MAST Programmer's Help Contents

This table of contents lists every form and page that has code attached to it in the MAST application. It also includes the library, MASTLIB.LSL, which is used by the application. Click a topic to display its associated material.

DIVEWRCK

Methods

mouseEnter

mouseExit

mouseRightUp

keyPhysical

menuAction

Procedures

customMenu

Var, Const, and Uses windows

Var window

Const window

Uses window

Methods and procedures from MASTLIB.LSL

clearPageValues

createObjMenu

mastObjMsg

setPageValues

pageNameOf

returnisCalled

setisCalledTrue

setisCalledFalse

mouseEnter

diveWrck.mouseEnter

A form-level messaging routine that displays names of objects (excluding Text and Bitmap classes) in the status bar depending on mouse cursor position. Uses the Mastlib library method, mastLib.mastObjMsg.

```
method mouseEnter(var eventInfo MouseEvent)

; Form level messaging system that evaluates each object as mouse enters
  it,
; and looks up the appropriate message from MsgHelp.db under control
; of mastObjMsg library method.

; prevent repeat method calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(Uio)  ; what is the object?
     uioName = uio.Name
   ; exclude text and bitmap objects, and noise names
  if uio.name <> self.name AND  ; eliminate message if form
  (uio.class = "Field" OR uio.class = "Multirecord" OR uio.class =
  "Button")
     then
     mastLib.mastObjMsg(formName.name, mastlib.pageNameOf(uio), uioName)
  endIf
endIf

endMethod
```

mouseExit

diveWrck.mouseExit

Clears the mouseEnter status bar message when the mouse cursor leaves an object's region.

```
method mouseExit(var eventInfo MouseEvent)

; prevent repeat MouseExits
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio)  ; what is the object?
  if uio.class <> "Text" and uio.class <> "Bitmap" then
     message("")
  endIf
endIf

endMethod
```

mouseRightUp

diveWrck.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu. If you click on a Text or Bitmap object, the Help menu for the containing object appears because the objects have no code attached.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls a mastlib.createObjMenu() method that displays an object
; level help menu.

; prevent repeat method calls
if eventInfo.isPreFilter() then
   eventInfo.getTarget(uio) ; what is the object?
   ;if object not diveFlagBox bitmap,
   ;find ui container if object text or bitmap
   if uio.name <> "diveFlagBox" and uio.name <> "diveFlagBox1" then
           while
           uio.class = "Text" or uio.class = "Bitmap"
           uio.attach(uio.ContainerName)
           endWhile
   endIf
  mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

keyPhysical

diveWrck.keyPhysical

Intercepts specified keystrokes and then performs an appropriate pushButton or moveTo call. For example, *Alt+A* calls acceptBttn.pushButton, and *Alt+G* calls passengerBttn.moveTo. *F1* is also intercepted to call helpShowContext.

```
method keyPhysical(var eventInfo KeyEvent)

var
  theKey String
endvar

theKey = eventInfo.vChar()  ; tell me what key was pressed

if eventInfo.isPreFilter() then

  if eventInfo.isAltKeyDown() then
     switch
           case theKey = "A" or theKey = "a" : disableDefault
                 acceptBttn.pushbutton()
           case theKey = "C" or theKey = "c" : disableDefault
                 cancelBttn.pushbutton()
           case theKey = "D" or theKey = "d" : disableDefault
                 intactBttn.moveto()
           case theKey = "E" or theKey = "e" : disableDefault
                 clearMyQueryBttn.pushbutton()
           case theKey = "G" or theKey = "g" : disableDefault
                 passengerBttn.moveto()
           case theKey = "I" or theKey = "i" : disableDefault
                 treasureBttn.moveto()
           case theKey = "S" or theKey = "s" : disableDefault
                 wreckFld.moveto()
           otherwise: doDefault
     endswitch

   else
    switch
     case theKey = "VK_F1" : disableDefault
                 helpShowContext(helpfile, 20005)
     otherwise : doDefault
   endSwitch
  endIf

endIf

endMethod
```

