Mixed Layer Temperatures Data Filename'
      write (ParFilUnit, '(a,a,a)') &
              DataDir(indexTW)(1:len_trim(DataDir(indexTW))), dbs,&
              FileName(indexTW)
  end select
  select case (TSsetup(indexMLD))
    case (1)
```

```
      write (ParFilUnit, '(a)') &
             ' 1 Monthly Averaged Mixed Layer Depths in meters'
      write (ParFilUnit, '(12f8.2)') (MixedLayer(i), i=1,12)
    case (2, 3)
      if (TSSetup(indexMLD) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexMLD) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') &
             'Averaged Mixed Layer Depth Data Filename'
      write (ParFilUnit, '(a,a,a)') DataDir(indexMLD)(1:len_trim(Data-
Dir(indexMLD))), dbs, FileName(indexMLD)
  end select
  select case (TSsetup(indexLD))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Monthly Averaged Lake Depths in meters'
      write (ParFilUnit, '(12f8.2)') (LakeDepth(i), i=1,12)
    case (2, 3)
      if (TSSetup(indexLD) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexLD) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') &
             'Averaged Lake Depth Data Filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexLD)(1:len_trim(DataDir(indexLD))), dbs,&
             FileName(indexLD)
  end select
  select case (TSsetup(indexIN))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Monthly averaged Lake Inflow in m^3/day'
      write (ParFilUnit, '(12f10.1)') (Inflow(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexIN) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexIN) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') 'Averaged Lake Inflow data filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexIN)(1:len_trim(DataDir(indexIN))), dbs,&
             FileName(indexIN)
  end select
  select case (TSsetup(indexOUT))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Monthly averaged Lake Outflow in m^3/day'
      write (ParFilUnit, '(12f10.1)') (Outflow(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexOUT) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexOUT) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') 'Averaged Lake Outflow data filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexOUT)(1:len_trim(DataDir(indexOUT))), dbs,&
             FileName(indexOUT)
  end select
  select case (TSsetup(indexTA))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Monthly Averaged Air Temperatures in deg-C'
      write (ParFilUnit, '(12f8.2)') (AirTemp(i), i=1,12)
    case (2, 3)
      if (TSSetup(indexTA) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexTA) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') &
```

```
           'Averaged Air Temperatures Data Filename'
    write (ParFilUnit, '(a,a,a)') &
           DataDir(indexTA)(1:len_trim(DataDir(indexTA))), dbs,&
           FileName(indexTA)
end select
select case (TSsetup(indexU))
  case (1)
    write (ParFilUnit, '(a)') &
           ' 1 Monthly Averaged Wind Speeds in meters per second'
    write (ParFilUnit, '(12f9.3)') (WindSpeed(i), i=1,12)
  case (2, 3)
    if (TSSetup(indexU) .eq. 2) write (ParFilUnit, '(a\)') weekly
    if (TSSetup(indexU) .eq. 3) write (ParFilUnit, '(a\)') daily
    write (ParFilUnit, '(a)') 'Averaged Wind Speeds Data Filename'
    write (ParFilUnit, '(a,a,a)') &
           DataDir(indexU)(1:len_trim(DataDir(indexU))), dbs,&
           FileName(indexU)
end select
select case (TSsetup(indexPA))
  case (1)
    write (ParFilUnit, '(a)') &
           ' 1 Monthly averaged barometric pressure in atmospheres'
    write (ParFilUnit, '(12f10.4)') AtmosPress
  case (2, 3)
    if (TSSetup(indexPA) .eq. 2) write (ParFilUnit, '(a\)') weekly
    if (TSSetup(indexPA) .eq. 3) write (ParFilUnit, '(a\)') daily
    write (ParFilUnit, '(a)') &
           'Averaged Barometric Pressure Data Filename'
    write (ParFilUnit, '(a,a,a)') &
           DataDir(indexPA)(1:len_trim(DataDir(indexPA))), dbs,&
           FileName(indexPA)
end select
select case (TSsetup(indexMTBE))
  case (1)
    write (ParFilUnit, '(a)') &
           ' 1 Monthly Averaged VOC Inputs in kg/month'
    write (ParFilUnit, '(12f9.3)') (MTBEInput(i), i=1,12)
  case (2, 3)
    if (TSSetup(indexMTBE) .eq. 2) write (ParFilUnit, '(a\)') weekly
    if (TSSetup(indexMTBE) .eq. 3) write (ParFilUnit, '(a\)') daily
    write (ParFilUnit, '(a)') 'Averaged VOC Input Data Filename'
    write (ParFilUnit, '(a,a,a)') &
           DataDir(indexMTBE)(1:len_trim(DataDir(indexMTBE))), dbs,&
           FileName(indexMTBE)
end select
select case (TSsetup(indexAirMTBE))
  case (1)
    write (ParFilUnit, '(a)') &
           ' 1 Monthly Averaged Atmospheric VOC Concentrations in ppbv'
    write (ParFilUnit, '(12e14.4)') (AtmMTBEConc(i), i=1,12)
  case (2, 3)
    if (TSSetup(indexAirMTBE).eq.2) write (ParFilUnit, '(a\)') weekly
    if (TSSetup(indexAirMTBE).eq.3) write (ParFilUnit, '(a\)') daily
    write (ParFilUnit, '(a)') 'Averaged Atm. VOC Conc. Data Filename'
    write (ParFilUnit, '(a,a,a)') &
           DataDir(indexAirMTBE)(1:len_trim(Data-Dir(indexAirMTBE))), &
           dbs, FileName(indexAirMTBE)
end select
write (ParFilUnit,'(a)') ' Surf. area of lake vs. depth profile data'
write (ParFilUnit, '(a)') ' Number of points in profile'
write (ParFilUnit, '(i4)') ProfilePoints
```

```
  write (ParFilUnit, '(a)') ' Depth (m)    :   Lake Area (sq. meters)'
  do i = 1, ProfilePoints
    write (ParFilUnit, '(f10.2,3x,f16.2)') LakeArea(i,1), LakeArea(i,2)
  end do
  select case (TSsetup(indexINHe))
    case (1)
      write(ParFilUnit,'(a)') ' 1 Monthly aver. Lake Inflow height (m)'
      write (ParFilUnit, '(12f10.1)') (InflowHeight(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexINHe) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexINHe) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit,'(a)') 'Aver. Lake Inflow Height data filename'
      write (ParFilUnit, &
             '(a,a,a)') DataDir(indexIN)(1:len_trim(DataDir(index-INHe))), &
             dbs, FileName(indexINHe)
  end select
  select case (TSsetup(indexOUTHe))
    case (1)
      write(ParFilUnit,'(a)')' 1 Monthly aver. Lake Outflow Height (m)'
      write (ParFilUnit, '(12f10.1)') (OutflowHeight(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexOUTHe) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexOUTHe) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit,'(a)') 'Aver. Lake Outflow Height data filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexOUTHe)(1:len_trim(DataDir(index-OUTHe))), &
             dbs, FileName(indexOUTHe)
  end select
  write (ParFilUnit,'(a)') ' Init. VOC epilim. conc. in micrograms/L'
  write (ParFilUnit, '(f8.3)') Initial_MTBEConc
  write (ParFilUnit, '(a)') ' VOC molecular weight in g/mole'
  write (ParFilUnit, '(f9.3)') MolWeight
  write (ParFilUnit, '(a)') ' Total model runtime in years'
  write (ParFilUnit, '(f12.3)') TotalRuntime
  write (ParFilUnit, '(a)') ' Time step ASCII data file points (days)'
  write (ParFilUnit, '(f12.4)') OutputTimestep
  write (ParFilUnit, '(a)') ' Tolerance for Runge-Kutta DEQ integrator'
  write (ParFilUnit, '(e15.4)') Tolerance
  write (ParFilUnit, '(a)') &
    ' Diffusivity characterization (1 for Wilke-Chang, 2 for Wanninkhof)'
  write (ParFilUnit, '(i5)') DiffParam
  select case (DiffParam)
    case (1)
      write (ParFilUnit, '(a)') &
             ' Molar volume in ml/mol at boiling point for Wilke Chang'
      write (ParFilUnit, '(f15.5)') MolarVolume
    case (2)
      write (ParFilUnit, '(a)') &
             ' Coefficients for polynomial characterization of Schmidt &
number (Wanninkhof (1992))'
      write (ParFilUnit, '(4(4x,e15.6))') wd0, wd1, wd2, wd3
  end select
  write (ParFilUnit, '(a)') &
         ' Solubility characterization (1 for exp(-(A-B/T)), 2 for Wanninkhof)'
  write (ParFilUnit, '(i5)') SolParam
  select case (SolParam)
    case (1)
      write (ParFilUnit, '(a)') &
             ' A and B coefficients to give solubility in atm-m^3/mol'
      write (ParFilUnit, '(2(4x,e15.6))') SolA, SolB
    case (2)
      write (ParFilUnit, '(a)') &
```

```
        ' Coefficients for polynomial characterization of Ostwald solubility &
        (Wanninkhof (1992))'
      write (ParFilUnit, '(6(4x,e15.6),f10.3)') wa0, wa1, wa2, wb0,&
             wb1, wb2, salinity
  end select
  write (ParFilUnit, '(a)') ' Title for run and two lines of comments,&
         comments not used'
  call space2dot(title)
  call space2dot(comment(1))
  call space2dot(Comment(2))
  write (ParFilUnit, '(a72)') title
  write (ParFilUnit, '(a72)') comment(1)
  write (ParFilUnit, '(a72)') comment(2)
  call dot2space(title)
  call dot2space(comment(1))
  call dot2space(Comment(2))
  select case (TSsetup(indexEpiL))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Biochemical degradation rates for epilimnion 1/days'
      write (ParFilUnit, '(12f10.1)') (EpiLossRate(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexEpiL) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexEpiL) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') &
             'Biochemical degradation rates for epilimnion filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexEpiL)(1:len_trim(DataDir(indexEpiL))), dbs, &
             FileName(indexEpiL)
  end select
  select case (TSsetup(indexHypL))
    case (1)
      write (ParFilUnit, '(a)') &
             ' 1 Biochemical degradation rates for hypolimnion 1/days'
      write (ParFilUnit, '(12f10.1)') (HypLossRate(i), i = 1, 12)
    case (2, 3)
      if (TSSetup(indexHypL) .eq. 2) write (ParFilUnit, '(a\)') weekly
      if (TSSetup(indexHypL) .eq. 3) write (ParFilUnit, '(a\)') daily
      write (ParFilUnit, '(a)') 'Averaged Lake Inflow data filename'
      write (ParFilUnit, '(a,a,a)') &
             DataDir(indexHypL)(1:len_trim(DataDir(indexHypL))), dbs, &
             FileName(indexHypL)
  end select
  write (ParFilUnit, '(a)') ' Relative Humidity (%)'
  write (ParFilUnit, '(f8.3)') 100.0*rel_hum
  close (parfilunit)
  Par_File_Sav = .TRUE.
  return
end subroutine write_parfile


subroutine space2dot(tline)
  implicit none
  character*72 tline
  integer length, i
  length = len_trim(tline)
  do i = 1, length
    if (tline(i:i) .eq. ' ') tline(i:i) = '~'
  end do
  return
```

```
end subroutine space2dot


subroutine datfilout(checked)
! saves the output data file from the model
  use msflib
  use inputinfo
  use msfwinty
  use msfwin
  use mtbecom
  implicit none
  type (t_openfilename) fred
  logical (kind=4) ret
  integer (kind=4) ierror
  character (len=26) filter(7)
  character (len=60) dlgtitle
  logical (kind=4) checked
!  external write_datfile
  call unusedqq(checked)
  if (menuactive) then
    msg0 = 'Please close open set-up menu\nbefore opening new window'C
    msg1 = 'Window Error'c
    ierror = messageboxqq(msg0, msg1, mb$iconexclamation .or. mb$ok)
    return
  endif
!* set up file search filters
  filter(1) = 'VOC data files (*.dat)  'C
  filter(2) = '*.dat                    'C
  filter(3) = 'all files (*.*)          'C
  filter(4) = '*.*                      'C
  filter(5) = ''C
  filter(6) = ''C
  filter(7) = ''C
!* dialog title
  dlgtitle = 'save model output to file'C
!* set up structure used by common dialogs - see win32 api help for explanation
  fred%lstructsize = (bit_size(fred%lstructsize) +        &
                      bit_size(fred%hwndowner) +        &
                      bit_size(fred%hinstance) +        &
                      bit_size(fred%lpstrfilter) +        &
                      bit_size(fred%lpstrcustomfilter) +        &
                      bit_size(fred%nmaxcustfilter) +        &
                      bit_size(fred%nfilterindex) +        &
                      bit_size(fred%lpstrfile) +        &
                      bit_size(fred%nmaxfile) +        &
                      bit_size(fred%lpstrfiletitle) +        &
                      bit_size(fred%nmaxfiletitle) +        &
                      bit_size(fred%lpstrinitialdir) +        &
                      bit_size(fred%lpstrtitle) +        &
                      bit_size(fred%flags) +        &
                      bit_size(fred%nfileoffset) +        &
                      bit_size(fred%nfileextension) +        &
                      bit_size(fred%lpstrdefext) +        &
                      bit_size(fred%lcustdata) +        &
                      bit_size(fred%lpfnhook) +        &
                      bit_size(fred%lptemplatename))/8
  fred%hwndowner = null
  fred%hinstance = null
  fred%lpstrfilter = loc(filter(1))
  fred%lpstrcustomfilter = null
  fred%nmaxcustfilter = null
  fred%nfilterindex = 1
  fred%lpstrfile = loc(datfile_out)
  fred%nmaxfile = len(datfile_out)
  fred%lpstrfiletitle = null
  fred%nmaxfiletitle = null
  fred%lpstrinitialdir = null
  fred%lpstrtitle = loc(dlgtitle)
  fred%flags = null
  fred%nfileoffset = null
  fred%nfileextension = null
  fred%lpstrdefext = null
  fred%lcustdata = null
  fred%lpfnhook = null
  fred%lptemplatename = null
!* create dialog
  ret = getsavefilename(fred)
!* check for error
  call comdlger(ierror)
!* check to see if the ok button has been pressed
  mtbe_file_sav = .false.
  if (ret .and. (ierror == 0))then
    call sav_mtbe
    mtbe_file_sav = .true.
  endif
  return
end subroutine datfilout


subroutine comdlger(iret)
  use msflib
  use msfwinty
  use msfwin
  implicit none
  character*30 msg1
  character(len=210) msg3
  integer(kind=4)iret
  iret = commdlgextendederror()
  msg1 = 'Tile open dialog failure'c
  select case(iret)
    case (cderr_findresfailure)
      msg3 = 'The common dialog box procedure failed to find a specified resource.'C
    case (cderr_initialization)
      msg3 = 'The common dialog box procedure failed during initialization. this &
              error often occurs when insufficient memory is available.'C
    case (cderr_lockresfailure)
      msg3 = 'The common dialog box procedure failed to load a specified resource.'C
    case (cderr_loadresfailure)
      msg3 = 'The common dialog box procedure failed to load a specified resource.'C
    case (cderr_loadstrfailure)
      msg3 = 'The common dialog box procedure failed to load a specified string.'C
    case (cderr_memallocfailure)
      msg3 = 'The common dialog box procedure was unable to allocate memory for &
              internal structures.'C
    case (cderr_memlockfailure)
      msg3 = 'The common dialog box procedure was unable to load the memory &
              associated with a handle.'C
    case (cderr_nohinstance)
      msg3 = 'The enabletemplate flag was specified in the flags member of a &
              structure for the corresponding common dialog box, but the appl&
              ication failed to provide a corresponding instance handle.'C
    case (cderr_nohook)
      msg3 = 'The enablehook flag was specified in the flags member of a &
              structure for the corresponding common dialog box, but the &
              application failed to provide a pointer to a corresponding &
              hook function'C
    case (cderr_notemplate)
      msg3 = 'The enabletemplate flag was specified in the flags member of &
              a structure for the corresponding common dialog box, but the &
              application failed to provide a corresponding template.'C
    case (cderr_structsize)
      msg3 = 'The lstructsize member of a structure for the corresponding &
              common dialog box is invalid.'C
    case (fnerr_buffertoosmall)
      msg3 = 'The buffer for a filename is too small. (this buffer is pointed &
              to by the lpstrfile member of the structure for a common dialog &
              box.)'C
    case (fnerr_invalidfilename)
      msg3 = 'A filename is invalid.'C
    case (fnerr_subclassfailure)
      msg3 = 'An attempt to subclass a list box failed because insufficient &
              memory was available.'C
    case default
      msg3 = 'Unknown error number'C
  end select
  if(iret /= 0)then
    iret = messageboxqq(msg3, msg1,mb$iconexclamation .or. mb$ok)
  endif
  return
end subroutine comdlger


subroutine dialog_error_display (id)
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  integer id,iret,i
  msg1 = ' PARAMETER SETUP ERROR'C
  select case (id)
    case (IDD_MTBEParams)
      open (ErrWinUnit, file='USER', title='VOC PARAMETER INPUT ERRORS')
      if (err_dlg(1)) then
        write (ErrWinUnit, '(/a)') &
               ' Error reading MOLECULAR WEIGHT.  Weight must be numeric and >0'
        write (ErrWinUnit, '(a,a,a)') &
               ' MOLECULAR WEIGHT dialog entry:  ', trim(adjustl(err_str(1))),&
               '  is invalid'
        write (ErrWinUnit, '(//)')
      endif
      if (err_dlg(2)) then
        write (ErrWinUnit, '(/a,a)') ' Error reading INITIAL CONCENTRATION.',&
               '    Concentration must be numeric and >=0'
        write (ErrWinUnit, '(/a,a,a)') &
               ' INITIAL CONCENTRATION dialog entry:  ', &
                 trim(adjustl(err_str(2))),' is invalid'
      endif
      MSG0 = ' Error in VOC parameters\nCheck error window and re-enter'C
    case (IDD_RuntimeParams)
      open (ErrWinUnit, file='USER', title='RUNTIME PARAMETER INPUT ERRORS')
      if (err_dlg(1)) then
        write (ErrWinUnit, '(/a,a)') ' Error reading TOTAL RUNTIME. ',&
               '  Runtime must be numeric and >0'
        write (ErrWinUnit, '(/a,a,a)') ' TOTAL RUNTIME dialog entry:  ',&
               trim(adjustl(err_str(1))), '  is invalid'
        write (ErrWinUnit, '(//)')
      endif
      if (err_dlg(2)) then
        write (ErrWinUnit, '(/a)') &
               ' Error reading OUTPUT TIME STEP.  Time step must be numeric and >0'
        write (ErrWinUnit, '(/a,a,a)') ' OUTPUT TIME STEP dialog entry:  ',&
               trim(adjustl(err_str(2))), '  is invalid'
        write (ErrWinUnit, '(//)')
      endif
      if (err_dlg(3)) then
        write (ErrWinUnit, '(/a)') &
               ' Error reading TOLERANCE.  Tolerance must be numeric and >0'
        write (ErrWinUnit, '(/a,a,a)') &
               ' TOLERANCE dialog entry:  ', trim(adjustl(err_str(2))),&
               '  is invalid'
        write (ErrWinUnit, '(//)')
      endif
      if (err_dlg(4)) then
        write (ErrWinUnit, '(/a/a)') &
               'Total Time/Time Step exceeds 10,000 data points',&
               'Modify TOTAL TIME and TIME STEP accordingly'
      endif
      MSG0 = ' Error in runtime parameters\nCheck error window and re-enter'C
    case (IDD_MeteorParams)
      open (ErrWinUnit, file='USER', title = &
            'METEOROLOGICAL PARAMETER INPUT ERROR')
      if (err_dlg(1)) then
        write (ErrWinUnit, '(/a,a)') ' Error reading relative humidity. ',&
               'R.H. must be numeric, <100 and >=0'
        write (ErrWinUnit, '(a,a,a)') &
               ' RELATIVE HUMIDITY dialog entry:  ', trim(adjustl(err_str(1))),&
               '  is invalid'
        write (ErrWinUnit, '(//)')
      endif
      MSG0 = ' Error in meteorol. params\nCheck error window and re-enter'C
!   end of cases for main dialogs, time series errors from here on
    case (IDC_MixedLayer, IDC_SurfaceTemp, IDC_LakeDepth, IDC_Inflow,&
          IDC_Outflow, IDC_InflowHeight, IDC_OutflowHeight,&
          IDC_WindSpeed, IDC_AirTemp, IDC_AtmPressure, IDC_MTBEInputSeries,&
          IDC_AtmMTBEConc, IDC_EpiLossRate, IDC_HypLossRate)
      MSG0 = ' Error in time series\nCheck error window and re-enter'C
      open (ErrWinUnit, file='USER', title='TIME SERIES INPUT ERRORS')
      select case (id)
        case (IDC_MixedLayer)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Mixed-Layer Depth for Month: ',i,&
                     '  MLD must be numeric and LakeDepth >= MLD >= 0'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Mixed-Layer Depth dialog entry:  ',&
                     trim(adjustl(err_str(i))), '  is invalid'
            endif
          end do
          if (err_dlg(13)) then
```