menuAction

diveWrck.menuAction

Tests the string returned by a diveWrck.customMenu selection and calls the appropriate method. For example, choosing the Cancel item calls cancelBttn.pushButton. Choosing Next calls active.action(DataNextRecord). An ampersand (&) in the menu item makes the following key a hot key. For example, &Cancel calls cancelBttn.pushButton when *Alt+C* is pressed.

```
method menuAction(var eventInfo MenuEvent)

var
  mc String
  aboutFm Form
endvar

if eventInfo.isPrefilter()
  then
  doDefault
else
  ; what menu option has been chosen?
  mc = eventInfo.menuChoice()
  switch
      case mc = "&Accept"          : acceptBttn.pushbutton()
    case mc = "&Cancel"          : cancelBttn.pushbutton()
   case mc = "Cl&ear"           : clearMyQueryBttn.pushButton()
    case mc = "&First\tCtrl+F11" : myBar.firstButton.pushButton()
    case mc = "&Last\tCtrl+F12"  : myBar.lastButton.pushButton()
    case mc = "&Next\tF12"       : myBar.nextButton.pushButton()
    case mc = "&Previous\tF11"   : myBar.priorButton.pushButton()
    case mc = "&Contents"        : helpShowIndex(helpFile)
    case mc = "&Using Help"      : helpOnHelp()
     case mc = "&About MAST"      : mastLib.setIsCalledTrue()

   ; block user from closing form with System menu
   case eventInfo.ID() = MenuControlClose :
     eventInfo.setErrorCode(1)
     myBar.hide()
     formReturn(True)
endSwitch
endIf

endMethod
```

customMenu

diveWrck.customMenu

Custom procedure attached to the form. Called by diveWrck.wrecksPage.open to display the Wrecks selection menu.

```
proc customMenu()

Var
 mainMenu Menu
 dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar

; this custom proc displays a custom menu
dropMenu1.addText("&Accept")
dropMenu1.addText("&Cancel")
dropMenu1.addText("Cl&ear")
mainMenu.addPopUp("Se&lections", dropMenu1)

dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("&Previous\tF11")
mainMenu.addPopUp("Rec&ord", dropMenu2)

dropMenu3.addText("&Contents")
dropMenu3.addText("&Using Help")
mainMenu.addPopUp("&Help", dropMenu3)

mainMenu.show()

endproc
```

diveWrck Var Window

Variables declared in a form's Var window are visible to all objects in the form.

```
Var
  uioName String
  formName, myBar Form
  wasCalled Logical
  uio UIObject
  mastLib Library
endVar
```

diveWrck Const Window

Constants declared in a form's Const window are visible to all objects in the form.

```
Const
  helpFile = ":WORK:Mastuser.hlp"
endConst
```

diveWrck Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  createObjMenu(var formName Form,
                pageName String, uio UIObject)
  mastObjMsg(formText String, pageName String, objID String)
  setPageValues(formName String, pageName String) Logical
  pageNameOf(ui UIObject) String
  returnIsCalled() Logical
  setIsCalledTrue()
  setIsCalledFalse()
enduses
```

DIVEWRCK.wrecksPage

Methods

open

close

setFocus

clearPageObjects

Uses window

Uses

open

diveWrck.wrecksPage.open

This form must be called from DIVEPLAN.FSL; it cannot be run directly. Opens the MASTLIB.LSL library if found in the WORK directory. Opens a custom SpeedBar (VIDEOBAR.FSL) and positions it in the upper left corner of screen. Calls the customMenu procedure to display the Wrecks selection menu.

```
method open(var eventInfo Event)

; This open method will not open Divewrck.fsl unless it
; has been called from Diveplan.fsl. The variable isCalled in
; Mastlib.lsl has to be set to True for Divewrck to be opened.
; isCalled is set to True by mastlib.setIsCalledTrue() which is
; invoked by the pushbutton method on Diveplan.

delayScreenUpdates(Yes)

;open library (make sure that it is in the WORK directory)
if not mastLib.Open("MastLib", GlobalToDesktop) then
  msgStop("Failure", "Mastlib could not be opened.")
  formReturn(false)

else
;  wasCalled = mastLib.returnIsCalled()
  if wasCalled = False then
     msgStop("Stop!", "You cannot open this form directly. It must be
  opened from Diveplan.fsl.")
     close()

  else
     ;display custom menu
     customMenu()
     formName.attach()
  endIf

endIf

delayScreenUpdates(No)

endMethod
```

close

diveWrck.wrecksPage.close
Closes the VIDEOBAR.FSL SpeedBar.