```
              write (ErrWinUnit, '(a/a)') &
                     ' Error in splined Mixed-Layer Depth time series',&
                     '  LD >= MLD'
          endif
        case (IDC_SurfaceTemp)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Water Surface Temperature for Month: ',i,&
                     '  Temperature must be numeric'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Water Surface Temperature dialog entry:  ',&
                     trim(adjustl(err_str(i))), '  is invalid'
            endif
          end do
        case (IDC_LakeDepth)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Lake Depth for Month: ',i,&
                     '  LD must be numeric and LD <= Max LD in LakeArea'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Lake Depth dialog entry:  ', trim(adjustl(err_str(i))),&
                     '  is invalid'
            endif
          end do
          if (err_dlg(13)) then
              write (ErrWinUnit, '(a/a/a)') &
                     ' Error in splined Lake Depth time series',&
                     '  LD < MLD',&
                     '  It is possible you need to reset MLD before setting LD'
          endif
        case (IDC_Inflow)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Lake Inflow for Month: ',i,&
                     '  Inflow must be numeric and >= 0'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Lake Inflow dialog entry:  ', trim(adjustl(err_str(i))),&
                     '  is invalid'
            endif
          end do
        case (IDC_Outflow)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Lake Outflow for Month: ',i,&
                     '  Outflow must be numeric and >= 0'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Lake Outflow dialog entry:  ', trim(adjustl(err_str(i))),&
                     '  is invalid'
            endif
          end do
        case (IDC_InflowHeight)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Lake Inflow Height for Month: ',i,&
                     '  Inflow height must be numeric and > 0'
```

```
      write (ErrWinUnit, '(a,a,a/)') &
             ' Lake Inflow Height dialog entry:  ', &
             trim(adjustl(err_str(i))), '  is invalid'
    endif
  end do
case (IDC_OutflowHeight)
  do i = 1, 12
    if (err_dlg(i)) then
      write (ErrWinUnit, '(a,i2/a)') &
             ' Error reading Lake Outflow Height for Month: ',i,&
             '  Outflow height must be numeric and > 0'
      write (ErrWinUnit, '(a,a,a/)') &
             ' Lake Outflow Height dialog entry:  ',&
             trim(adjustl(err_str(i))), '  is invalid'
    endif
  end do
case (IDC_WindSpeed)
  do i = 1, 12
    if (err_dlg(i)) then
      write (ErrWinUnit, '(a,i2/a)') &
             ' Error reading Wind Speed for Month: ',i,&
             '  Wind Speed must be numeric and >0'
      write (ErrWinUnit, '(a,a,a/)') &
             ' Wind Speed dialog entry:  ', trim(adjustl(err_str(i))),&
             '  is invalid'
    endif
  end do
case (IDC_AirTemp)
  do i = 1, 12
    if (err_dlg(i)) then
      write (ErrWinUnit, '(a,i2/a)') &
             ' Error reading Air Temperature for Month: ',i,&
             '  Temperature must be numeric'
      write (ErrWinUnit, '(a,a,a/)') &
             ' Air Temperature dialog entry:  ', trim(adjustl(err_str(i))),&
             '  is invalid'
    endif
  end do
case (IDC_AtmPressure)
  do i = 1, 12
    if (err_dlg(i)) then
      write (ErrWinUnit, '(a,i2/a)') &
             ' Error reading Atm. Pressure for Month: ',i,&
             '  Pressure must be numeric and >0.0'
      write (ErrWinUnit, '(a,a,a/)') &
             ' Atm. Pressure dialog entry:  ', trim(adjustl(err_str(i))),&
             '  is invalid'
    endif
  end do
case (IDC_MTBEInputSeries)
  do i = 1, 12
    if (err_dlg(i)) then
      write (ErrWinUnit, '(a,i2/a)') &
             ' Error reading VOC Input for Month: ',i,&
             '  VOC Input must be numeric and >0'
      write (ErrWinUnit, '(a,a,a/)') &
             ' VOC Input dialog entry:  ', trim(adjustl(err_str(i))),&
             '  is invalid'
    endif
  end do
```

```
        case (IDC_AtmMTBEConc)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Atm. VOC Concentration for Month: ',i,&
                     '  Concentration must be numeric'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Atm. VOC Concentration dialog entry:  ', &
                     trim(adjustl(err_str(i))), '  is invalid'
            endif
          end do
        case (IDC_EpiLossRate)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Epilimnion Loss Rate for Month: ',i,&
                     '  Concentration must be numeric and >= 0.0'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Epilimnion Loss Rate dialog entry:  ', &
                     trim(adjustl(err_str(i))), '  is invalid'
            endif
          end do
        case (IDC_HypLossRate)
          do i = 1, 12
            if (err_dlg(i)) then
              write (ErrWinUnit, '(a,i2/a)') &
                     ' Error reading Hypolimnion Loss Rate for Month: ',i,&
                     '  Concentration must be numeric and >= 0.0'
              write (ErrWinUnit, '(a,a,a/)') &
                     ' Hypolimnion Loss Rate dialog entry:  ',&
                     trim(adjustl(err_str(i))), '  is invalid'
            endif
          end do
      end select
      if (err_dlg(13)) then
        write (ErrWinUnit, '(a)') &
                  ' General error in splined time series.  Check data entered or &
                    data file'
      endif
    end select
    ErrorWindow = .true.
    iret = messageboxqq(msg0, msg1,mb$iconexclamation .or. mb$ok)
    return
  end subroutine dialog_error_display


subroutine exitprog(checked)
  use msflib
  use mtbecom
  implicit none
  logical(kind=4)checked
  integer iret
  call unusedqq(checked)
  if (lrunning) then
    msg0 = 'Stop model before exiting program!'C
    msg1 = 'Model Status'C
    iret = messageboxqq(msg0,msg1,mb$ok)
    return
  else
    stop
  endif
```

```
end subroutine exitprog


real*8 function interpolate(t_series, time, numpts)
  use modelcom
  ! interpolation routine for SPLINEPNT-pt time series.
  implicit none
  integer i, numpts, iday
  real*8 t_series(numpts), ti, tf, time, tinc, tinc2, tday
  i = 1
  tinc = 1.0
  tinc2 = 0.5*tinc
  !calculate time in Julian days
  !variable TIME passed to interpolate gives time as decimal month in the year.
  !INTERPOLATE wants time in Julian day.  The next line converts month to day.
  tday = 365.0*time/12.0
  !this is the day index for the time series to be interpolated
  iday = idnint(tday)
  if ((tday .ge. 0.5) .and. (tday .lt. 364.5)) then
    ti = float(iday)-0.5
    tf = tday
    interpolate = t_series(iday) + (tf-ti)*(t_series(iday+1)-t_series(iday))
  endif
  if ((tday .ge. 364.5) .or. (tday .lt. 0.5)) then
    ti = 364.5
    if (tday .ge. 364.5) then
      tf = tday
      interpolate = t_series(numpts) + (tf-ti)*(t_series(1)-t_series(numpts))
    elseif (tday .lt. 0.5) then
      tf = tday + 365.0
      interpolate = t_series(numpts) + (tf-ti)*(t_series(1)-t_series(numpts))
    endif
  endif
  return
end function interpolate


real*8 function kl(t)
  ! calculates kL from wind speed and water temperature
  ! V1.8 and above modified to calculate Koa assuming liquid and gas-phase
    !  rate control
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 airt, kl_mps, t, time, u, degc, interpolate, sc, diff_calc, nu,&
         ka_h2o, ka_h2o_20, ka_voc, kl_voc, kH, sol_calc
  external interpolate, diff_calc, sol_calc
  numpts = splinepnts
  ! variable T is time in days from model start, TIME is time in months
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  u = interpolate(spl_WindSpeed, time, numpts)
  degc = interpolate(spl_SurfaceTemp, time, numpts)
  airt = interpolate(spl_AirTemp, time, numpts)
  diffusivity = diff_calc(degc, DiffParam)
  kH = sol_calc(degc, SolParam)
  ! Kin. Visc. (nu) in cm^2/s is calculated from temperature in deg-C.
  !   The underlying data are from CRC 63rd edition.
  !   The polynomial fit was done in the spreadsheet KINVISC.WB1 in QDATA
  nu = 0.017826598 - 5.76464E-04*degc +  1.12266E-05*degc**2 - 9.66507E-08*degc**3
  sc = nu/diffusivity
  ! kg estimated from H2O relation in Schwarzenbach et al. Env. Org. Chem.
  ka_h2o_20 = 0.01*(0.15*u)           !ka_h2o in m/s @ 20 deg-C
  ! correct ka @20 C to air temperature
  ka_h2o = ka_h2o_20 * dsqrt((airt+273.16)/293.16)
  ! correct ka_h2o to ka_voc
  ka_voc = ka_h2o*dsqrt(18.0/MolWeight)
  ! kL in m/s from Wanninkhof et al. (1991; GasEx2 symp. paper on Page 441)
  kl_voc = 0.01*0.45*dsqrt(600/sc)*u**1.64/3600.0
  ! if-then needed for kG param.  can't divide by zero for kl_voc at u=0
  if (u .gt. 0.0) then
    kl_mps = 1.0/(1.0/kl_voc + 1.0/(ka_voc/(kH*gasconst*(degc+273.16))))
  else
    kl_mps = 0.0
  endif
  kl = 24.0*3600.0*kl_mps  ! change m/s to m/d
  return
end function kl


real*8 function csat(t)
  ! calculates csat from water temperature
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, atmpr, deg_c, airconc, interpolate, sol_calc
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  deg_c = interpolate(spl_SurfaceTemp, time, numpts)
  airconc = interpolate(spl_AtmMTBEConc, time, numpts)
  atmpr = interpolate(spl_AtmosPress, time, numpts)
  airconc = airconc * 1.0d-9   !  change ppbv into atmospheres
  solubility = sol_calc(deg_c, SolParam) !  solubility in atm-m^3/mol
  csat = solubility * airconc * atmpr
  return
end function csat


real*8 function ii(t)
  ! function calculates the MTBE input to the lake from motorboats etc.
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 ii_kgday, t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  ii_kgday = interpolate(spl_MTBEInput, time, numpts) ! input in kg(MTBE)/d
  ii = ii_kgday*1000.0/molweight     ! input changed to mol(MTBE)/d
  return
end function ii


real*8 function mld(t)
  !Computes MLD using linear interpolation of the mixed-layer depth time series,
  !MixedLayer
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  mld_last = mld_curr
  mld_curr = interpolate(MLD_Data, time, numpts)
  mld = mld_curr
  return
end function mld


real*8 function ld(t)
  !Computes LakeDepth using linear interpolation of the lake depth time series,
  !spl_LakeDepth
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  ld = interpolate(spl_LakeDepth, time, numpts)
  return
end function ld


real*8 function EpiLoss(t)
  !computes Epilimnion Loss Rate using linear interpolation &
  !of the EpiLossRate time series, spl_EpiLossRate
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  EpiLoss = interpolate(spl_EpiLossRate, time, numpts)
  return
end function EpiLoss


real*8 function HypLoss(t)
  !computes Hypolimnion Loss Rate using linear interpolation &
  !of the HypoLossRate time series, spl_HypLossRate
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  HypLoss = interpolate(spl_HypLossRate, time, numpts)
  return
end function HypLoss
```

```
real*8 function Calc_Inflow(t)
  !computes Inflow using linear interpolation of the inflow time series,
  !spl_Inflow
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  Calc_Inflow = interpolate(spl_Inflow, time, numpts)
  return
end function Calc_Inflow


real*8 function Calc_Outflow(t)
  !computes Outflow using linear interpolation of the outflow time series,
  !spl_Outflow
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  Calc_Outflow = interpolate(spl_Outflow, time, numpts)
  return
end function Calc_Outflow


real*8 function Calc_InHeight(t)
  !computes Inflow using linear interpolation of the inflow time series,
  !spl_Inflow
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  Calc_InHeight = interpolate(spl_InflowHeight, time, numpts)
  return
end function Calc_InHeight


real*8 function Calc_OutHeight(t)
  !computes Inflow using linear interpolation of the inflow time series,
  !spl_Inflow
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  Calc_OutHeight = interpolate(spl_OutflowHeight, time, numpts)
  return
end function Calc_OutHeight
```

```
real*8 function la_func(t)
  !computes LakeArea using linear interpolation of the lake area versus depth profile
  use mtbecom
  use modelcom
  implicit none
  logical AreaFound
  integer i, iret
  real*8 depth, t, ld
  external ld
  depth = ld(t)
  i = 1
  AreaFound = .false.
  do while ((i .le. ProfilePoints-1) .and. (.not. AreaFound))
    if ((depth .le. LakeArea(i,1)) .and. (depth .gt. LakeArea(i+1,1))) then
      la_func = LakeArea(i,2) + ((LakeArea(i,1)-depth)/(LakeArea(i,1)- &
      LakeArea(i+1,1)))*(LakeArea(i,2) - LakeArea(i+1,2))
      AreaFound = .true.
    endif
    i = i + 1
  end do
  if (.not. AreaFound) then
    msg1 = 'Error Calculating Lake Area'C
    msg0 = 'la_func exited without setting Lake Area/This usually indicates a &
            problem with the Lake Area versus Depth profile/Halt the model run &
            and check the profile'
    iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    pause
    la_func = LakeArea(1,2)
  endif
  return
end function la_func


real*8 function mldp(t)
  ! calculates derivative of MLD w.r.t. time using splined MLD data in MLD_data
    ! and year_time
  use mtbecom
  use modelcom
  implicit none
  logical test
  integer i, ipnt, numpts, indexnum
  parameter (numpts=3)
  real*8 t,time,b
  ! begin subroutine
  indexnum = numpts/2 + 1
  time = 365.0*(t/365.0 - float(int(t/365.0)))
  test = .false.
  ipnt = -1
  i = 1
  if ((time .ge. year_time(splinepnts)) .or. (time .lt. year_time(1))) then
    test = .true.
    ipnt = splinepnts
  endif
  do while ((i .le. splinepnts-1) .and. (.not. test))
    if ((time .ge. year_time(i)) .and. (time .lt. year_time(i+1))) then
      ipnt = i
      test = .true.
    endif
    i = i + 1
  end do
  if (ipnt .lt. splinepnts) then
    b = (MLD_Data(ipnt+1) - MLD_Data(ipnt))/(year_time(ipnt+1) - year_time(ipnt))
  elseif (ipnt .eq. splinepnts) then
    b = (MLD_Data(1) - MLD_Data(splinepnts))/(year_time(1)+365.0 - &
         year_time(splinep-nts))
  endif
  mldp = b    ! MLD gradient in meters/day
  return
end function mldp


real*8 function diff_calc(tempc, iform)
  ! function returns D in cm^2/s which is converted to Sc for calculating kL
  ! nu used in Sc relation is also calculated in cm^2/s so no need to convert D
  ! to SI units
  use mtbecom
  use modelcom
  implicit none
  integer iform
  real*8 tempc
  real*8 mu, nu, sc
  select case (iform)
    case(1)
  ! Abs. viscosity (mu) in (g/cm)/s is calculated from temperature in deg-C.
  !   The underlying data are from CRC 63rd edition.
  !   The polynomial fit was done in the spreadsheet KINVISC.WB1 in QDATA
      mu = 1.7825047 - 0.0575921*tempc + 0.00111378*tempc**2 -&
           9.55317E-06*tempc**3
      diff_calc = (4.7199e-07*(tempc+273.16))/(mu*(MolarVolume**0.6))
    case(2)
      sc = wd0 + wd1*tempc + wd2*tempc**2 + wd3*tempc**3
  !   Kin. Visc. (nu) in cm^2/s is calculated from temperature in deg-C.
  !   The underlying data are from CRC 63rd edition.
  !   The polynomial fit was done in the spreadsheet KINVISC.WB1 in QDATA
      nu = 0.017826598 - 5.76464E-04*tempc +  1.12266E-05*tempc**2 -&
           9.66507E-08*tempc**3
      diff_calc = 0.0
      if (sc .ne. 0.0) diff_calc = nu/sc
  end select
  return
end function diff_calc


real*8 function sol_calc(tempc, iform)
  use mtbecom
  use modelcom
  implicit none
  integer iform
  real*8 tempc
  real*8 tempk, logalpha, alpha, sol1
  tempk = tempc + 273.16
  select case (iform)
    case(1)
  !   following Robbins et al., atm-m^3/mol
      sol_calc = 1.0/dexp(solA - solB/(tempk))
    case(2)
  !   calculating Bunsen solubilities assuming salinity = zero
  !   using Wanninkhof relation
      logalpha = wa0 + wa1*(100/tempk) + wa2*dlog(0.01*tempk) + &
                 salinity * (wb1 + wb2*(0.01*tempk) + wb2*(0.01*tempk)**2)
      alpha = dexp(logalpha)
      sol1 = alpha /(0.0820575 * (tempk))   ! mol/L-atm
      sol_calc = 1000.0*sol1                 ! mol/m^3-atm
  end select
  return
end function sol_calc


real*8 function flux_h2o(t)
  use mtbecom
  use modelcom
  real*8 t
  real*8 airt, tempk, degc, logvp, vp, delc, time, u, interpolate, ka_h2o, flux_mass
  external interpolate
  numpts = splinepnts
  ! variable T is time in days from model start, TIME is time in months
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  u = interpolate(spl_WindSpeed, time, numpts)
  degc = interpolate(spl_SurfaceTemp, time, numpts)
  airt = interpolate(spl_AirTemp, time, numpts)
  ! ka_H2O relation in Schwarzenbach et al. Env. Org. Chem.
  ka_h2o = 0.01*(0.2*u+0.3)*dsqrt((airt+273.16)/293.16) !ka_h2o: m/s at AirTemp
  tempk = degc+273.16
  ! Vapor pressure of water predicted from empirical fit.
  ! Data are from CRC-63rd, fit done in Quattro
  logvp = 31.4004128517 - 67.88619575*(100/tempk) -&
          5.00162020852*dlog(0.01*tempk)
  vp = dexp(logvp)
  vpatm = vp/760.000
  delc = (1.0 - rel_hum) * vpatm / (gasconst * tempk)   ! delta-C for water
  ! amount of water lost in (g/day)/m^2
  flux_mass = 24.00*3600.00*18.0* ka_h2o * delc
  flux_h2o = 1.0e-03*flux_mass  ! height of water loss in mm over time step
  return
end function flux_h2o


subroutine Lake_volume_calc
  ! calculates volumes of epilimnion and hypolimnion
  use mtbecom
  use modelcom
  implicit none
  integer i, iepilayers, ihyplayers, iTopEpi, iBotEpi, itophyp, j, iret, index
  real*8 height, epiheight, epithick, hypheight, hypthick, epivol, hypvol, t,&
         topthick, laythick, thickness, offset, r1(:), r2(:), h1(:), theta(:),&
         delh
  allocatable r1, r2, h1, theta
  !external functions
  real*8 VolCalc, dhypvol_dt, depivol_dt, epi_vol, hyp_vol, calc_inflow, calc_outflow
  external VolCalc, dhypvol_dt, depivol_dt, epi_vol, hyp_vol, calc_inflow,
           calc_outflow
  real*8 sumvol, totvol, eh(splinepnts), hh(splinepnts)
  if (allocated(r1)) deallocate (r1)
  if (allocated(r2)) deallocate (r2)
  if (allocated(h1)) deallocate (h1)
  if (allocated(theta)) deallocate (theta)
  allocate (r1(profilepoints-1))
  allocate (r2(profilepoints-1))
  allocate (h1(profilepoints-1))
  allocate (theta(profilepoints-1))
```