```
method close(var eventInfo Event)

   ; Videobar is open
   if mastlib.isVCROpen()
     then
      ; close videobar
      myBar.close()
      mastlib.VCRisClosed()
   endIf

endMethod
```

setFocus

diveWrck.wrecksPage.setFocus

Displays the VIDEOBAR.FSL SpeedBar, opening it first, if necessary.

```
method setFocus(var eventInfo Event)

  var
   x, y, w, h LongInt
  endvar

  ; videoBar is already open
  if MastLib.isVCROpen()
    then
     if myBar.isAssigned()
       then
           ; Videobar open -- attach it to this form
           myBar.bringToTop()
           myBar.setCallFormName(getTitle(), 5500, 400)
     else
           ; attach to Videobar
           myBar.attach(mastLib.vcrName())
           myBar.bringToTop()
           myBar.setCallFormName(getTitle(), 5500, 400)
     endIf
  else
     ; Videobar not open -- open it
     myBar.Open("Videobar.fsl")
     myBar.setCallFormName(getTitle(), 5500, 400)
     mastLib.VCRisOpen()
  endIf

endMethod
```

clearPageObjects

DIVEWRCK.wrecksPage.clearPageObjects

Clears all field objects on the page by setting their values to the empty string ("").

```
method clearPageObjects(pageName String) Logical

; This custom methods clears all objects on that page that have been
; selected.

var
  pageObjTc TCursor
   fieldObj UIObject
endVar

; list all the selectable fields on the page
mastLib.listPageObjects(formName.name, pageName)

; open tcursor on list of selectable fields
pageObjTc.open("Objlist.db")

; clear all selected fields
scan pageObjTc :
   fieldObj.attach(pageObjTc."Object Path")
  if fieldObj  <> ""
    then
     fieldObj = ""
  endIf
endScan

; clear form.page values in Qbeval.db
mastLib.clearPageValues(formName.Name, pageName)

RETURN TRUE

endMethod
```

DIVEWRCK.wrecksPage Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal

  clearPageValues(formText String, pageName String)
  setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
  isVCROpen() Logical
  VCRisOpen()
  VCRisClosed()
  vcrName() String

endUses
```

newValue

DIVEWRCK.wrecksPage.Ship_Name.newValue

Resets the value of the *Ship_Name* field object when a new ship has been selected in the Shipwreck drop-down list.

```
method newValue(var eventInfo Event)

; the wreckFld drop-down needs to updated to reflect
; new Ship Name field value
if self <> wreckFld then
   wreckFld = self
endIf

endMethod
```

DIVEWRCK.wrecksPage.wreckFld

Methods

changeValue

newValue

changeValue

DIVEWRCK.wrecksPage.wreckFld.changeValue

Checks to see if the value of this field object matches the value of *Ship_Name* . If not, the two fields are resynchronized.

```
method changeValue(var eventInfo ValueEvent)

var
  newValStr String
  shipTc TCursor
endVar

;what is the new value?
newValStr = eventInfo.newValue()

;if the new value is not the same as the Ship Name field
;and new value is not blank, resync fields
if newValStr <> Ship_Name and newValStr <> ""
  then
  ;resync fields
   shipTc.attach(Ship_Name)
   shipTc.Locate("Ship Name", newValStr)
   Ship_Name.moveToRecord(shipTc)
endIf

endMethod
```

newValue

DIVEWRCK.wrecksPage.wreckFld.newValue

Posts the new value of this field object to trigger its built-in changeValue method.

```
method newValue(var eventInfo Event)

; commit the field if the user has changed it
; in order to cause a changeValue
if eventInfo.reason() = EditValue
  then
   self.action(EditCommitField)
endIf

endMethod
```