```
do i = 1, profilepoints - 1
  r1(i) = dsqrt(LakeArea(i,2)/pi)
  r2(i) = dsqrt(LakeArea(i+1,2)/pi)
  h1(i) = LakeArea(i,1) - LakeArea(i+1,1)
  if (r1(i) .gt. r2(i)) then  !  layer is conical
    theta(i) = datan(h1(i)/(r1(i)-r2(i)))
  elseif (r1(i) .eq. r2(i)) then
!   layer is cylindrical, theta unused in calculations
    theta(i) = 0.0
  elseif (r1(i) .lt. r2(i)) then  !  error in lake area profile
    msg1 = 'LakeVol Calculation Failure'C
    msg0 = 'Error in Lake Area versus Depth profile\n&
            Lake Areas must stay constant or decrease with depth'C
    iret = messageboxqq(msg0,msg1,MB$ICONEXCLAMATION)
    return
  endif
end do
do j = 1, splinepnts
  height = spl_LakeDepth(j)
  epiheight = height - MLD_Data(j)
  epithick = MLD_Data(j)
  if (epithick .eq. 0.0) epithick = height
! if MLD_Data= 0., no mixed layer
  hypthick = height - epithick
  hypheight = epiheight
  if (epiheight .lt. height) then   !STRATIFIED LAKE
    do i = 1, profilepoints-1
      if ((epiheight .gt. LakeArea(i+1,1)) .and.&
          (epiheight .le. LakeArea(i,1))) then
        ibotepi = i + 1
        itophyp = i
      endif
      if ((height .gt. LakeArea(i+1,1)) .and.&
          (height .le. LakeArea(i,1))) then
        itopepi = i
      endif
    end do
    iepilayers = ibotepi - itopepi
    ihyplayers = ProfilePoints - itophyp
  else                 ! unstratified lake
    itopepi = 1
    iepilayers = ProfilePoints-1
    ihyplayers = 0
  endif       !STRATIFIED LAKE ENDIF
  if (iepilayers .eq. 1) then    ! START CALCULATING EPILIMNION VOLUME
    thickness = epithick
    offset = height - epithick - LakeArea(itopepi+1,1)
    epivol = VolCalc(r1(itopepi), r2(itopepi), h1(itopepi), thickness,&
                     offset, theta(itopepi))
    spl_epivol(j) = epivol
  elseif (iepilayers .eq. 2) then
    thickness = height-LakeArea(itopepi+1,1)
    offset = 0.0
    epivol = VolCalc(r1(itopepi), r2(itopepi), h1(itopepi), thickness,&
                     offset, theta(itopepi))
    thickness = epithick - thickness
    index = itopepi + 1
    offset = LakeArea(index,1) - thickness - LakeArea(index+1,1)
    epivol = epivol + VolCalc(r1(index), r2(index), h1(index), thickness,&
                              offset, theta(index))
      spl_epivol(j) = epivol
    elseif (iepilayers .gt. 2) then
      thickness = height-LakeArea(itopepi+1,1)
      topthick = thickness
      offset = 0.0
      epivol = VolCalc(r1(itopepi), r2(itopepi), h1(itopepi), thickness,&
               offset, theta(itopepi))
      do i = 2, iepilayers-1
        thickness = LakeArea(itopepi+i-1,1) - LakeArea(itopepi+i,1)
        topthick = topthick + thickness
        index = itopepi + i - 1
        epivol = epivol + VolCalc(r1(index), r2(index), h1(index), thickness,&
                 offset, theta(index))
      end do
      thickness = epithick - topthick
      offset = LakeArea(itopepi+iepilayers-1,1) - thickness -&
               LakeArea(itopepi+iepilayers,1)
      index = itopepi + iepilayers - 1
      epivol = epivol + VolCalc(r1(index), r2(index), h1(index), thickness,&
               offset, theta(index))
      spl_epivol(j) = epivol
    endif     !  END CALCULATING EPILIMNION VOLUME
    if (ihyplayers .eq. 0) then    ! START CALCULATING HYPOLIMNION VOLUME
      hypvol = 0.0
      spl_hypvol(j) = hypvol
    elseif (ihyplayers .eq. 1) then
      index = profilePoints - 1
      offset = 0.0
      hypvol = VolCalc(r1(index), r2(index), h1(index), hypthick, offset,&
               theta(index))
      spl_hypvol(j) = hypvol
    elseif (ihyplayers .ge. 2)  then
      hypvol = 0.0
      laythick = 0.0
      offset = 0.0
      do i = 1, ihyplayers-1
        index = ProfilePoints - i
        laythick = (LakeArea(index,1)-LakeArea(index+1,1))
        hypthick = hypthick - laythick
        hypvol = hypvol + VolCalc(r1(index), r2(index), h1(index), laythick,&
                 offset, theta(index))
      end do
      index = ProfilePoints - ihyplayers
      hypvol = hypvol + VolCalc(r1(index), r2(index), h1(index), hypthick,&
               offset, theta(index))
      spl_hypvol(j) = hypvol
    endif                    !  END OF HYPOLIMNION VOLUME CALCULATION
  end do
  if (allocated(r1)) deallocate (r1)
  if (allocated(r2)) deallocate (r2)
  if (allocated(h1)) deallocate (h1)
  if (allocated(theta)) deallocate (theta)
  return
end subroutine Lake_volume_calc


real*8 function VolCalc(r1, r2, htot, hlay, hbeg, theta)
  use parameters  !  gives access to variable pi
  implicit none
  ! passed variables
  real*8 r1, r2, htot, hlay, hbeg, theta
  !     r1 = radius of top layer from LakeArea
  !     r2 = radius of bottom layer from LakeArea
  !     htot = total depth between r1 and r2
  !     hlay is thickness of layer to calculate volume
  !     hbeg is offset from bottom of layer at r2 for calculating volume
  !     (hbeg=0 starts at bottom)
  ! local variables
  real*8 a1, abot, atop, cbot, ctop, rbot, rtop, vtop, vbot
  !     a1 is area of very top layer for conical volume
  !     atop is area of top of volume
  !     abot is area of bottom of volume
  !     ctop is conical height of top area
  !     cbot is conical height of bottom area
  !     rbot is radius of bottom
  !     rtop is radius of top
  !     vtop is volume of top cone
  !     vbot is volume of bottom cone
  a1 = pi * r1**2
  if (r1 .eq. r2) then   !  cylindrical layer, only need a1 for volume
    VolCalc = a1 * hlay
    return
  else     !   conical layer, need all three areas for volume
    rbot = r2 + hbeg * (r1 - r2) / htot
    rtop = r2 + (hlay + hbeg) * (r1 - r2) / htot
    abot = pi * rbot**2
    atop = pi * rtop**2
    cbot = rbot * dtan(theta)
    ctop = rtop * dtan(theta)
    vbot = 0.333333333 * abot * cbot
    vtop = 0.333333333 * atop * ctop
    VolCalc = vtop - vbot
    return
endif
return
end function VolCalc


real*8 function epi_vol(t)
  ! function calculates the epilimnion volume
  use mtbecom
  use modelcom
  implicit none
  integer numpts
  real*8 volume, t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  volume = interpolate(spl_epivol, time, numpts) ! epilimnion volume in m^3
  epi_vol = volume
  return
end function epi_vol


real*8 function hyp_vol(t)
  ! function calculates the hypolimnion volume
  use mtbecom
  use modelcom
  implicit none
  integer numpts, iret
  real*8 volume, t, time, interpolate
  external interpolate
  numpts = splinepnts
  time = 12.0*(t/365.0 - float(int(t/365.0)))
  volume = interpolate(spl_hypvol, time, numpts) ! hypolimnion volume in m^3
  hyp_vol = volume
      if (hyp_vol .lt. 0.0) then
        msg1 = 'Error in Lake Volume'C
        write (msg0, '(a, f12.5, a, e15.5)')&
                      'time: ', time, '    Hyp_Vol: ', hyp_vol
        iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
      endif
  return
end function hyp_vol


real*8 function depivol_dt(t)
  ! calculates derivative of lake volumes w.r.t. time using volume time series
  use mtbecom
  use modelcom
  implicit none
  logical test
  integer i, ipnt, numpts, indexnum, iday
  parameter (numpts=3)
  real*8 t, b, tday
  ! begin subroutine
  indexnum = numpts/2 + 1
  tday = 365.0*(t/365.0 - float(int(t/365.0)))
  test = .false.
  iday = idint(tday)
  if ((tday .gt. 1.0) .and. (tday .lt. 365.0)) then
    b = spl_EpiVol(iday+1) - spl_epivol(iday)
  elseif ((tday .ge. 365.0) .or. (tday .le. 1.0)) then
    b = spl_EpiVol(1) - spl_epivol(splinepnts)
  endif
  depivol_dt = b             ! change in volume in m^3/d
  return
end function depivol_dt


real*8 function dhypvol_dt(t)
  use mtbecom
  use modelcom
  implicit none
  logical test
  integer i, ipnt, numpts, indexnum, iday
  parameter (numpts=3)
  real*8 t, b, tday
  ! begin subroutine
  indexnum = numpts/2 + 1
  tday = 365.0*(t/365.0 - float(int(t/365.0)))
  test = .false.
  iday = idint(tday)
  if ((tday .gt. 1.0) .and. (tday .lt. 365.0)) then
    b = spl_HypVol(iday+1) - spl_HypVol(iday)
  elseif ((tday .ge. 365.0) .or. (tday .le. 1.0)) then
    b = spl_HypVol(1) - spl_HypVol(splinepnts)
  endif
  dhypvol_dt = b                   ! hypolimnion volume change in m^3/d
  return
```

```
end function dhypvol_dt


!This file contains most of the subroutines and functions used to initialize
!spline, and reset the various time series used in the model.  The values for
!the index variables (e.g., indexTW, indexMLD) are set in PARAMETS.F90


subroutine spline_data(index)
  ! subroutine grids the monthly MLD's to a finer mesh for calculating derivatives
  use mtbecom
  use modelcom
  use tser_com
  implicit none
  logical epi_forming
  integer i, j, index, numpts, imax, MLDindex(365), MLDpnts, &
          index_beg, index_end, indexstep, ts_index_new, ts_index_predict
  parameter (numpts=3)
  real*8 timefrac, monthconv, weekconv, time, time_beg, time_end
  !begin subroutine
    monthconv = 12.0/365.0
    weekconv = 52.0/365.0
    timefrac = 1.0
    if (TSDatLen(index) .eq. 365) then
      do i = 1, 365
        select case (index)
          case (indexTW)
            spl_SurfaceTemp(i) = SurfaceTemp(i)
          case (indexMLD)
            MLD_Data(i) = MixedLayer(i)
          case (indexLD)
            spl_LakeDepth(i) = LakeDepth(i)
          case (indexIN)
            spl_Inflow(i) = Inflow(i)
          case (indexOUT)
            spl_Outflow(i) = Outflow(i)
          case (indexINHe)
            spl_InflowHeight(i) = InflowHeight(i)
          case (indexOUTHe)
            spl_OutflowHeight(i) = OutflowHeight(i)
          case (indexTA)
            spl_AirTemp(i) = AirTemp(i)
          case (indexU)
            spl_WindSpeed(i) = WindSpeed(i)
          case (indexPA)
            spl_AtmosPress(i) = AtmosPress(i)
          case (indexMTBE)
            spl_MTBEInput(i) = MTBEInput(i)
          case (indexAirMTBE)
            spl_AtmMTBEConc(i) = AtmMTBEConc(i)
          case (indexEpiL)
            spl_EpiLossRate(i) = EpiLossRate(i)
          case (indexHypL)
            spl_HypLossRate(i) = HypLossRate(i)
          case default
            spl_dummy(i) = dummy_dat(i)
        end select
      end do
  ! begin section for weekly data
    elseif (TSDatLen(index) .eq. 52) then
      imax = 52
      call ts_do_the_spline(timefrac, imax, index, weekconv)
  !   SPECIAL PROCESSING REQUIRED FOR INDIVIDUAL TIME SERIES
  !   So far U and MTBEInput require special handling
      select case (index)
        case (indexU)
          call ts_smooth(index, numpts)   ! wind speed
        case (indexMTBE)
          do i = 1, splinepnts
            spl_MTBEInput(i) = spl_MTBEInput(i)*weekconv
          end do
      end select
  ! begin section for monthly data
    elseif (TSDatLen(index) .eq. 12) then
      imax = 12
      call ts_do_the_spline(timefrac, imax, index, monthconv)
  !   SPECIAL PROCESSING REQUIRED FOR INDIVIDUAL TIME SERIES
  !   So far U and MTBEInput require special handling
      select case (index)
        case (indexU)
          call ts_smooth(index, numpts)   ! wind speed
        case (indexMTBE)
          do i = 1, splinepnts
            spl_MTBEInput(i) = spl_MTBEInput(i)*monthconv
          end do
      end select
    endif
    if (index .eq. indexLD) then  ! see if changed LD and reset MaxDepth
      maxdepth = 0.0
      do i = 1,365
        if (spl_LakeDepth(i) .gt. maxdepth) maxdepth = spl_LakeDepth(i)
      end do
    endif
    if ((index .eq. indexMLD) .and. (.not. MLD_setyet)) MLD_setyet = .true.
    if (((index .eq. indexLD) .or. (index .eq. indexMLD)) .and. MLD_setyet) &
       then  ! check to see if this is LD or MLD, special stuff for MLD_data
      do i = 1, 365
        MLDindex(i) = 0   ! first clear array showing where MLD forming happens
      end do
      j = 0
      do i = 1, 365
        if (MLD_data(i) .eq. spl_LakeDepth(i)) then
          j = j + 1
          MLD_Data(i) = 0.0
          MLDindex(j) = i
        endif
      end do
      if (TSDatLen(index) .ne. 365) then
        MLDpnts = j
        i = 1
        do while (i .le. MLDpnts-1)
          epi_forming = .false.
          do while (((MLDindex(i+1)-MLDindex(i)).eq.1) .and. (i.le.MLDpnts-1))
            i = i + 1
          end do
          if (i .lt. MLDpnts) then
            if ((MLDindex(i+1)-MLDindex(i)) .gt. 1) epi_forming = .true.
            if (epi_forming) then
              time = float(MLDindex(i))
              select case (TSDatLen(index))
```

```
                case (52)
                  indexstep = 7     ! number of days until next point (week)
                case(12)
                  indexstep = 30    ! number of days until next point (month)
              end select
              do j = 1, indexstep
                ts_index_new = mod(MLDindex(i)+j, 365)
                if (ts_index_new .eq. 0) ts_index_new = 365
                ts_index_predict = mod(MLDindex(i)+indexstep+1, 365)
                if (ts_index_predict .eq. 0) ts_index_predict = 365
                MLD_Data(ts_index_new) = &
                  (float(j)/float(indexstep))*MLD_Data(ts_index_predict)
              end do
            endif
            i = i + 1
          endif
        end do
      endif
    endif
    return
end subroutine spline_data


subroutine ts_do_the_spline(timefrac, imax, index, timeconv)
  implicit none
  real*8 timefrac, timeconv
  integer imax, index
  integer i, j
  real*8 temptime, time_week, ti, tf, time
  logical test
      do j = 1, 365
        time = float(j)*timefrac
        test = .false.
        i = 1
        do while ((i .le. imax-1) .and. (.not. test))
          ti = float(i) - 0.5
          tf = ti + 1.0
          time_week = time*timeconv
          if ((time_week .ge. ti) .and. (time_week .lt. tf)) then
            call set_value_month(i, i+1, j, index, time_week, ti)
            test = .true.
          endif
          i = i + 1
        end do
        if (.not. test) then
          if (time_week .ge. float(imax)-0.5) then
            temptime = float(imax)-0.5
            call set_value_month(imax, 1, j, index, time_week, temptime)
            test = .true.
          elseif (time_week .lt. 0.5) then
            temptime = -0.5
            call set_value_month(imax, 1, j, index, time_week, temptime)
            test = .true.
          endif
        endif
      enddo
    return
end subroutine ts_do_the_spline
```

```
subroutine set_value_month (i, iplus1, j, index, tmonth, ti)
  use mtbecom
  use modelcom
  use tser_com
  implicit none
  integer i, iplus1, j, index
  real*8 tmonth, ti
    select case (index)
      case (indexTW)
        spl_SurfaceTemp(j) = SurfaceTemp(i) + &
                            (tmonth-ti)*(SurfaceTemp(iplus1)-SurfaceTemp(i))
      case (indexMLD)
        MLD_data(j) = MixedLayer(i) + &
                     (tmonth-ti)*(MixedLayer(iplus1)-MixedLayer(i))
      case (indexLD)
        spl_LakeDepth(j) = LakeDepth(i) + &
                          (tmonth-ti)*(LakeDepth(iplus1)-LakeDepth(i))
      case (indexIN)
        spl_Inflow(j) = Inflow(i) + (Inflow(iplus1)-Inflow(i))
      case (indexOUT)
        spl_Outflow(j) = Outflow(i) + (Outflow(iplus1)-Outflow(i))
      case (indexINHe)
        spl_InflowHeight(j) = InflowHeight(i) + &
                             (InflowHeight(iplus1)-InflowHeight(i))
      case (indexOUTHe)
        spl_OutflowHeight(j) = OutflowHeight(i) + &
                              (OutflowHeight(iplus1)-OutflowHeight(i))
      case (indexTA)
        spl_AirTemp(j) = AirTemp(i) + &
                        (tmonth-ti)*(AirTemp(iplus1)-AirTemp(i))
      case (indexU)
        spl_WindSpeed(j) = WindSpeed(i) + &
                          (tmonth-ti)*(WindSpeed(iplus1)-WindSpeed(i))
      case (indexPA)
        spl_AtmosPress(j) = AtmosPress(i) + &
                           (tmonth-ti)*(AtmosPress(iplus1)-AtmosPress(i))
      case (indexMTBE)
        spl_MTBEInput(j) = MTBEInput(i) + &
                          (tmonth-ti)*(MTBEInput(iplus1)-MTBEInput(i))
      case (indexAirMTBE)
        spl_AtmMTBEConc(j) = AtmMTBEConc(i) + &
                            (tmonth-ti)*(AtmMTBEConc(iplus1)-AtmMTBEConc(i))
      case (indexEpiL)
        spl_EpiLossRate(j) = EpiLossRate(i) + &
                            (tmonth-ti)*(EpiLossRate(iplus1)-EpiLossRate(i))
      case (indexHypL)
        spl_HypLossRate(j) = HypLossRate(i) + &
                            (tmonth-ti)*(HypLossRate(iplus1)-HypLossRate(i))
      case default
        spl_dummy(j) = dummy_dat(i) + &
                      (tmonth-ti)*(dummy_dat(iplus1)-dummy_dat(i))
    end select
    return
end subroutine set_value_month


subroutine Initialize_TimeSeries
  use mtbecom
  use tser_com
  use modelcom
  implicit none
  integer i
  real*8 tLakeDepth(12), tMixedLayer(12), tSurfaceTemp(12), tInflow(12),&
         tOutflow(12), tInflowHeight(12), tOutflowHeight(12), tAirTemp(12),&
         tWindSpeed(12), tAtmosPress(12), tMTBEInput(12), tAtmMTBEConc(12),&
         tEpiLossRate(12), tHypLossRate(12)
  real*8 tLakeArea
  data tSurfaceTemp /14.6, 14.3, 13.9, 15.5, 21.3, 24.4, 25.3, 25.7, 25.9,&
                     21.1, 19.2, 16.8/
  data tMixedLayer /28., 2., 3., 6. , 7., 8., 9., 9., 10., 14., 28., 28./
  data tLakeDepth/28., 28., 28., 28., 28., 28., 28., 28., 28., 28., 28., 28./
  data tInflow/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
  data tOutflow/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
  data tInflowHeight/10.,10.,10.,10.,10.,10.,10.,10.,10.,10.,10.,10./
  data tOutflowHeight/10.,10.,10.,10.,10.,10.,10.,10.,10.,10.,10.,10./
  data tLakeArea/9.1e6/
  data tAirTemp /9.4, 11.3, 12.3, 14.3, 17.8, 20.8, 25.0, 25.0, 23.6, 19.3,&
                 14.1, 10.4/
  data tWindSpeed /2.07, 2.19, 2.09, 2.19, 2.12, 2.04, 1.99, 1.94, 1.86, &
                   1.89, 2.04, 2.04/
  data tAtmosPress/0.9883,0.9883,0.9883,0.9883,0.9883,0.9883,0.9883,0.9883,&
                   0.9883, 0.9883,0.9883,0.9883/
  data tMTBEInput /0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
  data tAtmMTBEConc /1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0/
  data tEpiLossRate/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
  data tHypLossRate/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
    call reset_allocation
    maxdepth = 0.0
    do i = 1, 12
      SurfaceTemp(i) = tSurfaceTemp(i)
      MixedLayer(i) = tMixedLayer(i)
      LakeDepth(i) = tLakeDepth(i)
      Inflow(i) = tInflow(i)
      Outflow(i) = tOutflow(i)
      InflowHeight(i) = tInflowHeight(i)
      OutflowHeight(i) = tOutflowHeight(i)
      AirTemp(i) = tAirTemp(i)
      WindSpeed(i) = tWindSpeed(i)
      AtmosPress(i) = tAtmosPress(i)
      MTBEInput(i) = tMTBEInput(i)
      AtmMTBEConc(i) = tAtmMTBEConc(i)
      EpiLossRate(i) = tEpiLossRate(i)
      if (maxDepth .lt. LakeDepth(i)) MaxDepth = LakeDepth(i)
    end do
  ! initial Lake Area vs. Depth Profile is for lake with vertical sides
    LakeArea(1,1) = MaxDepth
    LakeArea(1,2) = tLakeArea
    LakeArea(2,1) = 0.0
    LakeArea(2,2) = tLakeArea
    return
end subroutine Initialize_TimeSeries


subroutine reset_allocation
  use tser_com
  use mtbecom
  implicit none
  ProfilePoints = 2
  if (Allocated(SurfaceTemp)) deallocate (SurfaceTemp)
  if (allocated(MixedLayer)) deallocate (MixedLayer)
  if (allocated(LakeDepth)) deallocate (LakeDepth)
  if (allocated(Inflow)) deallocate (Inflow)
  if (allocated(Outflow)) deallocate (Outflow)
  if (allocated(InflowHeight)) deallocate (InflowHeight)
  if (allocated(OutflowHeight)) deallocate (OutflowHeight)
  if (allocated(LakeArea)) deallocate (LakeArea)
  if (allocated(AirTemp)) deallocate (AirTemp)
  if (allocated(WindSpeed)) deallocate (WindSpeed)
  if (allocated(AtmosPress)) deallocate (AtmosPress)
  if (allocated(MTBEInput)) deallocate (MTBEInput)
  if (allocated(AtmMTBEConc)) deallocate (AtmMTBEConc)
  if (allocated(EpiLossRate)) deallocate (EpiLossRate)
  if (allocated(HypLossRate)) deallocate (HypLossRate)
  allocate (SurfaceTemp(12))
  allocate (MixedLayer(12))
  allocate (LakeDepth(12))
  allocate (Inflow(12))
  allocate (Outflow(12))
  allocate (InflowHeight(12))
  allocate (OutflowHeight(12))
  allocate (LakeArea(ProfilePoints,2))
  allocate (AirTemp(12))
  allocate (WindSpeed(12))
  allocate (AtmosPress(12))
  allocate (MTBEInput(12))
  allocate (AtmMTBEConc(12))
  allocate (EpiLossRate(12))
  allocate (HypLossRate(12))
  return
end subroutine reset_allocation


subroutine ts_smooth(index, numpts)
  use modelcom
  use mtbecom
  use tser_com
  implicit none
  integer index, numpts
  integer indexnum, i, j, k
  real*8 array(numpts), datapoint, arraysum, get_series_point
  external get_series_point
    indexnum = numpts/2
    do i = 1, numpts-1
      k = mod(i-indexnum, splinepnts)
      if (k .le. 0) then
        k = k + splinepnts
      endif
        array(i) = get_series_point(index,k)
    enddo
    do i = 1, splinepnts
      j = mod(i+indexnum, splinepnts)
      if (j .eq. 0) j = splinepnts
      k = mod(i+numpts-1, numpts)
      if (k .eq. 0) k = numpts
      array(k) = get_series_point(index,j)
      arraysum = array(1)
      do k = 2, numpts
        arraysum = arraysum + array(k)
      enddo
      datapoint = arraysum/float(numpts)
      call ts_series_select(index, datapoint, i)
    enddo
  return
end subroutine ts_smooth


real*8 function get_series_point(index, j)
  use tser_com
  use mtbecom
  implicit none
  integer index, j
    select case (index)
      case (indexTW)
        get_series_point = spl_SurfaceTemp(j)
      case (indexMLD, NumTimeSeries+1)
        get_series_point = MLD_data(j)
      case (indexLD)
        get_series_point = spl_LakeDepth(j)
      case (indexIN)
        get_series_point = spl_Inflow(j)
      case (indexOUT)
        get_series_point = spl_Outflow(j)
      case (indexINHe)
        get_series_point = spl_InflowHeight(j)
      case (indexOUTHe)
        get_series_point = spl_OutflowHeight(j)
      case (indexTA)
        get_series_point = spl_AirTemp(j)
      case (indexU)
        get_series_point = spl_WindSpeed(j)
      case (indexPA)
        get_series_point = spl_AtmosPress(j)
      case (indexMTBE)
        get_series_point = spl_MTBEInput(j)
      case (indexAirMTBE)
        get_series_point = spl_AtmMTBEConc(j)
      case (indexEpiL)
        get_series_point = spl_EpiLossRate(j)
      case (indexHypL)
        get_series_point = spl_HypLossRate(j)
    end select
    return
end function get_series_point


subroutine ts_series_select(index, datapoint, j)
  use tser_com
  use mtbecom
  implicit none
  integer index, j
  real*8 datapoint
    select case (index)
      case (indexTW)
        spl_SurfaceTemp(j) = datapoint
      case (indexMLD)
        MLD_data(j) = datapoint
      case (indexLD)
        spl_LakeDepth(j) = datapoint
      case (indexIN)
```

```
        spl_Inflow(j) = datapoint
      case (indexOUT)
        spl_Outflow(j) = datapoint
      case (indexINHe)
        spl_InflowHeight(j) = datapoint
      case (indexOUTHe)
        spl_OutflowHeight(j) = datapoint
      case (indexTA)
        spl_AirTemp(j) = datapoint
      case (indexU)
        spl_WindSpeed(j) = datapoint
      case (indexPA)
        spl_AtmosPress(j) = datapoint
      case (indexMTBE)
        spl_MTBEInput(j) = datapoint
      case (indexAirMTBE)
        spl_AtmMTBEConc(j) = datapoint
      case (indexEpiL)
        spl_EpiLossRate(j) = datapoint
      case (indexHypL)
        spl_HypLossRate(j) = datapoint
    end select
    return
end  subroutine ts_series_select


subroutine Pause_Model(checked)
  use mtbecom
  use modelcom
  use errorcom
  use inputinfo
    implicit none
    logical(kind=4)checked
    call unusedqq(checked)
    pause_mod = .true.
    return
end subroutine Pause_Model


subroutine Continue_Model(checked)
  use mtbecom
  use modelcom
  use errorcom
  use inputinfo
    implicit none
    integer iret
    logical(kind=4)checked
    call unusedqq(checked)
    if (menuactive) then
      msg0 = 'Please close open set-up menu\nbefore restarting model'C
      msg1 = 'Window Error'C
      iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
      return
  endif
  pause_mod = .false.
  return
end subroutine Continue_Model


subroutine restore_default (checked)
```

```
  use msflib
  use dialogm
  use mtbecom
  use errorcom
  implicit none
  include 'resource.fd'
  type(dialog) dlg
  logical(kind=4) ret
  integer(kind=4) iret, ierr
  external ResetParams_OK
  logical (kind=4) checked
    ret = checked
    ierr = 0
    msg0 = ''c
    msg1 = ''c
    if (MenuActive) then
      msg0 = 'Please close open set-up menu\nbefore opening new window'C
      msg1 = 'Window Error'C
      iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
      return
    endif
    if (.not. lrunning) then
      menuactive = .true.
  !   initialize the dialog box
      ret = dlginit(IDD_ResetParams, dlg)
  !   write initial values and set subroutines
      ret = dlgsetsub(dlg, IDOK, ResetParams_OK)
      iret = dlgmodal(dlg)
      call dlguninit(dlg)
      menuactive = .false.
      msg0 = 'Setup parameters reset to\ntheir default values'C
      msg1 = 'Information'C
      iret = messageboxqq(msg0, msg1, MB$OK)
    else
      msg0 = &'Model running: cannot change parameters\n&
             Press <Run\\Stop> to end'C
      msg1 = ' PARAMETER SETUP ERROR'C
      iret = messageboxqq(msg0, msg1, MB$ICONEXCLAMATION .OR. MB$OK)
    endif
    return
end subroutine restore_default


subroutine ResetParams_OK (dlg, id, callbacktype)
  use msflib
  use dialogm
  use mtbecom
  use modelcom
  use errorcom
  use tser_com
  implicit none
  type(dialog) dlg
    character*72 dtitle, dcomment(2)
    integer dTSSetup(NumTimeSeries), dTSDatLen(NumTimeSeries)
    integer i, id, callbacktype, dDiffParam, dSolParam, dProfilePoints
    real*8 dinitconc, dTotalRuntime, dOutputTimestep
    real*8 dMolarVolume, dMolWeight
    real*8 dsola, dsolb, dwa0, dwa1, dwa2, dwb0, dwb1, dwb2, dsal, dwd0, dwd1, &
           dwd2, dwd3, dTol, drel_hum
    data dtitle &
```

```
     /'Lake Perris Default Data Set; Atmospheric Equilibrium; No Boat Input'/
    data dcomment(1) /'Default Model Data Set-Comment #1'/
    data dcomment(2) /'Default Model Data Set-Comment #2'/
    data dProfilePoints/2/
    data dTotalRuntime,dOutputTimestep,dinitconc/2.0,1.0,0.20/
    data dDiffParam,dSolParam,dMolWeight,dMolarVolume /1,1,88.15,129.4/
    data dsola, dsolb, dwa0, dwa1, dwa2, dwb0, dwb1, dwb2, dsal, dwd0, dwd1, &
         dwd2, dwd3, drel_hum/18.4,7666.0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.7/
    data dTol /1.0d-8/
    call unusedqq (dlg, id, callbacktype)
  ! ok button checked, reset all of the parameters
  ! Reset the number of Area/Depth profile pnts BEFORE calling reset_allocation
    ProfilePoints = dProfilePoints
  ! reset all the time series parameters
    do i = 1, NumTimeSeries
      FileLoaded(i) = .false.
      TSError(i) = .false.
      TSAllocated(i) = .false.
      TSSetup(i) = 1
      TSDatlen(i) = 12
      File_Status(i) = 'Unknown'
      File_Status2(i) = 'Not Loaded'
      FileName(i) = ''
      DataDir(i) = ''
    end do
  ! reset all the allocatable arrays, found in INTERP_F
    call initialize_timeseries
    do i = 1, NumTimeSeries
      call spline_data(i)
    end do
    call lake_volume_calc
    TotalRuntime = dTotalRuntime
    OutputTimestep = dOutputTimestep
    DiffParam = dDiffParam
    SolParam = dSolParam
    MolWeight = dMolWeight
    MolarVolume = dMolarVolume
    sola = dsola
    solb = dsolb
    wa0 = dwa0
    wa1 = dwa1
    wa2 = dwa2
    wb0 = dwb0
    wb1 = dwb1
    wb2 = dwb2
    salinity = dsal
    wd0 = dwd0
    wd1 = dwd1
    wd2 = dwd2
    wd3 = dwd3
    Initial_MTBEConc = dinitconc
    Tolerance = dTol
    rel_hum = drel_hum
    title = dtitle
    comment(1) = dcomment(1)
    comment(2) = dcomment(2)
    call dlgsetreturn(dlg, IDOK)
    call dlgexit(dlg)
    return
end subroutine ResetParams_OK
```

```
subroutine go_mtbedrv(arg2)
!**********************************************************************!*
!*  description for subroutine go_mtbedrv(arg2)
!*
!*  this subroutine is used to call the subroutine that does the mtbe
!*  modeling calculations
!*
!**********************************************************************
  use dfmt
  implicit none
  integer(4) arg2
  arg2 = 0
  call mtbedrv
  call exitthread(0)!exit code is 0
  return
end subroutine go_mtbedrv


subroutine mtbedrv
  use msflib, setpixel0=>setpixel
  use mtbecom
  use modelcom
  use errorcom
  use scigraph
  use inputinfo
  use tser_com
  implicit none
  integer i, j, irelab, ifail, iret
  real*8 conc(num_eqs), w(num_eqs,20)
  real*8 time_beg, total_time, cfunc, gout, csat,mld,mldp, maxminfunc,&
         ld, la_func, epi_vol, hyp_vol
  external cfunc, gout, csat, mld, mldp, maxminfunc, ld, la_func, &
           epi_vol, hyp_vol
  call clearscreen($GCLEARSCREEN)
  write (*, '(1x,a72)') title
  total_time = 365.0 * TotalRunTime
! set the total number of points to plot in GRAPHOUT
  itotalsets = 10 + int(total_time/OutputTimestep)
  do i = 1, isets
    do j = 1, maxoldpts
      data_save(1,j,i) = float(j-1)*OutputTimestep/365.0
    end do
  end do
  time_beg = 0.0
  MaxInput = maxminfunc(spl_MTBEInput, 1, splinepnts)
  MinVolume = maxminfunc(spl_EpiVol, -1, splinepnts)
  MaxConc = 0.333*1000.0*MaxInput/(MinVolume*molweight)
  if ((maxconc .eq. 0.0) .or. (maxconc .lt. csat(time_beg))) &
       maxconc = 1.3*csat(time_beg)
! 1000.0*molweight changes maxconc from mol/m^3 to ug/L
  maxconc = 1000.0*molweight*maxconc
  if (maxconc .lt. Initial_MTBEConc) maxconc = 1.3*Initial_MTBEConc
  if (maxconc .gt. 0.0) then
    scalefactor = 10.0**(-1*int(dlog10(maxconc)))
  else
    maxconc = 1.0
    scalefactor = 1.0
  endif
  MaxConc = MaxConc*scalefactor
! conc(1) = epilimnion volume
! conc(2) = hypolimnion volume
! conc(3) = Epilimnion concentration (mol m^-3)
! conc(4) = Hypolimnion concentration (mol m^-3)
! conc(5) = total mass in lake, unused outside of RKINTOUT
  conc(1) = epi_vol(time_beg)
  conc(2) = hyp_vol(time_beg)
  conc(3) = Initial_MTBEConc/(1000.0*molweight)
  conc(4) = conc(3)
! specifies wintertime with no mixed-layer
  conc(5) = (conc(2)*mld(time_beg)+conc(2)*(ld(time_beg)-&
             mld(time_beg)))/ld(time_beg)
  hypconc_curr = conc(4)
  lastconc = conc(3)
  hypconc_last = conc(4)
  irelab = 0
! 0 = mixed error test, 1 = decimal places, 2 = signif. figs.
  ifail = 0
  plotpnts = 0
  PlotInit = .false.
! reset PlotInit so that axes will be redrawn
  OpenWindow = .true.
! tells Plot subroutine to open graphics window
  ch_index = 1
! specifies wintertime with no mixed-layer
  pause_mod = .false.
  if (.not. DatOut) then
    msg0 = 'Model output will not be saved to data file\n&
           Continue run?'C
    iret = messageboxqq&
(msg0,'Data Output Status'C,MB$ICONEXCLAMATION .OR. MB$YESNO)
    if (iret == MB$IDNO) lrunning = .false.
  else
    write (DatFilUnit,'(a)') &
'   Time C(Epil)(ug/L) C(Hypol)(ug/L) VolEpi(m^3) VolHyp(m^3) U(m/s) &
    Tw kL(m/d) & MLD dMLD/dt Input Cs Cair Evap(mm/day) &
    Case InHeight & outheight inflow outflow iexchange makeup makeup_mass LakeArea'
  endif
  if (lrunning) then
!   note that TOLERANCE is set by user in dialog menu RUNTIME params
    call d02bbf(time_beg,total_time,num_eqs,conc,tolerance,irelab,&
         cfunc, gout,w,ifail)
    iret = messageboxqq('Run completed'C,'Model Status'C,MB$OK)
    lrunning = .FALSE.
    menuactive = .false.
  else
    IRET = messageboxqq('Run stopped by user'C,'Model Status'C,MB$OK)
    menuactive = .false.
  endif
  if (DatOut) then
    close (DatFilUnit)
    DatOut = .false.
    write (MSG0, '(a)') datfile_out(1:len_trim(datfile_out))
      msg1 = 'Closed Output File'C
        iret = messageboxqq(msg0, msg1, mb$iconexclamation .or. mb$ok)
  endif
  return
end subroutine mtbedrv
```