DIVEWRCK.wrecksPage.WreckList

METHODS

open

pushButton

open

DIVEWRCK.wrecksPage.WreckList.open

Fills this list with entries from the Ship_Name fields of SHIPWRCK.DB.

```
method open(var eventInfo Event)
  self.dataSource = "[Shipwrck.Ship Name]"
endMethod
```

pushButton

DIVEWRCK.wrecksPage.WreckList.pushButton

Commits the value selected from *WreckList* to *WreckFld*.

```
method pushButton(var eventInfo Event)
  wreckFld.action(EditCommitField)
endMethod
```

pushButton

DIVEWRCK.wrecksPage.acceptBttn.pushButton

Uses Mastlib.setPageValues to post the selected shipwreck criteria and return to DIVEPLAN.FSL. Also hides the custom SpeedBar.

```
method pushButton(var eventInfo Event)

  eventInfo.getTarget(uio)
  ; assign page values to Qbeval.db
  mastLib.setPageValues(formName.name, mastlib.pageNameof(uio))
  ; hide Videobar
  myBar.hide()
  formReturn(True)

endMethod
```

pushButton

DIVEWRCK.wrecksPage.helpButton.pushButton

Displays Help by calling helpShowContext. The value 20005 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method pushButton(var eventInfo Event)
  helpShowContext(helpfile, 20005)
endMethod
```

pushButton

DIVEWRCK.wrecksPage.cancelBttn.pushButton

Hides the custom SpeedBar and returns to DIVEPLAN.FSL with no further processing.

```
method pushButton(var eventInfo Event)

  ; hide Videobar
  myBar.hide()
  ;return to divePlan without any processing
  formReturn(True)

endMethod
```

pushButton

DIVEWRCK.wrecksPage.clearMyQueryBttn.pushButton

Clears current selected shipwreck values using MastLib.clearPageValues. Called by choosing the Clear Query button.

```
method pushButton(var eventInfo Event)

  ; what is the object?
  eventInfo.getTarget(uio) ; what is the object?

  ; clear field objects and also Qbeval.db values
  clearPageObjects(Mastlib.pageNameOf(Uio))

endMethod
```

DIVEBIO

Methods

mouseEnter

mouseExit

mouseRightUp

keyPhysical

menuAction

Procedures

customMenu

Var, Const, and Uses windows

Var window

Const window

Uses window

Methods and procedures from MASTLIB.LSL

clearPageValues

createObjMenu

mastObjMsg

setPageValues

pageNameOf

returnisCalled

setisCalledTrue

setisCalledFalse

mouseEnter

DIVEBIO.mouseEnter

A form-level messaging routine that displays the names of objects (excluding Text and Bitmap classes) in the status bar depending on mouse cursor position. Uses the Mastlib library method masLib.mastObjMsg.

```
method mouseEnter(var eventInfo MouseEvent)

; prevent repeat mouseEnter calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio)  ; what is the object?
     uioName = uio.Name
   ; exclude text and bitmap objects, and noise names
  if uio.name <> self.name AND  ; eliminate message if form
  (uio.class = "Field" OR uio.class = "Multirecord" OR uio.class =
  "Button")
     then
     mastLib.mastObjMsg(formName.name, mastlib.pageNameOf(uio), uioName)
  endIf
endIf

endMethod
```

mouseExit

DIVEBIO.mouseExit

Clears the mouseEnter status bar message when the mouse cursor leaves an object's region.

```
method mouseExit(var eventInfo MouseEvent)

; prevent repeat mouseExit calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio)  ; what is the object?
  if uio.class <> "Text" and uio.class <> "Bitmap" then
     message("")
  endIf
endIf

endMethod
```

mouseRightUp

DIVEBIO.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu. If you click on a Text or Bitmap object, the Help menu for the containing object appears because the objects have no code attached.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls a mastlib.createObjMenu() method that displays an object
; level help menu.

; prevent repeat method calls
if eventInfo.isPreFilter() then
   eventInfo.getTarget(uio) ; what is the object?
   ;if object not diveFlagBox bitmap,
   ;find ui container if object text or bitmap
   if uio.name <> "diveFlagBox" and uio.name <> "diveFlagBox1" then
     while
           uio.class = "Text" or uio.class = "Bitmap"
           uio.attach(uio.ContainerName)
           endWhile
   endIf
     mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

keyPhysical

DIVEBIO.keyPhysical

Intercepts specified keystrokes and then performs an appropriate pushButton or moveTo call (or invokes the help system, if you press *F1*). For example, *Alt+A* calls acceptBttn.pushButton.

```
method keyPhysical(var eventInfo KeyEvent)

var
  theKey String
endvar

theKey = eventInfo.vChar()  ; tell me what key was pressed

if eventInfo.isPreFilter() then

  if eventInfo.isAltKeyDown() then
     switch
           case theKey = "A" or theKey = "a" : disableDefault
           acceptBttn.pushbutton()
           case theKey = "C" or theKey = "c" : disableDefault
           cancelBttn.pushbutton()
           case theKey = "E" or theKey = "e" : disableDefault
           clearMyQueryBttn.pushbutton()
           case theKey = "M" or theKey = "m" : disableDefault
           commonFld.moveto()
           case theKey = "S" or theKey = "s" : disableDefault
           latinFld.moveto()
                 otherwise: doDefault
     endswitch

  else
     switch
           case theKey = "VK_F1" : disableDefault
                 helpShowContext(helpfile, 20002)
           otherwise : doDefault
     endSwitch
  endIf

endIf

endMethod
```

menuAction

DIVEBIO.menuAction

Tests the string returned by a menu selection and calls the appropriate method. For example, choosing the Cancel item calls cancelBttn.pushButton. Choosing Next calls myBar.nextButton.pushButton(). An ampersand (&) in the menu item makes the following key a hot key. For example, &Cancel calls cancelBttn.pushButton when *Alt+C* is pressed.

```
method menuAction(var eventInfo MenuEvent)
var
 mc String
 aboutFM Form
endvar

if eventInfo.isPrefilter() then
  doDefault
else
  ; what menu option has been chosen?
  mc = eventInfo.menuChoice()
  switch
     case mc = "&Accept"          : acceptBttn.pushbutton()
     case mc = "&Cancel"          : cancelBttn.pushbutton()
     case mc = "Cl&ear"           : clearMyQueryBttn.moveto()
     case mc = "&First\tCtrl+F11" : myBar.firstButton.pushButton()
     case mc = "&Last\tCtrl+F12"  : myBar.lastButton.pushButton()
     case mc = "&Next\tF12"       : myBar.nextButton.pushButton()
     case mc = "&Previous\tF11"   : myBar.priorButton.pushButton()
     case mc = "&Contents"        : helpShowContext(helpFile, 20002)
     case mc = "&Using Help"      : helpOnHelp()
     case mc = "&About MAST"      : mastLib.setIsCalledTrue()
           ; if aboutFm.open("Abtmast.fsl") then
                 ; aboutFm.Wait()  ;modal dialog opened
                 ; aboutFm.Close() ;now close it
                 ; mastLib.setIscalledFalse()
           ; endIf
    ; block user from closing form with System menu
     case eventInfo.ID() = MenuControlClose :
           eventInfo.setErrorCode(1)
           myBar.hide()
           formReturn(True)
  endswitch
endIf

endMethod
```

customMenu

DIVEBIO.customMenu

Called by diveBio.marineLifePage.open to build and display the *diveBio* selection menu.

```
proc customMenu()

Var
 MainMenu Menu
 dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar

; this custom proc displays a custom menu
dropMenu1.addText("&Accept")
dropMenu1.addText("&Cancel")
dropMenu1.addText("Cl&ear")
mainMenu.addPopUp("Se&lections", dropMenu1)

dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("&Previous\tF11")
mainMenu.addPopUp("Rec&ord", dropMenu2)

dropMenu3.addText("&Contents")
dropMenu3.addText("&Using Help")
dropMenu3.addText("&About MAST")
mainMenu.addPopUp("&Help", dropMenu3)

mainMenu.show()

endproc
```

DIVEBIO Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
  uioName String
  formName, myBar Form
  wasCalled Logical
  uio UIObject
  choice, newVal String
  mastLib Library
  fishTc TCursor
endVar
```

DIVEBIO Const Window

Constants declared in the form's Const window are visible to all objects in the form.

```
Const
  helpFile = ":work:mastuser.hlp"
  ; path to Help file must be hardcoded (or file must be on path)
endConst
```

DIVEBIO Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  clearPageValues(formName String, pageName String)
  createObjMenu(var formName Form,
                 pageName String, uio UIObject)
  mastObjMsg(formName String, pageName String, objID String)
  setPageValues(formName String, pageName String) Logical
  pageNameOf(ui UIObject) String
  returnIsCalled() Logical
  setIsCalledTrue()
  setIsCalledFalse()
enduses
```

DIVEBIO.marineLifePage

METHODS

open

close

setFocus

Uses window

Uses window

open

DIVEBIO.marineLifePage.open