```
subroutine XYPlot(time, data1, ipnts)
  use msflib, setpixel0=>setpixel
  use mtbecom
  use modelcom, chsave=>ch_index
  use scigraph
  implicit none
  record /DataSettings/ OldData(3)       ! 3 data sets (ranges)
  logical plotolddata
  character*20 xyDataLegends(3)          ! data legends
  character*25 dummytitle
  integer  retcode, ipnts, i, j
  integer  setlegends, iscale
  real*8  time(imaxpnts), data1(isets,imaxpnts), xyData(:,:,:), replot(:,:,:)
  allocatable xyData, replot
  data xydatalegends/'11', '22', '33'/
  allocate (xyData(iaxes, ipnts, isets), replot(iaxes, plotpnts, isets))
  !ch_save = ch_index
  plotolddata = .false.
  do j = 1, isets
    do i = 1, ipnts
      xydata(1, i, j) = time(i)/365.0
      xydata(2, i, j) = data1(j,i)
    end do
  end do
  if (.not. PlotInit) then
    if (.not. OpenWindow) plotolddata = .true.
    if (Openwindow) then
      if( .not. GetWindowConfig(wc) ) stop 'Window Not Open'
      OpenWindow = .false.
    endif
    retcode=GetGraphDefaults($GTXY,xyGraph)
    xyGraph.setGraphMode=.FALSE.
    xyGraph.graphbgcolor = $CIBLACK
    xyGraph.x1 = 20
    xyGraph.y1 = 30
    xyGraph.x2 = 620
    xyGraph.y2 = 430
    xyGraph.title='Yel=Epi Whi=Hypo Mag=Equil'
    retcode = &
  GetMultiDataDefaults (xyGraph, ipnts, xyData, isets, xyDataSets)
    do setLegends=1,isets
  !    xyDataSets(setLegends).PlotLegends = .true.
      xyDataSets(setLegends).title=xyDataLegends(setLegends)
      xyDataSets(setlegends).markertype = $MKNONE
      xyDataSets(setlegends).numPoints = ipnts
      xyDataSets(setLegends).TitleFont = xyGraph.TitleFont
      DataSetColor(setLegends) = xyDataSets(setLegends).linecolor
    end do
    retcode = GetAxisMultiDefaults(xyGraph, isets, xyDataSets, $ATX, &
              $AFLINEAR, xyAxes(1))
      xyAxes(1).title = 'Time(Years)'
      xyAxes(1).lowVal = 0.0
      xyAxes(1).highVal = TotalRuntime
      xyAxes(1).tickColor = 15  !bright white
      if (TotalRuntime .gt. 10.0) then
        xyAxes(1).increment = 2.0
        xyAxes(1).tickratio = 4
        xyAxes(1).numdigits = 0
      elseif (TotalRuntime .gt. 1.0) then
        xyAxes(1).increment = 1.0
        xyAxes(1).tickratio = 2
        xyAxes(1).numdigits = 0
      elseif (TotalRunTime .le. 1.0) then
        xyAxes(1).increment = (xyAxes(1).highVal-xyAxes(1).lowVal)/5.0
        xyAxes(1).tickratio = 2
        xyAxes(1).numdigits = 2
      endif
      xyAxes(1).gridStyle=$GSNONE
      xyAxes(1).gridLineType=$LTNONE
      xyAxes(1).ticktype = $TTOUTSIDE
      xyAxes(1).axisfont = xyAxes(1).titlefont
    retcode = GetAxisMultiDefaults(xyGraph, isets, xyDataSets, $ATY, &
              $AFLINEAR, xyAxes(2))
      xyAxes(2).lowVal = 0.0
      xyAxes(2).highVal = MaxConc
      xyAxes(2).increment = 0.1*(xyAxes(2).highVal-xyAxes(2).lowVal)
      iscale = -1*nint(dlog10(scalefactor))
      write (dummytitle, '(a,i2,a)') '[VOC] (x10^', iscale, ' ug/L)'
      xyAxes(2).title=dummytitle
      xyAxes(2).gridStyle=$GSNONE
      xyAxes(2).gridLineType=$LTNONE
      xyAxes(2).ticktype = $TTOUTSIDE
      xyAxes(2).numdigits = 1
      xyAxes(2).tickratio = 1
      xyAxes(2).axisfont = xyAxes(2).titlefont
    retcode = GetAxisMultiDefaults(xyGraph, isets, xyDataSets, $ATX, &
              $AFLINEAR, xyAxes(3))
      xyAxes(3).title = ''
      xyAxes(3).lowVal=xyAxes(1).lowVal
      xyAxes(3).highVal=xyAxes(1).highVal
      xyAxes(3).increment=xyAxes(1).increment
      xyAxes(3).gridStyle=$GSNONE
      xyAxes(3).gridLineType=$LTNONE
      xyAxes(3).ticktype = $TTOUTSIDE
      xyAxes(3).numdigits = xyAxes(1).numdigits
      xyAxes(3).tickratio = xyAxes(1).tickratio
      xyAxes(3).axisfont = xyAxes(3).titlefont
    retcode = GetAxisMultiDefaults(xyGraph, isets, xyDataSets, $ATY, &
              $AFLINEAR, xyAxes(4))
      xyAxes(4).title=''
      xyAxes(4).lowVal=xyAxes(2).lowVal
      xyAxes(4).highVal=xyAxes(2).highVal
      xyAxes(4).increment=xyAxes(2).increment
      xyAxes(4).gridStyle=$GSNONE
      xyAxes(4).gridLineType=$LTNONE
      xyAxes(4).ticktype = $TTOUTSIDE
      xyAxes(4).numdigits = 2
      xyAxes(4).tickratio = 1
      xyAxes(4).axisfont = xyAxes(4).titlefont
    retcode=PlotGraph(xyGraph, 4, xyAxes, itotalsets)
    PlotInit = .true.
    if (plotolddata) then
      do i = 1, isets
        do j = 1, plotpnts
          replot(1,j,i) = data_save(1,j,i)
          replot(2,j,i) = data_save(2,j,i)
        enddo
      end do
      retcode = GetMultiDataDefaults (xyGraph, plotpnts, replot, isets, &
                OldData)
      do setLegends=1,isets
  !      OldData(setLegends).PlotLegends = .true.
        OldData(setLegends).title=xyDataLegends(setLegends)
        OldData(setlegends).markertype = $MKNONE
        OldData(setlegends).numPoints = plotpnts
        OldData(setLegends).TitleFont = xyGraph.TitleFont
        OldData(setLegends).linecolor = DataSetColor(setLegends)
      enddo
      retcode=PlotMultiData(xyGraph, replot, isets, OldData, xyAxes(1), &
              xyAxes(2))
    else
      retcode = GetMultiDataDefaults (xyGraph, ipnts, xyData, isets, &
                xyDataSets)
      do setLegends=1,isets
        xyDataSets(setLegends).title=xyDataLegends(setLegends)
        xyDataSets(setLegends).titleColor = $CIBLACK
        xyDataSets(setlegends).markertype = $MKNONE
        xyDataSets(setlegends).numPoints = ipnts
        xyDataSets(setLegends).TitleFont = xyGraph.TitleFont
        xyDataSets(setLegends).TitleFont = xyGraph.TitleFont
        xyDataSets(setLegends).lineColor = DataSetColor(setLegends)
      end do
    endif
  else
    retcode = GetMultiDataDefaults (xyGraph, ipnts, xyData, isets, &
              xyDataSets)
    do setLegends=1,isets
  !    xyDataSets(setLegends).PlotLegends = .true.
      xyDataSets(setLegends).title=xyDataLegends(setLegends)
      xyDataSets(setLegends).titleColor = $CIBLACK
      xyDataSets(setlegends).markertype = $MKNONE
      xyDataSets(setlegends).numPoints = ipnts
      xyDataSets(setLegends).TitleFont = xyGraph.TitleFont
      xyDataSets(setLegends).TitleFont = xyGraph.TitleFont
      xyDataSets(setLegends).lineColor = DataSetColor(setLegends)
    end do
  endif
  retcode=PlotMultiData(xyGraph, xyData, isets, xyDataSets, xyAxes(1), xyAxes(2))
  deallocate (xydata, replot)
  return
end subroutine XYPlot


real*8 function maxminfunc(timeseries, code, datlength)
  implicit none
  integer i, code, datlength
  real*8 max, min, timeseries(datlength)
  if (code .gt. 0) then
!   code .gt. 0 and we're finding a maximum
    max = timeseries(1)
    do i = 2, datlength
      if (max .lt. timeseries(i)) max = timeseries(i)
    end do
    maxminfunc = max
    return
  else
!   code .le. 0 and we're finding a minimum
    min = timeseries(1)
    do i = 2, datlength
      if (min .gt. timeseries(i)) min = timeseries(i)
    end do
    maxminfunc = min
    return
  endif
  return
end function maxminfunc
```

MODULES AND COMPILER RESOURCES

```
module mtbecom
  use dialogm
  use modelcom
  use parameters
  use scigraph
  implicit none
!****************************************************************************
!* DESCRIPTION FOR MODULE MTBECOM                                           *
!*                                                                          *
!* This module contains common data structures used for the model and windows*
!* Before you modify any of these names, make sure you change the names of   *
!* the corresponding variables in *all* subroutines.  Modify this module with*
!* care and patience.  DO NOT delete items without ensuring that the program *
!* will recompile and link                                                  *
!****************************************************************************
  record /GraphSettings/ xyGraph
  record /DataSettings/ xyDataSets(isets) !  data sets defined in MODELCOM
  record /DataSettings/ xyTimeSeries    !  data set defined in MODELCOM
  record /AxisSettings/ xyAxes(4)        ! 4 axes: 2 y, 2 x
  record /windowconfig/ wc, textwindow
  integer ts_entry_id, DatFilUnit, ParFilUnit
  character*10 units(12)
  character*25 FileName(NumTimeSeries)
  character($MAXPATH) datadir(NumTimeSeries)
  character*20 temp_dlg(5)
  character*72 Title, comment(2)
  character*90 ctemp
  character*255 msg0, msg1
! temporary storage values used in dialog boxes
  real*8 TempLA,TempTR,TempOT,TempMW,TempSol,TempDiff,TempMV,TempIC,&
         Tempwa0,Tempwa1,Tempwa2,Tempwb0,Tempwb1,Tempwb2,Tempsal,&
         Tempwd0,Tempwd1,Tempwd2,Tempwd3,Tempsola,Tempsolb,&
         TempTol,TempIH,TempOH,TempEpiLoss,TempHypLoss,TempRel_Hum
  real*8 SaveLA,SaveTR,SaveOT,SaveMW,SaveSol,SaveDiff,SaveMV,SaveIC,&
         Savewa0,Savewa1,Savewa2,Savewb0,Savewb1,Savewb2,Savesal,&
         Savewd0,Savewd1,Savewd2,Savewd3,Savesola,Savesolb,&
         SaveTol,SaveIH,SaveOH,SaveEpiLoss,SaveHypLoss,SaveRel_Hum
  integer TempPP, PointsChanged, TempInfChoi
! scalar constants used in dialog boxes and program
  real*8 TotalRuntime,OutputTimeStep,MolWeight,ScVal,HVal,&
         Solubility,Diffusivity,molarVolume,Initial_MTBEConc,MaxConc,&
         wa0, wa1, wa2, wb0, wb1, wb2, salinity, wd0, wd1, wd2, wd3, sola,&
         solb, maxdepth, rel_hum
  integer ProfilePoints, InflowChoice
! Splined arrays used in program.  Splined by routines in INTERP_F.F90
  real*8 spl_LakeDepth(splinepnts), MLD_data(splinepnts),&
         spl_SurfaceTemp(splinepnts), spl_Inflow(splinepnts),&
         spl_Outflow(splinepnts), spl_AirTemp(splinepnts), &
         spl_WindSpeed(splinepnts), spl_AtmosPress(splinepnts), &
         spl_MTBEInput(splinepnts), spl_AtmMTBEConc(splinepnts), &
         spl_EpiVol(splinepnts), spl_HypVol(splinepnts),&
         spl_InflowHeight(splinepnts), spl_OutflowHeight(splinepnts),&
         spl_EpiLossRate(splinepnts), spl_HypLossRate(splinepnts)
! array variables.  Most are splined by subroutines in INTERP_F.F90
  real*8 LakeDepth(:), MixedLayer(:), SurfaceTemp(:), Inflow(:), Outflow(:),&
         LakeArea(:,:), AirTemp(:), WindSpeed(:), AtmosPress(:),&
         MTBEInput(:), AtmMTBEConc(:), TempArea(:,:), InflowHeight(:),&
         OutflowHeight(:), EpiLossRate(:), HypLossRate(:),&
         year_time(splinepnts)
  allocatable LakeDepth, MixedLayer, SurfaceTemp, Inflow, Outflow, LakeArea,&
              TempArea, AirTemp, WindSpeed, AtmosPress, MTBEInput,&
              AtmMTBEConc, InflowHeight, OutflowHeight, EpiLossRate,&
              HypLossRate
! MLD_DATA and YEAR_TIME are used for getting smooth function for MLD and MLD'
! HypConc_Last and HypConc_Curr are used to reset hypolimnion concentration
  logical LRunning, DatOut, ParOut, OpenWindow, MenuActive, MLD_setyet
  type(dialog) dlg_save
  integer diffparam, solparam, savediffparam, savesolparam
! default setup data and parameters.  Identical copy in RESETPAR.F90
  data title /'Lake Perris MTBE Data; Atmospheric Equilibrium; No Boat Input'/
  data comment(1) /'Default Model Data Set-Comment #1'/
  data comment(2) /'Default Model Data Set-Comment #2'/
  data ProfilePoints, PointsChanged/2,0/
  data TotalRuntime,OutputTimestep,Initial_MTBEConc/2.0,1.0,0.20/
  data sola,solb,wa0,wa1,wa2,wb0,wb1,wb2,salinity,wd0,wd1,wd2,wd3,rel_hum&
      /18.4,7666.0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.7/
  data Diffusivity,Solubility,MolWeight,MolarVolume /1.2e-05,1.0,88.15,129.4/
  data LRunning, DatOut, ParOut, MenuActive, MLD_setyet&
       /.FALSE., .false., .true., .false.,    .false./
  data DatFilUnit, ParFilUnit, diffparam, solparam /15, 16, 1, 1/
  data datadir/'','','','','','','','','','','','','',''/
  data FileName/'','','','','','','','','','','','','',''/
end module mtbecom


module parameters
  implicit none
  integer splinepnts, NumTimeSeries, iaxes, imaxpnts, isets,&
          maxoldpts, num_eqs
  integer indexTW, indexMLD, indexLD, indexIN, indexOUT, indexTA, indexU,&
          indexPA, indexMTBE, indexAirMTBE, indexINHe, indexOUTHe,&
          indexEpiL, indexHypL
  integer startHydro, endHydro, startAtm, endAtm, startVOC, endVOC
  parameter (num_eqs=5)
  parameter (splinepnts = 365)
  parameter (NumTimeSeries = 14)
  parameter (iaxes=2, imaxpnts=10, isets=3, maxoldpts=10000)
  parameter (indexTW=1, indexLD=2, indexMLD=3, indexIN=4, indexOUT=5,&
             indexINHe=6, indexOUTHe=7, indexTA=8, indexU=9, indexPA=10,&
             indexMTBE=11, indexAirMTBE=12, indexEpiL=13, indexHypL=14)
  parameter (startHydro=1, endHydro=7, startAtm=8, endAtm=10, startVOC=11,&
             endVOC=14)
  real*8 pi
  parameter (pi=3.141592856)
end module parameters


module inputinfo
! this module contains the variables involved with the various input files
  character(len=255) parfile_out, parfile_in, datfile_out
  logical Par_File_Read, Par_File_Sav, MTBE_File_Sav
  data Par_File_Sav /.true./
end module inputinfo


module parcom
  implicit none
  real*8 LD_temp(365)
  real*8 md
end module parcom


module graphcom
  use parameters
  integer graph_id
  real*8 GraphSeries(365)
  character*20 graphtitle(NumTimeSeries)
  character*15 axistitle(NumTimeSeries)
  character*25 gfile
  character*65 gstatus
  data graphtitle/'Epilimnion Temp.', 'Lake Depth', 'Mixed-Layer Depth',&
                  'Inflow Volume', 'Outflow Volume', 'Inflow Height',&
                  'Outflow Height', 'Air Temperature', 'Wind Speed', &
                  'Atmos. Press.', 'Epilim. VOC Input', 'Atmos. VOC Conc.',&
                  'Epilim. Degrad. Rate', 'Hypolim. Deg. Rate'/
  data axistitle/'deg-C', 'meters', 'meters', 'meters^3/day', 'meters^3/day',&
                 'meters', 'meters', 'deg-C', 'meters/second', 'Atmospheres',&
                 'kg/day', 'ppbv', 'day^-1', 'day^-1'/
end module graphcom


module errorcom
!  Character*10 units(12)
  Character*72 err_str(13)
  logical winfunc, errorwindow, err_dlg(13)
  integer ErrWinUnit
  data errwinunit /17/
  parameter (winfunc = .true.)
end module errorcom


module diffsolcom
  integer TempDiffParam, TempSolParam
end module diffsolcom


//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS
```

```
/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_MTBEParams DIALOG DISCARDABLE  0, 0, 234, 294
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "VOC Concentrations/Inputs"
FONT 10, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Enter Data",IDC_MTBEInputSeries,154,19,36,14
    PUSHBUTTON      "Enter Data",IDC_AtmMTBEConc,154,67,36,14
    PUSHBUTTON      "Enter Data",IDC_EpiLossRate,154,118,36,14
    PUSHBUTTON      "Enter Data",IDC_HypLossRate,154,165,36,14
    PUSHBUTTON      "Modify D",IDC_CallDiffParam,28,206,50,15
    PUSHBUTTON      "Modify H",IDC_CallSolParam,143,207,50,15
    EDITTEXT        IDC_MolWeight,15,249,44,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_InitialConc,131,249,44,12,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,39,273,40,14
    PUSHBUTTON      "Cancel",IDCANCEL,150,273,40,14
    GROUPBOX        "VOC Input Time Series",IDC_STATIC,24,3,171,45
    CTEXT           "Current",IDC_STATIC,31,13,33,9
    CTEXT           "Data",IDC_STATIC,31,22,33,9
    CTEXT           "Monthly",IDC_MTBEOK3,31,31,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Data",IDC_STATIC,77,13,24,8
    CTEXT           "Needed",IDC_STATIC,77,22,24,8
    CTEXT           "Monthly",IDC_MTBEInputMonthly1,73,31,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data Entry",IDC_STATIC,113,13,36,8
    CTEXT           "Needed",IDC_STATIC,119,22,24,8
    CTEXT           "No",IDC_MTBEInputWeekly1,115,31,33,10,SS_SUNKEN |
                    WS_BORDER
    GROUPBOX        "Atm. VOC Concs. Time Series",IDC_STATIC,24,51,171,45
    CTEXT           "Current",IDC_STATIC,31,61,33,9
    CTEXT           "Data",IDC_STATIC,31,70,33,9
    CTEXT           "Monthly",IDC_AirMTBEOK3,31,79,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data",IDC_STATIC,77,61,24,8
    CTEXT           "Needed",IDC_STATIC,77,70,24,8
    CTEXT           "Monthly",IDC_AtmMTBEMonthly1,73,79,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data Entry",IDC_STATIC,113,61,36,8
    CTEXT           "Needed",IDC_STATIC,119,70,24,8
    CTEXT           "No",IDC_AtmMTBEWeekly1,115,79,33,10,SS_SUNKEN |
                    WS_BORDER
    GROUPBOX        "Diffusivity Parameterization",IDC_STATIC,6,195,105,33
    GROUPBOX        "Solubility Parameterization",IDC_STATIC,122,195,105,33
    GROUPBOX        "Molecular Weight of VOC",IDC_STATIC,6,234,105,33
    GROUPBOX        "Initial Concentration of VOC",IDC_STATIC,122,234,105,33
    LTEXT           "g/mole",IDC_STATIC,63,249,20,10
    LTEXT           "ug/L",IDC_STATIC,179,249,20,10
    GROUPBOX        "Degradation Rate in Epilimnion",IDC_STATIC,24,101,171,
                    45
    GROUPBOX        "Degradation Rate in Hypolimnion",IDC_STATIC,24,150,172,
                    42
    CTEXT           "Current",IDC_STATIC,30,112,33,9
    CTEXT           "Data",IDC_STATIC,30,121,33,9
    CTEXT           "Monthly",IDC_EpiLossOK3,30,130,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data",IDC_STATIC,76,112,24,8
    CTEXT           "Needed",IDC_STATIC,76,121,24,8
    CTEXT           "Monthly",IDC_EpiLossMonthly1,72,130,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data Entry",IDC_STATIC,112,112,36,8
    CTEXT           "Needed",IDC_STATIC,118,121,24,8
    CTEXT           "No",IDC_EpiLossWeekly1,114,130,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Current",IDC_STATIC,30,159,33,9
    CTEXT           "Data",IDC_STATIC,30,168,33,9
    CTEXT           "Monthly",IDC_HypLossOK3,30,177,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data",IDC_STATIC,76,159,24,8
    CTEXT           "Needed",IDC_STATIC,76,168,24,8
    CTEXT           "Monthly",IDC_HypLossMonthly1,72,177,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Data Entry",IDC_STATIC,112,159,36,8
    CTEXT           "Needed",IDC_STATIC,118,168,24,8
    CTEXT           "No",IDC_HypLossWeekly1,114,177,33,10,SS_SUNKEN |
                    WS_BORDER
END

IDD_MeteorParams DIALOG DISCARDABLE  0, 0, 186, 208
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Meteorological Parameter Input"
FONT 10, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Enter Data",IDC_AirTemp,136,22,36,14
    PUSHBUTTON      "Enter Data",IDC_WindSpeed,136,72,36,14
    PUSHBUTTON      "Enter Data",IDC_AtmPressure,136,122,36,14
    DEFPUSHBUTTON   "OK",IDOK,18,188,40,14
    PUSHBUTTON      "Cancel",IDCANCEL,123,188,40,14
    GROUPBOX        "Atmospheric Pressure Time Series",IDC_STATIC,6,106,171,
                    45
    GROUPBOX        "Wind Speed Time Series",IDC_STATIC,6,56,171,45
    CTEXT           "Current",IDC_STATIC,13,66,33,9
    CTEXT           "Data",IDC_STATIC,59,66,24,8
    CTEXT           "Data Entry",IDC_STATIC,95,66,36,8
    CTEXT           "Data",IDC_STATIC,13,75,33,9
    CTEXT           "Needed",IDC_STATIC,59,75,24,8
    CTEXT           "Needed",IDC_STATIC,101,75,24,8
    CTEXT           "Monthly",IDC_UOK3,13,84,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_UMonthly1,55,84,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_UWeekly1,97,84,33,10,SS_SUNKEN | WS_BORDER
    GROUPBOX        "Air Temperature Time Series",IDC_STATIC,6,6,171,45
    CTEXT           "Current",IDC_STATIC,13,16,33,9
    CTEXT           "Data",IDC_STATIC,59,16,24,8
    CTEXT           "Data Entry",IDC_STATIC,95,16,36,8
    CTEXT           "Data",IDC_STATIC,13,25,33,9
    CTEXT           "Needed",IDC_STATIC,59,25,24,8
    CTEXT           "Needed",IDC_STATIC,101,25,24,8
    CTEXT           "Monthly",IDC_TAOK3,13,34,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_TAMonthly1,55,34,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_TAWeekly1,97,34,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Current",IDC_STATIC,13,116,33,9
    CTEXT           "Data",IDC_STATIC,59,116,24,8
    CTEXT           "Data Entry",IDC_STATIC,95,116,36,8
    CTEXT           "Data",IDC_STATIC,13,125,33,9
    CTEXT           "Needed",IDC_STATIC,59,125,24,8
    CTEXT           "Needed",IDC_STATIC,101,125,24,8
    CTEXT           "Monthly",IDC_PAOK3,13,134,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_PAMonthly1,55,134,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_PAWeekly1,97,134,33,10,SS_SUNKEN | WS_BORDER
    GROUPBOX        "Relative Humidity Input",IDC_STATIC,44,156,96,26
    EDITTEXT        IDC_RelativeHumidity,72,166,40,10,ES_AUTOHSCROLL
    LTEXT           "%",IDC_STATIC,116,168,10,10
END

IDD_TimeSeriesEntry DIALOG DISCARDABLE  0, 0, 154, 297
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Enter Monthly Time Series"
FONT 10, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_JanVal,51,31,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_FebVal,51,51,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_MarVal,51,71,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_AprVal,51,91,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_MayVal,51,111,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_JunVal,51,131,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_JulVal,52,151,64,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_AugVal,51,171,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_SepVal,51,191,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_OctVal,51,211,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_NovVal,51,231,65,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_DecVal,51,251,65,14,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,7,276,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,87,276,50,14
    CTEXT           "January",IDC_STATIC,11,33,35,10
    CTEXT           "February",IDC_STATIC,11,53,35,10
    CTEXT           "March",IDC_STATIC,11,73,35,10
    CTEXT           "April",IDC_STATIC,11,93,35,10
    CTEXT           "May",IDC_STATIC,11,113,35,10
    CTEXT           "June",IDC_STATIC,11,133,35,10
    CTEXT           "July",IDC_STATIC,17,153,25,10
    CTEXT           "August",IDC_STATIC,11,173,35,10
    CTEXT           "September",IDC_STATIC,11,193,35,10
    CTEXT           "October",IDC_STATIC,11,213,35,10
    CTEXT           "November",IDC_STATIC,11,233,35,10
    CTEXT           "December",IDC_STATIC,11,253,35,10
    CTEXT           " units",IDC_JanUnits,119,33,28,10
    CTEXT           "units",IDC_FebUnits,117,53,30,10
    CTEXT           "units",IDC_MarUnits,117,73,30,10
    CTEXT           "units",IDC_AprUnits,117,93,30,10
    CTEXT           "units",IDC_MayUnits,117,113,30,10
    CTEXT           "units",IDC_JunUnits,117,133,30,10
    CTEXT           "units",IDC_JulUnits,117,153,30,10
    CTEXT           "units",IDC_AugUnits,117,173,30,10
    CTEXT           "units",IDC_SepUnits,117,192,30,10
    CTEXT           "units",IDC_OctUnits,117,212,30,10
    CTEXT           "units",IDC_NovUnits,117,232,30,10
    CTEXT           "units",IDC_DecUnits,117,252,30,10
    CTEXT           "Static",IDC_MonthlyTitle,7,13,140,11,SS_SUNKEN |
                    WS_BORDER
END

IDD_RuntimeParams DIALOG DISCARDABLE  0, 0, 246, 213
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Runtime Parameters"
FONT 10, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_TotalTime,40,20,55,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_OutputTimestep,40,55,60,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_Tolerance,37,93,60,14,ES_AUTOHSCROLL
    EDITTEXT        IDC_Title,10,125,225,10,ES_AUTOHSCROLL
    EDITTEXT        IDC_Comment1,10,155,225,10,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,10,190,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,185,190,50,14
    LTEXT           "Total Simulation Time",IDC_STATIC,38,6,65,8
    LTEXT           "Years",IDC_STATIC,100,20,18,8
    LTEXT           "Data Output Time Step",IDC_STATIC,33,42,74,8
    LTEXT           "Days",IDC_STATIC,105,58,18,8
    LTEXT           "Runge-Kutta Tolerance",IDC_STATIC,30,80,74,8
    LTEXT           "Simulation Title",IDC_STATIC,10,115,50,10
    LTEXT           "Comments",IDC_STATIC,10,140,35,10
    EDITTEXT        IDC_Comment2,10,170,225,10,ES_AUTOHSCROLL
END

IDD_ResetParams DIALOG DISCARDABLE  0, 0, 119, 60
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Reset Parameters to Default"
FONT 10, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Yes",IDOK,10,35,25,14,BS_CENTER | BS_VCENTER
    PUSHBUTTON      "No",IDCANCEL,85,35,25,14
    CTEXT           "Are you sure you want to reset all parameters to default &
                     values?",
                    IDC_STATIC,10,10,100,15
END

IDD_HydrogParams DIALOG DISCARDABLE  0, 0, 362, 191
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Hydrographical Parameters"
FONT 10, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Enter Data",IDC_SurfaceTemp,136,19,36,14
    PUSHBUTTON      "Enter Data",IDC_MixedLayer,310,19,36,14
    PUSHBUTTON      "Enter Data",IDC_Inflow,136,78,36,10
    PUSHBUTTON      "Enter Data",IDC_InflowHeight,136,97,36,10
    PUSHBUTTON      "Enter Data",IDC_Outflow,310,78,36,10
    PUSHBUTTON      "Enter Data",IDC_OutflowHeight,310,97,36,10
    PUSHBUTTON      "Enter Data",IDC_LakeDepth,136,135,36,14
    EDITTEXT        IDC_ProfilePoints,193,146,39,12,ES_AUTOHSCROLL
    PUSHBUTTON      "Enter Profile",IDC_LakeArea,247,146,42,12
    PUSHBUTTON      "View Profile",IDC_LakeArea2,301,146,42,12
    DEFPUSHBUTTON   "OK",IDOK,64,169,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,247,169,50,14
    GROUPBOX        "Epilimnion Temperature Time Series",IDC_STATIC,6,3,171,
                    45
    CTEXT           "Current",IDC_STATIC,13,13,33,8
    CTEXT           "Data",IDC_STATIC,59,13,24,8
    CTEXT           "Data Entry",IDC_STATIC,95,13,36,8
    CTEXT           "Data",IDC_STATIC,13,22,33,8
    CTEXT           "Needed",IDC_STATIC,59,22,24,8
    CTEXT           "Required",IDC_STATIC,97,22,30,8
    CTEXT           "Monthly",IDC_TWOK3,13,31,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_TWMonthly1,55,31,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_TWWeekly1,97,31,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Current",IDC_STATIC,187,13,33,8
    CTEXT           "Data",IDC_STATIC,233,13,24,8
    CTEXT           "Data Entry",IDC_STATIC,269,13,36,8
    CTEXT           "Data",IDC_STATIC,187,22,33,8
    CTEXT           "Needed",IDC_STATIC,233,22,24,8
    CTEXT           "Monthly",IDC_MLDOK3,187,31,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_MLDMonthly1,229,31,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_MLDWeekly1,271,31,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Current",IDC_STATIC,8,62,33,8
    CTEXT           "Data",IDC_STATIC,56,62,24,8
    CTEXT           "Data Entry",IDC_STATIC,92,62,36,8
    CTEXT           "Data",IDC_STATIC,8,69,33,8
    CTEXT           "Needed",IDC_STATIC,56,69,24,8
    CTEXT           "Monthly",IDC_InflowOK3,10,78,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Monthly",IDC_InflowMonthly1,52,78,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_InflowWeekly1,94,78,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Current",IDC_STATIC,187,62,33,8
    CTEXT           "Data",IDC_STATIC,233,62,24,8
    CTEXT           "Data Entry",IDC_STATIC,269,62,36,8
    CTEXT           "Data",IDC_STATIC,187,69,33,8
    CTEXT           "Needed",IDC_STATIC,233,69,24,8
    CTEXT           "Monthly",IDC_OutflowOK3,187,78,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Monthly",IDC_OutflowMonthly1,229,78,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_OutflowWeekly1,271,78,33,10,SS_SUNKEN |
                    WS_BORDER
    GROUPBOX        "Lake Depth Time Series",IDC_STATIC,7,119,171,45
    GROUPBOX        "Epilimnion Depth Time Series",IDC_STATIC,181,3,171,45
    GROUPBOX        "Lake Inflow Time Series",IDC_STATIC,7,52,171,62
    GROUPBOX        "Lake Outflow Time Series",IDC_STATIC,181,52,171,62
    GROUPBOX        "Lake Surface Area Data",IDC_STATIC,181,119,171,45
    CTEXT           "Current",IDC_STATIC,10,129,33,9
    CTEXT           "Data",IDC_STATIC,56,129,24,8
    CTEXT           "Data Entry",IDC_STATIC,92,128,36,8
    CTEXT           "Data",IDC_STATIC,10,137,33,9
    CTEXT           "Needed",IDC_STATIC,56,137,24,8
    CTEXT           "Monthly",IDC_LDOK3,10,146,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Monthly",IDC_LDMonthly1,52,146,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_LDWeekly1,94,146,33,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Required",IDC_STATIC,94,137,30,8
    CTEXT           "Required",IDC_STATIC,94,69,30,8
    CTEXT           "Required",IDC_STATIC,271,69,30,8
    CTEXT           "Required",IDC_STATIC,271,22,30,8
    LTEXT           "Number of Points          in Profile",IDC_STATIC,187,
                    128,54,15
    CTEXT           "Monthly",IDC_InflowHeightOK3,10,97,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Monthly",IDC_InflowHeightMonthly1,52,97,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "No",IDC_InflowHeightWeekly1,94,97,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Volume",IDC_STATIC,136,69,35,8
    CTEXT           "Height",IDC_STATIC,137,89,35,8
    CTEXT           "Monthly",IDC_OutflowHeightOK3,187,97,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Monthly",IDC_OutflowHeightMonthly1,229,97,33,10,
                    SS_SUNKEN | WS_BORDER
    CTEXT           "No",IDC_OutflowHeightWeekly1,271,97,33,10,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Volume",IDC_STATIC,308,69,35,8
    CTEXT           "Height",IDC_STATIC,309,89,35,8
END

IDD_DiffParam DIALOG DISCARDABLE  0, 0, 249, 165
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Diffusivity Parameterization"
FONT 10, "MS Sans Serif"
BEGIN
    CONTROL         "Use:",IDC_DiffButtWilk,"Button",BS_AUTORADIOBUTTON,14,
                    26,25,10
    CONTROL         "Use:",IDC_DiffButtWann,"Button",BS_AUTORADIOBUTTON,15,
                    86,24,10
    EDITTEXT        IDC_MolarVolume,54,52,50,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a0,19,117,41,15,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a1,70,117,45,15,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a2,130,117,42,15,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a3,185,117,44,15,ES_AUTOHSCROLL
    EDITTEXT        IDC_ScDay,204,18,30,10,ES_CENTER | ES_AUTOHSCROLL
    PUSHBUTTON      "Recalculate",IDC_CalcSc,188,52,42,12
    DEFPUSHBUTTON   "OK",IDOK,10,143,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,190,143,50,14
    LTEXT           "D = 4.72x10^-7*T/(mu*V)     (D in cm^2/sec)",IDC_STATIC,
                    39,27,133,10
    CTEXT           "Sc=d0 + d1*T + d2*T^2 + d3*T^3   (Sc:  Schmidt Number = nu/D)",
                    IDC_STATIC,39,87,193,10
    GROUPBOX        "Wilke-Chang Diffusivity Parameterization",IDC_STATIC,
                    10,5,164,65
    GROUPBOX        "Wanninkhof Diffusivity Parameterization, JGR  97C:  7373-7382
                    (1992)",
                    IDC_STATIC,10,75,230,63
    LTEXT           "A",IDC_STATIC,35,107,8,8
    LTEXT           "B",IDC_STATIC,92,107,8,8
    LTEXT           "C",IDC_STATIC,149,107,8,8
    LTEXT           "D",IDC_STATIC,204,107,8,8
    LTEXT           "AIChEJ, 20: 611-615 (1955)",IDC_STATIC,26,14,90,8
    CTEXT           "(nu: kinematic viscosity)",IDC_STATIC,130,97,93,10
    LTEXT           "V (Molar Volume in ml/mole)",IDC_STATIC,39,40,87,10
    GROUPBOX        "Schmidt Number",IDC_STATIC,180,5,60,65
    CTEXT           "Static",IDC_ScVal,204,34,30,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Time",IDC_STATIC,184,20,16,8
    CTEXT           "Sc",IDC_STATIC,188,34,9,8,SS_CENTERIMAGE
END

IDD_SolParam DIALOG DISCARDABLE  0, 0, 250, 194
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Solubility Parameterization"
```

```
FONT 10, "MS Sans Serif"
BEGIN
    CONTROL         "Use:",IDC_SolButtRobbins,"Button",BS_AUTORADIOBUTTON,15,
                    25,25,10
    CONTROL         "Use:",IDC_SolButtWann,"Button",BS_AUTORADIOBUTTON,15,90,
                    25,10
    LTEXT           "A",IDC_STATIC,51,44,8,8
    EDITTEXT        IDC_RobbinsA,30,53,47,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_RobbinsB,85,53,48,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a0_sol,15,128,41,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a1_sol,75,128,41,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_a2_sol,126,128,42,12,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,13,171,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,193,171,50,14
    GROUPBOX        "Robbins et al. solubility parameterization",IDC_STATIC,
                    10,5,140,65
    LTEXT           "H = exp(A-B/T)\n(H in atm-m^3/mol; T in deg-K)",
                    IDC_STATIC,40,26,95,16
    LTEXT           "B",IDC_STATIC,105,44,8,8
    GROUPBOX        "Wanninkhof Solubility Parameterization, JGR  97C:  7373-7382
                    (1992)",
                    IDC_STATIC,10,75,230,92
    LTEXT           "A1",IDC_STATIC,28,120,9,8
    LTEXT           "A2",IDC_STATIC,91,120,9,8
    LTEXT           "A3",IDC_STATIC,144,120,9,8
    LTEXT           "ln(Alpha) = a0+a1*(100/T)+a2*ln(T/100) + \n
                     Salinity*(B1+B2*(T/100)+B3*(T/100)^2)\n
                     (Alpha=Ostwald Solubility; T in deg-K)",
                    IDC_STATIC,40,91,150,25
    EDITTEXT        IDC_Wank_b0_sol,14,151,41,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_b1_sol,74,151,41,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_Wank_b2_sol,125,151,42,12,ES_AUTOHSCROLL
    LTEXT           "B1",IDC_STATIC,28,144,9,8
    LTEXT           "B2",IDC_STATIC,90,144,9,8
    LTEXT           "B3",IDC_STATIC,144,144,9,8
    EDITTEXT        IDC_Wank_Salinity,183,141,42,12,ES_AUTOHSCROLL
    LTEXT           "Salinity (o/oo)",IDC_STATIC,183,130,42,8
    CTEXT           "Anal. Chem.  65: 3113-3118 (1993)",IDC_STATIC,22,14,106,
                    6,SS_CENTERIMAGE
    EDITTEXT        IDC_HDay,192,17,30,10,ES_CENTER | ES_AUTOHSCROLL
    PUSHBUTTON      "Recalculate",IDC_CalcH,182,56,42,12
    GROUPBOX        "Solubility, KH",IDC_STATIC,168,6,70,65
    CTEXT           "Static",IDC_HVal,184,31,48,10,SS_SUNKEN | WS_BORDER
    CTEXT           "Time",IDC_STATIC,171,18,16,8
    CTEXT           "KH",IDC_STATIC,172,32,12,8,SS_CENTERIMAGE
    CTEXT           "mol / m^3-atm",IDC_STATIC,182,44,50,8,SS_CENTERIMAGE
END