```
; Paradox for Windows Sample Application 12/19/92

method open(var eventInfo Event)

; This open method will not open Divebio.fsl unless it
; has been called from Diveplan.fsl. The variable isCalled in
; Mastlib.lsl has to be set to True for Divebio to be opened.
; isCalled is set to True by mastlib.setIsCalledTrue() which is
; invoked by the pushbutton method on Diveplan.

delayScreenUpdates(Yes)

;open library (make sure that it is in the WORK directory)
if not mastLib.Open("MastLib", GlobalToDesktop) then
  disableDefault
  msgStop("Failure", "Mastlib could not be opened.")
  formReturn(false)
  wasCalled = False

else
  wasCalled = mastLib.returnIsCalled()
  if wasCalled = False then
    msgStop("Stop!", "You cannot open this form directly. It must be opened
  from Diveplan.fsl.")
    formReturn(False)

  else
   ;display custom menu
   customMenu()
   formName.attach()
  endif

endif

delayScreenUpdates(No)

endmethod
```

close

DIVEBIO.marineLifePage.close

Closes the attached VIDEOBAR.FSL SpeedBar.

```
method close(var eventInfo Event)

  ; Videobar is open
  if mastlib.isVCROpen()
   then
     ; close videobar
    myBar.close()
    mastlib.VCRisClosed()
  endIf

endMethod
```

setFocus

DIVEBIO.marineLifePage.setFocus

```
method setFocus(var eventInfo Event)

  var
   x, y, w, h LongInt
  endvar

  If wasCalled then  ; Don't load VIDEOBAR if not opened by DIVEPLAN

     ; videoBar is already open
     if MastLib.isVCROpen() then
           if myBar.isAssigned() then
                 ; Videobar open -- attach it to this form
                 myBar.bringToTop()
                 myBar.setCallFormName(getTitle(), 5500, 1050)
           else
                 ; attach to Videobar
                 myBar.attach(mastlib.vcrName())
                 myBar.bringToTop()
                 myBar.setCallFormName(getTitle(), 5500, 1050)
           endif
     else
           ; Videobar not open -- open it
           myBar.Open("Videobar.fsl")
           myBar.setCallFormName(getTitle(), 5500, 1050)
           mastLib.VCRisOpen()
     endif
  endIf

endmethod
```

DIVEBIO.marineLifePage Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal

  VCRisOpen()
  VCRisClosed()
  isVCROpen() Logical
  vcrName() String
  listPageObjects(formName String, pageName String) Logical

endUses
```

DIVEBIO.marineLifePage.commonFld

METHODS

newValue

changeValue

newValue

DIVEBIO.marineLifePage.commonFld.newValue

Commits the Common Name value of the selected species to the *commonFld* field object. This value is used later by DIVEPLAN.destqbe to find matching sites.

```
method newValue(var eventInfo Event)
;Ok, let's commit the field if the user has changed it
if eventInfo.reason() = EditValue then
   self.action(EditCommitField)
endIf
endMethod
```

changeValue

DIVEBIO.marineLifePage.commonFld.changeValue

Checks to see if the selected Species Common Name field has changed. If so, all field values are reset.

```
method changeValue(var eventInfo ValueEvent)

; what is the new value?
newVal = eventInfo.newValue()
; if the new value is not the same as the Common name field
if newVal <> biolife.fishMRO.Common_Name then
  ;resync fields
   fishTc.attach(biolife.fishMRO)
   fishTc.Locate("Common Name", newVal)
   biolife.fishMRO.moveToRecord(fishTc)
   latinFld = Species_Name
endIf

endMethod
```

open

DIVEBIO.marineLifePage.commonList.open

Fills the list with entries from the Common_Name field of BIOLIFE.DB.

```
method open(var eventInfo Event)
  ;fill list with Common Name field of Biolife.db
  self.dataSource = "[Biolife.Common Name]"
endMethod
```