IDD_RuntimeMTBEParams DIALOG DISCARDABLE  0, 0, 129, 125
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Runtime VOC Parameter Changes"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Enter Series",IDC_RuntimeMTBEInputSeries,35,20,50,14
    PUSHBUTTON      "Enter Series",IDC_RuntimeAtmMTBEConc,35,70,50,14
    DEFPUSHBUTTON   "OK",IDOK,10,100,30,14
    PUSHBUTTON      "Cancel",IDCANCEL,90,100,30,14
    LTEXT           "Monthly Averaged VOC Inputs",IDC_STATIC,20,5,90,10
    CTEXT           "Avg. Monthly Atm. VOC Concs.",IDC_STATIC,20,50,95,10
END
```

```
IDD_TimeSeriesSetup DIALOG DISCARDABLE  0, 0, 329, 298
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Time Series Setup"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,67,275,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,202,275,50,14
    LTEXT           "Water Temperature",IDC_STATIC,11,31,67,8
    CTEXT           "Monthly",IDC_TWOK2,231,30,50,9,WS_BORDER
    CTEXT           "OK",IDC_TWOK,296,31,15,9,WS_BORDER
    LTEXT           "Mixed Layer Depth",IDC_STATIC,11,46,57,8
    CTEXT           "Monthly",IDC_MLDOK2,231,45,50,9,WS_BORDER
    CTEXT           "OK",IDC_MLDOK,296,46,15,9,WS_BORDER
    LTEXT           "Total Lake Depth",IDC_STATIC,11,61,57,8
    CTEXT           "Monthly",IDC_LDOK2,231,60,50,9,WS_BORDER
    CTEXT           "OK",IDC_LDOK,296,61,15,9,WS_BORDER
    LTEXT           "Lake Inflow Volume",IDC_STATIC,11,75,63,8
    CTEXT           "Monthly",IDC_InflowOK2,231,75,50,9,WS_BORDER
    CTEXT           "OK",IDC_InflowOK,296,76,15,9,WS_BORDER
    LTEXT           "Lake Outflow Volume",IDC_STATIC,11,102,67,8
    CTEXT           "Monthly",IDC_OutflowOK2,231,101,50,9,WS_BORDER
    CTEXT           "OK",IDC_OutflowOK,296,102,15,9,WS_BORDER
    LTEXT           "Air Temperature",IDC_STATIC,12,147,55,8
    CTEXT           "Monthly",IDC_TAOK2,232,146,50,9,WS_BORDER
    CTEXT           "OK",IDC_TAOK,297,147,15,9,WS_BORDER
    LTEXT           "Wind Speed",IDC_STATIC,12,162,50,8
    CTEXT           "Monthly",IDC_UOK2,232,161,50,9,WS_BORDER
    CTEXT           "OK",IDC_UOK,297,162,15,9,WS_BORDER
    LTEXT           "Atm. Pressure",IDC_STATIC,12,177,50,8
    CTEXT           "Monthly",IDC_PAOK2,232,176,50,9,WS_BORDER
    CTEXT           "OK",IDC_PAOK,297,177,15,9,WS_BORDER
    LTEXT           "Direct VOC Input",IDC_STATIC,12,209,55,8
    CTEXT           "Monthly",IDC_MTBEOK2,232,208,50,9,WS_BORDER
    CTEXT           "OK",IDC_MTBEOK,297,208,15,9,WS_BORDER
    LTEXT           "Atm. VOC Conc.",IDC_STATIC,12,223,55,8
    CTEXT           "Monthly",IDC_AirMTBEOK2,232,222,50,9,WS_BORDER
    CTEXT           "Select Time Series Grid Step",IDC_STATIC,99,7,95,12,
                    SS_CENTERIMAGE | SS_SUNKEN | WS_BORDER
    CTEXT           "OK",IDC_AirMTBEOK,297,222,15,9,WS_BORDER
    CTEXT           "Time Series",IDC_STATIC,19,7,40,13,SS_CENTERIMAGE |
                    SS_SUNKEN | WS_BORDER
    CTEXT           "Status",IDC_STATIC,290,7,25,12,SS_CENTERIMAGE |
                    SS_SUNKEN | WS_BORDER
    GROUPBOX        "Hydrographical Parameters",IDC_STATIC,7,20,315,110
    GROUPBOX        "Meteorological Parameters",IDC_STATIC,7,136,315,55
    GROUPBOX        "VOC Parameters",IDC_STATIC,7,198,315,71
    CTEXT           "Series Data",IDC_STATIC,229,7,50,13,SS_CENTERIMAGE |
                    SS_SUNKEN | WS_BORDER
    LTEXT           "Lake Outflow Height",IDC_STATIC,11,115,67,8
    CTEXT           "Monthly",IDC_OutflowHeightOK2,231,114,50,9,WS_BORDER
    CTEXT           "OK",IDC_OutflowHeightOK,296,115,15,9,WS_BORDER
    LTEXT           "Lake Inflow Height",IDC_STATIC,11,89,63,8
    CTEXT           "Monthly",IDC_InflowHeightOK2,231,88,50,9,WS_BORDER
    CTEXT           "OK",IDC_InflowHeightOK,296,89,15,9,WS_BORDER
    CTEXT           "Epilimnion Loss Rate",IDC_STATIC,13,237,66,8,
                    SS_CENTERIMAGE
    CTEXT           "Monthly",IDC_EpiLossOK2,233,238,50,9,WS_BORDER
    CTEXT           "OK",IDC_EpiLossOK,298,238,15,9,WS_BORDER
    CTEXT           "Hypolimnion Loss Rate",IDC_STATIC,7,253,79,8,
                    SS_CENTERIMAGE
```

```
CTEXT               "Monthly",IDC_HypLossOK2,233,254,50,9,WS_BORDER
CTEXT               "OK",IDC_HypLossOK,298,254,15,9,WS_BORDER
CONTROL             "Monthly",IDC_TWMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,30,32,10
CONTROL             "Weekly",IDC_TWWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,30,32,10
CONTROL             "Daily",IDC_TWDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,30,32,10
CONTROL             "Monthly",IDC_MLDMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,45,32,10
CONTROL             "Weekly",IDC_MLDWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,45,32,10
CONTROL             "Daily",IDC_MLDDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,45,32,10
CONTROL             "Monthly",IDC_LDMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,60,32,10
CONTROL             "Weekly",IDC_LDWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,60,32,10
CONTROL             "Daily",IDC_LDDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,60,32,10
CONTROL             "Monthly",IDC_InflowMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,75,32,10
CONTROL             "Weekly",IDC_InflowWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,75,32,10
CONTROL             "Daily",IDC_InflowDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,75,32,10
CONTROL             "Monthly",IDC_InflowHeightMonthly,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,86,88,32,10
CONTROL             "Weekly",IDC_InflowHeightWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,88,32,10
CONTROL             "Daily",IDC_InflowHeightDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,88,32,10
CONTROL             "Monthly",IDC_OutflowMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,102,32,10
CONTROL             "Weekly",IDC_OutflowWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,102,32,10
CONTROL             "Daily",IDC_OutflowDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,102,32,10
CONTROL             "Monthly",IDC_OutflowHeightMonthly,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,86,114,32,10
CONTROL             "Weekly",IDC_OutflowHeightWeekly,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,135,114,32,10
CONTROL             "Daily",IDC_OutflowHeightDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,114,32,10
CONTROL             "Monthly",IDC_TAMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,146,32,10
CONTROL             "Weekly",IDC_TAWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,146,32,10
CONTROL             "Daily",IDC_TADaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,146,32,10
CONTROL             "Monthly",IDC_UMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,161,32,10
CONTROL             "Weekly",IDC_UWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,161,32,10
CONTROL             "Daily",IDC_UDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,161,32,10
CONTROL             "Monthly",IDC_PAMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,176,32,10
CONTROL             "Weekly",IDC_PAWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,176,32,10
```

```
    CONTROL         "Daily",IDC_PADaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,176,32,10
    CONTROL         "Monthly",IDC_MTBEMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,209,32,10
    CONTROL         "Weekly",IDC_MTBEWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,209,32,10
    CONTROL         "Daily",IDC_MTBEDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,209,32,10
    CONTROL         "Monthly",IDC_AtmMTBEMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,223,32,10
    CONTROL         "Weekly",IDC_AtmMTBEWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,223,32,10
    CONTROL         "Daily",IDC_AtmMTBEDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,223,32,10
    CONTROL         "Monthly",IDC_EpiLossMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,236,32,10
    CONTROL         "Weekly",IDC_EpiLossWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,236,32,10
    CONTROL         "Daily",IDC_EpiLossDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,236,32,10
    CONTROL         "Monthly",IDC_HypLossMonthly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,86,252,32,10
    CONTROL         "Weekly",IDC_HypLossWeekly,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,135,252,32,10
    CONTROL         "Daily",IDC_HypLossDaily,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,184,252,32,10
END

IDD_TimeSeriesFileEntry DIALOG DISCARDABLE  0, 0, 258, 251
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Time Series Data File Entry"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_CurrentWorkDir,12,33,234,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_DataDirectory,12,69,234,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_TimeSeriesFilename,12,105,99,12,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,6,228,50,14
    PUSHBUTTON      "Load Data File",IDC_LoadData,71,228,50,14
    PUSHBUTTON      "Graph Data",IDC_GraphData,136,228,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,201,228,50,14
    GROUPBOX        "Current Working Directory",IDC_STATIC,6,21,246,31
    GROUPBOX        "Data Directory",IDC_STATIC,6,57,246,31
    GROUPBOX        "Data Filename",IDC_STATIC,6,93,117,31
    GROUPBOX        "File Status",IDC_STATIC,135,93,117,31
    CTEXT           "Static",IDC_FileStatus,144,105,102,12,SS_SUNKEN |
                    WS_BORDER
    GROUPBOX        "File Input Errors",IDC_STATIC,7,170,246,50
    CTEXT           "Static",IDC_FileStatus2,13,185,234,27,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Time Series Being Entered",IDC_LocalTitle,6,6,246,12,
                    SS_SUNKEN | WS_BORDER
    GROUPBOX        "Data units needed by model:",IDC_STATIC,7,132,246,33
    CTEXT           "Static",IDC_DataUnits,13,147,234,12,SS_SUNKEN |
                    WS_BORDER
END

IDD_LakeDepthProfileEntry DIALOG DISCARDABLE  0, 0, 177, 203
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Lake Area/Depth Profile Entry"
FONT 8, "MS Sans Serif"
```

```
BEGIN
    COMBOBOX        IDC_LakeDepthProfile,21,7,129,99,CBS_DROPDOWN |
                    WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "Enter Point",IDC_EnterPoint,59,50,45,12
    DEFPUSHBUTTON   "OK",IDOK,14,182,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,110,182,50,14
    LTEXT          "Enter Individual Profile Points in the format:\n
                    Point Depth Area \n
                    e.g.:  3 25.5 1.45e6\n\n
                    Depths must decrease to zero\n
                    Areas must be constant or decrease with depth",
                    IDC_STATIC,16,126,141,49
    CONTROL         "Number of Points Modified/Entered:",IDC_STATIC,"Static",
                    SS_LEFTNOWORDWRAP | SS_CENTERIMAGE | WS_GROUP,16,66,108,
                    10
    CTEXT           "0",IDC_NumPointsChanged,126,65,21,12,SS_SUNKEN |
                    WS_BORDER
    CTEXT           "Click Down Arrow to view/select profile point\n
                     Click Enter Point to save changes to profile",
                    IDC_STATIC,20,25,137,18
    CTEXT           "Static",IDC_LAErrorMessage,16,90,144,28,SS_SUNKEN |
                    WS_BORDER
    LTEXT           "Error Messages",IDC_STATIC,18,80,68,10,SS_CENTERIMAGE
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_MTBEParams, DIALOG
    BEGIN
        LEFTMARGIN, 6
        RIGHTMARGIN, 227
        TOPMARGIN, 7
        BOTTOMMARGIN, 287
    END

    IDD_MeteorParams, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 201
    END

    IDD_TimeSeriesEntry, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 147
        TOPMARGIN, 7
        BOTTOMMARGIN, 290
    END

    IDD_RuntimeParams, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 239
        TOPMARGIN, 7
        BOTTOMMARGIN, 206
    END

    IDD_ResetParams, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 112
        TOPMARGIN, 7
        BOTTOMMARGIN, 53
    END

    IDD_HydrogParams, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 351
        TOPMARGIN, 7
        BOTTOMMARGIN, 184
    END

    IDD_DiffParam, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 242
        TOPMARGIN, 7
        BOTTOMMARGIN, 158
    END

    IDD_SolParam, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 243
        TOPMARGIN, 7
        BOTTOMMARGIN, 187
    END

    IDD_RuntimeMTBEParams, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 122
        TOPMARGIN, 7
        BOTTOMMARGIN, 118
    END

    IDD_TimeSeriesSetup, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 322
        TOPMARGIN, 7
        BOTTOMMARGIN, 291
    END

    IDD_TimeSeriesFileEntry, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 251
        TOPMARGIN, 7
        BOTTOMMARGIN, 244
    END
    IDD_LakeDepthProfileEntry, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 170
        TOPMARGIN, 7
        BOTTOMMARGIN, 196
    END
END
#endif    // APSTUDIO_INVOKED


#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON1               ICON    DISCARDABLE     "icon1.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Dialog Info
//

IDD_LakeDepthProfileEntry DLGINIT
BEGIN
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2031, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2032, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2033, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2034, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2035, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2036, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2037, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2038, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x2039, 0x0020,
    IDC_LakeDepthProfile, 0x403, 4, 0
0x3031, 0x0020,
    0
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "CompanyName", "APL-UW\0"
            VALUE "FileDescription", "lakevoc2_6\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "lakevoc2_6\0"
            VALUE "LegalCopyright", "Copyright © 1999\0"
            VALUE "OriginalFilename", "lakevoc2_6.exe\0"
            VALUE "ProductName", "APL-UW lakevoc2_6\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC

#endif    // English (U.S.) resources
```

```
/////////////////////////////////////////////////////////////////////////////
#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED



!MS$FREEFORM
! Microsoft Developer Studio generated include file.
! Used by main_win.rc
!
      integer, parameter :: IDD_MTBEParams                  = 102
      integer, parameter :: IDD_MeteorParams                = 103
      integer, parameter :: IDD_TimeSeriesEntry             = 105
      integer, parameter :: IDD_RuntimeParams               = 110
      integer, parameter :: IDD_ResetParams                 = 111
      integer, parameter :: IDD_HydrogParams                = 112
      integer, parameter :: IDD_DiffParam                   = 113
      integer, parameter :: IDD_SolParam                    = 114
      integer, parameter :: IDI_ICON1                       = 115
      integer, parameter :: IDD_RuntimeMTBEParams           = 116
      integer, parameter :: IDD_TimeSeriesSetup             = 117
      integer, parameter :: IDD_TimeSeriesFileEntry         = 118
      integer, parameter :: IDD_LakeDepthProfileEntry       = 119
      integer, parameter :: IDC_MTBEInputSeries             = 1014
      integer, parameter :: IDC_WindSpeed                   = 1017
      integer, parameter :: IDC_AirTemp                     = 1018
      integer, parameter :: IDC_AtmPressure                 = 1019
      integer, parameter :: IDC_JanVal                      = 1020
      integer, parameter :: IDC_JunVal                      = 1021
      integer, parameter :: IDC_JulVal                      = 1022
      integer, parameter :: IDC_AugVal                      = 1023
      integer, parameter :: IDC_FebVal                      = 1024
      integer, parameter :: IDC_DecVal                      = 1025
      integer, parameter :: IDC_OctVal                      = 1026
      integer, parameter :: IDC_SepVal                      = 1027
      integer, parameter :: IDC_NovVal                      = 1028
      integer, parameter :: IDC_MarVal                      = 1029
      integer, parameter :: IDC_AprVal                      = 1030
      integer, parameter :: IDC_MayVal                      = 1031
      integer, parameter :: IDC_JanUnits                    = 1032
      integer, parameter :: IDC_AprUnits                    = 1033
      integer, parameter :: IDC_MayUnits                    = 1034
      integer, parameter :: IDC_JunUnits                    = 1035
      integer, parameter :: IDC_AugUnits                    = 1036
      integer, parameter :: IDC_JulUnits                    = 1037
      integer, parameter :: IDC_SepUnits                    = 1038
      integer, parameter :: IDC_NovUnits                    = 1039
      integer, parameter :: IDC_OctUnits                    = 1040
      integer, parameter :: IDC_DecUnits                    = 1041
      integer, parameter :: IDC_FebUnits                    = 1042
      integer, parameter :: IDC_MarUnits                    = 1043
      integer, parameter :: IDC_TotalTime                   = 1051
      integer, parameter :: IDC_OutputTimestep              = 1052
```

```
integer, parameter :: IDC_Tolerance                   = 1053
integer, parameter :: IDC_Inflow                      = 1054
integer, parameter :: IDC_CallDiffParam               = 1055
integer, parameter :: IDC_MixedLayer                  = 1061
integer, parameter :: IDC_SurfaceTemp                 = 1062
integer, parameter :: IDC_LakeDepth                   = 1063
integer, parameter :: IDC_LakeArea                    = 1064
integer, parameter :: IDC_AtmMTBEConc                 = 1065
integer, parameter :: IDC_MolWeight                   = 1066
integer, parameter :: IDC_InitialConc                 = 1067
integer, parameter :: IDC_Outflow                     = 1068
integer, parameter :: IDC_DiffButtWilk                = 1069
integer, parameter :: IDC_DiffButtWann                = 1070
integer, parameter :: IDC_Wank_a0                     = 1071
integer, parameter :: IDC_Wank_a1                     = 1072
integer, parameter :: IDC_Wank_a2                     = 1073
integer, parameter :: IDC_Wank_a3                     = 1074
integer, parameter :: IDC_MolarVolume                 = 1075
integer, parameter :: IDC_SolButtWann                 = 1076
integer, parameter :: IDC_Wank_a0_sol                 = 1077
integer, parameter :: IDC_Wank_a1_sol                 = 1078
integer, parameter :: IDC_Wank_a2_sol                 = 1079
integer, parameter :: IDC_Wank_a3_sol                 = 1080
integer, parameter :: IDC_Wank_Salinity               = 1080
integer, parameter :: IDC_SolButtRobbins              = 1081
integer, parameter :: IDC_RobbinsA                    = 1082
integer, parameter :: IDC_RobbinsB                    = 1083
integer, parameter :: IDC_CallSolParam                = 1084
integer, parameter :: IDC_Comment1                    = 1085
integer, parameter :: IDC_Title                       = 1087
integer, parameter :: IDC_Comment2                    = 1088
integer, parameter :: IDC_RuntimeAtmMTBEConc          = 1089
integer, parameter :: IDC_MLDMonthly                  = 1090
integer, parameter :: IDC_MLDWeekly                   = 1091
integer, parameter :: IDC_MLDDaily                    = 1092
integer, parameter :: IDC_TWMonthly                   = 1093
integer, parameter :: IDC_TWWeekly                    = 1094
integer, parameter :: IDC_TWDaily                     = 1095
integer, parameter :: IDC_TAMonthly                   = 1096
integer, parameter :: IDC_TAWeekly                    = 1097
integer, parameter :: IDC_TADaily                     = 1098
integer, parameter :: IDC_UMonthly                    = 1099
integer, parameter :: IDC_UWeekly                     = 1100
integer, parameter :: IDC_UDaily                      = 1101
integer, parameter :: IDC_MTBEMonthly                 = 1102
integer, parameter :: IDC_MTBEWeekly                  = 1103
integer, parameter :: IDC_MTBEDaily                   = 1104
integer, parameter :: IDC_AtmMTBEMonthly              = 1105
integer, parameter :: IDC_AtmMTBEWeekly               = 1106
integer, parameter :: IDC_AtmMTBEDaily                = 1107
integer, parameter :: IDC_MLDOK                       = 1108
integer, parameter :: IDC_TWOK                        = 1109
integer, parameter :: IDC_TAOK                        = 1110
integer, parameter :: IDC_UOK                         = 1111
integer, parameter :: IDC_MTBEOK                      = 1112
integer, parameter :: IDC_AirMTBEOK                   = 1113
integer, parameter :: IDC_PAMonthly                   = 1114
integer, parameter :: IDC_PAWeekly                    = 1115
integer, parameter :: IDC_PADaily                     = 1116
integer, parameter :: IDC_PAOK                        = 1117
```

```
      integer, parameter :: IDC_LDMonthly                   = 1118
      integer, parameter :: IDC_LDWeekly                    = 1119
      integer, parameter :: IDC_LDDaily                     = 1120
      integer, parameter :: IDC_LDOK                        = 1121
      integer, parameter :: IDC_InflowMonthly               = 1122
      integer, parameter :: IDC_InflowWeekly                = 1123
      integer, parameter :: IDC_InflowDaily                 = 1124
      integer, parameter :: IDC_InflowOK                    = 1125
      integer, parameter :: IDC_OutflowMonthly              = 1126
      integer, parameter :: IDC_OutflowWeekly               = 1127
      integer, parameter :: IDC_OutflowDaily                = 1128
      integer, parameter :: IDC_OutflowOK                   = 1129
      integer, parameter :: IDC_MLDOK2                      = 1130
      integer, parameter :: IDC_TWOK2                       = 1131
      integer, parameter :: IDC_TAOK2                       = 1132
      integer, parameter :: IDC_UOK2                        = 1133
      integer, parameter :: IDC_MTBEOK2                     = 1134
      integer, parameter :: IDC_AirMTBEOK2                  = 1135
      integer, parameter :: IDC_PAOK2                       = 1136
      integer, parameter :: IDC_LDOK2                       = 1137
      integer, parameter :: IDC_InflowOK2                   = 1138
      integer, parameter :: IDC_OutflowOK2                  = 1139
      integer, parameter :: IDC_OutflowHeightMonthly        = 1140
      integer, parameter :: IDC_OutflowHeightWeekly         = 1141
      integer, parameter :: IDC_OutflowHeightDaily          = 1142
      integer, parameter :: IDC_OutflowHeightOK             = 1143
      integer, parameter :: IDC_PAOK3                       = 1145
      integer, parameter :: IDC_MLDMonthly1                 = 1146
      integer, parameter :: IDC_TWOK3                       = 1147
      integer, parameter :: IDC_TWMonthly1                  = 1148
      integer, parameter :: IDC_MTBEInputMonthly1           = 1149
      integer, parameter :: IDC_TWWeekly1                   = 1150
      integer, parameter :: IDC_RuntimeMTBEInputSeries      = 1151
      integer, parameter :: IDC_AtmMTBEWeekly1              = 1152
      integer, parameter :: IDC_LDMonthly1                  = 1153
      integer, parameter :: IDC_LDWeekly1                   = 1154
      integer, parameter :: IDC_InflowMonthly1              = 1155
      integer, parameter :: IDC_InflowWeekly1               = 1156
      integer, parameter :: IDC_OutflowMonthly1             = 1157
      integer, parameter :: IDC_OutflowWeekly1              = 1158
      integer, parameter :: IDC_OutflowOK3                  = 1161
      integer, parameter :: IDC_LakeArea2                   = 1162
      integer, parameter :: IDC_CurrentWorkDir              = 1163
      integer, parameter :: IDC_DataDirectory               = 1164
      integer, parameter :: IDC_TimeSeriesFilename          = 1165
      integer, parameter :: IDC_FileStatus                  = 1166
      integer, parameter :: IDC_FileStatus2                 = 1167
      integer, parameter :: IDC_AirMTBEOK3                  = 1168
      integer, parameter :: IDC_UMonthly1                   = 1169
      integer, parameter :: IDC_UWeekly1                    = 1170
      integer, parameter :: IDC_TAMonthly1                  = 1171
      integer, parameter :: IDC_UOK3                        = 1172
      integer, parameter :: IDC_TAWeekly1                   = 1173
      integer, parameter :: IDC_TAOK3                       = 1174
      integer, parameter :: IDC_PAWeekly1                   = 1175
      integer, parameter :: IDC_PAMonthly1                  = 1176
      integer, parameter :: IDC_MLDOK3                      = 1177
      integer, parameter :: IDC_LDOK3                       = 1178
      integer, parameter :: IDC_AtmMTBEMonthly1             = 1179
      integer, parameter :: IDC_MLDWeekly1                  = 1180
      integer, parameter :: IDC_MTBEInputWeekly1           = 1181
      integer, parameter :: IDC_InflowOK3                  = 1182
      integer, parameter :: IDC_MTBEOK3                    = 1183
      integer, parameter :: IDC_LocalTitle                 = 1185
      integer, parameter :: IDC_OutflowHeight              = 1186
      integer, parameter :: IDC_InflowHeight               = 1187
      integer, parameter :: IDC_NumPointsChanged           = 1188
      integer, parameter :: IDC_EnterPoint                 = 1191
      integer, parameter :: IDC_LakeDepthProfile           = 1192
      integer, parameter :: IDC_ProfilePoints              = 1193
      integer, parameter :: IDC_InflowHeightMonthly        = 1194
      integer, parameter :: IDC_InflowHeightWeekly         = 1195
      integer, parameter :: IDC_InflowHeightDaily          = 1196
      integer, parameter :: IDC_InflowHeightOK             = 1197
      integer, parameter :: IDC_InflowHeightOK2            = 1198
      integer, parameter :: IDC_InflowHeightWeekly1        = 1199
      integer, parameter :: IDC_InflowHeightMonthly1       = 1200
      integer, parameter :: IDC_InflowHeightOK3            = 1201
      integer, parameter :: IDC_OutflowHeightWeekly1       = 1202
      integer, parameter :: IDC_OutflowHeightMonthly1      = 1203
      integer, parameter :: IDC_OutflowHeightOK3           = 1204
      integer, parameter :: IDC_OutflowHeightOK2           = 1206
      integer, parameter :: IDC_EpiLossRate                = 1207
      integer, parameter :: IDC_EpiLossWeekly1             = 1208
      integer, parameter :: IDC_EpiLossMonthly1            = 1209
      integer, parameter :: IDC_EpiLossOK3                 = 1210
      integer, parameter :: IDC_HypLossRate                = 1211
      integer, parameter :: IDC_HypLossOK3                 = 1212
      integer, parameter :: IDC_HypLossMonthly1            = 1213
      integer, parameter :: IDC_HypLossWeekly1             = 1214
      integer, parameter :: IDC_EpiLossMonthly             = 1215
      integer, parameter :: IDC_EpiLossWeekly              = 1216
      integer, parameter :: IDC_EpiLossDaily               = 1217
      integer, parameter :: IDC_EpiLossOK2                 = 1218
      integer, parameter :: IDC_EpiLossOK                  = 1219
      integer, parameter :: IDC_HypLossMonthly             = 1220
      integer, parameter :: IDC_HypLossWeekly              = 1221
      integer, parameter :: IDC_HypLossDaily               = 1222
      integer, parameter :: IDC_HypLossOK2                 = 1223
      integer, parameter :: IDC_HypLossOK                  = 1224
      integer, parameter :: IDC_Wank_b0_sol                = 1225
      integer, parameter :: IDC_Wank_b1_sol                = 1226
      integer, parameter :: IDC_Wank_b2_sol                = 1227
      integer, parameter :: IDC_DataUnits                  = 1228
      integer, parameter :: IDC_MonthlyTitle               = 1229
      integer, parameter :: IDC_InflowChoice1              = 1230
      integer, parameter :: IDC_InflowChoice2              = 1231
      integer, parameter :: IDC_GraphData                  = 1232
      integer, parameter :: IDC_LoadData                   = 1233
      integer, parameter :: IDC_LAErrorMessage             = 1235
      integer, parameter :: IDC_ScDay                      = 1236
      integer, parameter :: IDC_ScVal                      = 1237
      integer, parameter :: IDC_CalcSc                     = 1238
      integer, parameter :: IDC_HDay                       = 1239
      integer, parameter :: IDC_HVal                       = 1240
      integer, parameter :: IDC_CalcH                      = 1241
      integer, parameter :: IDC_RelativeHumidity           = 1242

//{{NO_DEPENDENCIES}}
```

```
// Microsoft Developer Studio generated include file.
// Used by main_win.rc
//
#define IDD_MTBEParams                  102
#define IDD_MeteorParams                103
#define IDD_TimeSeriesEntry             105
#define IDD_RuntimeParams               110
#define IDD_ResetParams                 111
#define IDD_HydrogParams                112
#define IDD_DiffParam                   113
#define IDD_SolParam                    114
#define IDI_ICON1                       115
#define IDD_RuntimeMTBEParams           116
#define IDD_TimeSeriesSetup             117
#define IDD_TimeSeriesFileEntry         118
#define IDD_LakeDepthProfileEntry       119
#define IDC_MTBEInputSeries             1014
#define IDC_WindSpeed                   1017
#define IDC_AirTemp                     1018
#define IDC_AtmPressure                 1019
#define IDC_JanVal                      1020
#define IDC_JunVal                      1021
#define IDC_JulVal                      1022
#define IDC_AugVal                      1023
#define IDC_FebVal                      1024
#define IDC_DecVal                      1025
#define IDC_OctVal                      1026
#define IDC_SepVal                      1027
#define IDC_NovVal                      1028
#define IDC_MarVal                      1029
#define IDC_AprVal                      1030
#define IDC_MayVal                      1031
#define IDC_JanUnits                    1032
#define IDC_AprUnits                    1033
#define IDC_MayUnits                    1034
#define IDC_JunUnits                    1035
#define IDC_AugUnits                    1036
#define IDC_JulUnits                    1037
#define IDC_SepUnits                    1038
#define IDC_NovUnits                    1039
#define IDC_OctUnits                    1040
#define IDC_DecUnits                    1041
#define IDC_FebUnits                    1042
#define IDC_MarUnits                    1043
#define IDC_TotalTime                   1051
#define IDC_OutputTimestep              1052
#define IDC_Tolerance                   1053
#define IDC_Inflow                      1054
#define IDC_CallDiffParam               1055
#define IDC_MixedLayer                  1061
#define IDC_SurfaceTemp                 1062
#define IDC_LakeDepth                   1063
#define IDC_LakeArea                    1064
#define IDC_AtmMTBEConc                 1065
#define IDC_MolWeight                   1066
#define IDC_InitialConc                 1067
#define IDC_Outflow                     1068
#define IDC_DiffButtWilk                1069
#define IDC_DiffButtWann                1070
#define IDC_Wank_a0                     1071
```

```
#define IDC_Wank_a1                     1072
#define IDC_Wank_a2                     1073
#define IDC_Wank_a3                     1074
#define IDC_MolarVolume                 1075
#define IDC_SolButtWann                 1076
#define IDC_Wank_a0_sol                 1077
#define IDC_Wank_a1_sol                 1078
#define IDC_Wank_a2_sol                 1079
#define IDC_Wank_a3_sol                 1080
#define IDC_Wank_Salinity               1080
#define IDC_SolButtRobbins              1081
#define IDC_RobbinsA                    1082
#define IDC_RobbinsB                    1083
#define IDC_CallSolParam                1084
#define IDC_Comment1                    1085
#define IDC_Title                       1087
#define IDC_Comment2                    1088
#define IDC_RuntimeAtmMTBEConc          1089
#define IDC_MLDMonthly                  1090
#define IDC_MLDWeekly                   1091
#define IDC_MLDDaily                    1092
#define IDC_TWMonthly                   1093
#define IDC_TWWeekly                    1094
#define IDC_TWDaily                     1095
#define IDC_TAMonthly                   1096
#define IDC_TAWeekly                    1097
#define IDC_TADaily                     1098
#define IDC_UMonthly                    1099
#define IDC_UWeekly                     1100
#define IDC_UDaily                      1101
#define IDC_MTBEMonthly                 1102
#define IDC_MTBEWeekly                  1103
#define IDC_MTBEDaily                   1104
#define IDC_AtmMTBEMonthly              1105
#define IDC_AtmMTBEWeekly               1106
#define IDC_AtmMTBEDaily                1107
#define IDC_MLDOK                       1108
#define IDC_TWOK                        1109
#define IDC_TAOK                        1110
#define IDC_UOK                         1111
#define IDC_MTBEOK                      1112
#define IDC_AirMTBEOK                   1113
#define IDC_PAMonthly                   1114
#define IDC_PAWeekly                    1115
#define IDC_PADaily                     1116
#define IDC_PAOK                        1117
#define IDC_LDMonthly                   1118
#define IDC_LDWeekly                    1119
#define IDC_LDDaily                     1120
#define IDC_LDOK                        1121
#define IDC_InflowMonthly               1122
#define IDC_InflowWeekly                1123
#define IDC_InflowDaily                 1124
#define IDC_InflowOK                    1125
#define IDC_OutflowMonthly              1126
#define IDC_OutflowWeekly               1127
#define IDC_OutflowDaily                1128
#define IDC_OutflowOK                   1129
#define IDC_MLDOK2                      1130
#define IDC_TWOK2                       1131
```

```
#define IDC_TAOK2                       1132
#define IDC_UOK2                        1133
#define IDC_MTBEOK2                     1134
#define IDC_AirMTBEOK2                  1135
#define IDC_PAOK2                       1136
#define IDC_LDOK2                       1137
#define IDC_InflowOK2                   1138
#define IDC_OutflowOK2                  1139
#define IDC_OutflowHeightMonthly        1140
#define IDC_OutflowHeightWeekly         1141
#define IDC_OutflowHeightDaily          1142
#define IDC_OutflowHeightOK             1143
#define IDC_PAOK3                       1145
#define IDC_MLDMonthly1                 1146
#define IDC_TWOK3                       1147
#define IDC_TWMonthly1                  1148
#define IDC_MTBEInputMonthly1           1149
#define IDC_TWWeekly1                   1150
#define IDC_RuntimeMTBEInputSeries      1151
#define IDC_AtmMTBEWeekly1              1152
#define IDC_LDMonthly1                  1153
#define IDC_LDWeekly1                   1154
#define IDC_InflowMonthly1              1155
#define IDC_InflowWeekly1               1156
#define IDC_OutflowMonthly1             1157
#define IDC_OutflowWeekly1              1158
#define IDC_OutflowOK3                  1161
#define IDC_LakeArea2                   1162
#define IDC_CurrentWorkDir              1163
#define IDC_DataDirectory               1164
#define IDC_TimeSeriesFilename          1165
#define IDC_FileStatus                  1166
#define IDC_FileStatus2                 1167
#define IDC_AirMTBEOK3                  1168
#define IDC_UMonthly1                   1169
#define IDC_UWeekly1                    1170
#define IDC_TAMonthly1                  1171
#define IDC_UOK3                        1172
#define IDC_TAWeekly1                   1173
#define IDC_TAOK3                       1174
#define IDC_PAWeekly1                   1175
#define IDC_PAMonthly1                  1176
#define IDC_MLDOK3                      1177
#define IDC_LDOK3                       1178
#define IDC_AtmMTBEMonthly1             1179
#define IDC_MLDWeekly1                  1180
#define IDC_MTBEInputWeekly1            1181
#define IDC_InflowOK3                   1182
#define IDC_MTBEOK3                     1183
#define IDC_LocalTitle                  1185
#define IDC_OutflowHeight               1186
#define IDC_InflowHeight                1187
#define IDC_NumPointsChanged            1188
#define IDC_EnterPoint                  1191
#define IDC_LakeDepthProfile            1192
#define IDC_ProfilePoints               1193
#define IDC_InflowHeightMonthly         1194
#define IDC_InflowHeightWeekly          1195
#define IDC_InflowHeightDaily           1196
#define IDC_InflowHeightOK              1197
```

```
#define IDC_InflowHeightOK2             1198
#define IDC_InflowHeightWeekly1         1199
#define IDC_InflowHeightMonthly1        1200
#define IDC_InflowHeightOK3             1201
#define IDC_OutflowHeightWeekly1        1202
#define IDC_OutflowHeightMonthly1       1203
#define IDC_OutflowHeightOK3            1204
#define IDC_OutflowHeightOK2            1206
#define IDC_EpiLossRate                 1207
#define IDC_EpiLossWeekly1              1208
#define IDC_EpiLossMonthly1             1209
#define IDC_EpiLossOK3                  1210
#define IDC_HypLossRate                 1211
#define IDC_HypLossOK3                  1212
#define IDC_HypLossMonthly1             1213
#define IDC_HypLossWeekly1              1214
#define IDC_EpiLossMonthly              1215
#define IDC_EpiLossWeekly               1216
#define IDC_EpiLossDaily                1217
#define IDC_EpiLossOK2                  1218
#define IDC_EpiLossOK                   1219
#define IDC_HypLossMonthly              1220
#define IDC_HypLossWeekly               1221
#define IDC_HypLossDaily                1222
#define IDC_HypLossOK2                  1223
#define IDC_HypLossOK                   1224
#define IDC_Wank_b0_sol                 1225
#define IDC_Wank_b1_sol                 1226
#define IDC_Wank_b2_sol                 1227
#define IDC_DataUnits                   1228
#define IDC_MonthlyTitle                1229
#define IDC_InflowChoice1               1230
#define IDC_InflowChoice2               1231
#define IDC_GraphData                   1232
#define IDC_LoadData                    1233
#define IDC_LAErrorMessage              1235
#define IDC_ScDay                       1236
#define IDC_ScVal                       1237
#define IDC_CalcSc                      1238
#define IDC_HDay                        1239
#define IDC_HVal                        1240
#define IDC_CalcH                       1241
#define IDC_RelativeHumidity            1242

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        123
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1191
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```