DIVEBIO.marineLifePage.latinFld

Methods

newValue

changeValue

newValue

DIVEBIO.marineLifePage.latinFld.newValue

Commits the Latin Name value of the selected Species to the *latinFld* field object. This value is used later by DIVEPLAN.destqbe to find matching sites.

```
method newValue(var eventInfo Event)
;Ok, let's commit the field if the user has changed it
if eventInfo.reason() = EditValue then
   self.action(EditCommitField)
endIf
endMethod
```

changeValue

DIVEBIO.marineLifePage.latinFld.changeValue

Checks to see if the selected Species Latin Name field has changed. If so, the selected Common Name field is set to match and displayed.

```
method changeValue(var eventInfo ValueEvent)

;what is the new value?
newVal = eventInfo.newValue()
;if the new value is not the same as the Species Name field
if newVal <> Species_Name then
  ;resync fields
   fishTc.attach(biolife.fishMRO)
   fishTc.Locate("Species Name", newVal)
   biolife.fishMRO.moveToRecord(fishTc)
   commonFld = fishTC."Common Name"
endIf

endMethod
```

open

DIVEBIO.marineLifePage.latinList.open

Fills the list with entries from the Latin_Name ("Species Name") field of BIOLIFE.DB.

```
method open(var eventInfo Event)
  ;fill list with Species Name field of Biolife.db
  self.dataSource = "[Biolife.Species Name]"
endMethod
```

newValue

DIVEBIO.marineLifePage.Graphic.newValue

If the selected Common_Name field changes, resets *CommonFld* and *LatinFld* field objects and displays the corresponding graphic from BIOLIFE.DB.

```
method newValue(var eventInfo Event)

if Common_Name <> CommonFld then
   CommonFld = Common_Name
  LatinFld = Species_Name
endIf

endMethod
```

pushButton

DIVEBIO.marineLifePage.acceptBttn.pushButton

Uses **Mastlib.setPageValues** to post the selected marine life species and returns to DIVEPLAN.FSL. Also hides the custom SpeedBar.

```
method pushButton(var eventInfo Event)

  eventInfo.getTarget(uio)  ; what object is this?
  ; assign field values to Qbeval.db
  MastLib.setPageValues(formName.name, MastLib.pageNameOf(uio))
  ; hide Videbar
  myBar.hide()
  formReturn(True)

endMethod
```

pushButton

DIVEBIO.marineLifePage.helpButton.pushButton

Displays Help by calling **helpShowContext**. The value 20002 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method pushButton(var eventInfo Event)
  helpShowContext(helpfile, 20002)
endMethod
```

pushButton

DIVEBIO.marineLifePage.cancelBttn.pushButton

Hides the custom SpeedBar and returns to DIVEPLAN.FSL with no further processing.

```
method pushButton(var eventInfo Event)

  ; hide Videbar
  myBar.hide()
  ; return to Diveplan without any processing
  formReturn(True)

endMethod
```

pushButton

DIVEBIO.marineLifePage.clearMyQueryBttn.pushButton
Called by choosing the Clear Query button. Clears current selected marine life values using **MastLib.clearPageValues.**

```
method pushButton(var eventInfo Event)

; what is the object?
eventInfo.getTarget(uio) ; what is the object?

; clear form.page values in Qbeval.db
mastLib.clearPageValues(formName.Name, mastlib.pageNameOf(uio))

endMethod
```

ABTMAST

Methods

open

mouseRightUp

keyPhysical

Var, Const, and Uses Windows

Var window

Const window

Uses window

Methods and procedures from library MASTLIB.LSL

createObjMenu

pageNameOf

returnIsCalled

open

ABTMAST.open

The ABTMAST form must be called from DIVEPLAN.FSL; it cannot be called directly. Opens ABTMAST.FSL. Opens the MASTLIB.LSL library if found in the WORK directory.

```
; This open method will not open Abtmast.fsl unless it
; has been called from Diveplan.fsl. The variable isCalled in
; Mastlib.lsl has to be set to True for Abtmast to be opened.
; isCalled is set to True by mastlib.setIsCalledTrue() which is
; invoked by the pushbutton method on Diveplan.

method open(var eventInfo Event)
if not eventInfo.isPreFilter()
  then
  ;open library (make sure that it is in the WORK directory)
  if not mastLib.open("MastLib", globalToDesktop) then
     msgStop("Failure", "Mastlib could not be opened.")
     close()
  else
     wasCalled = mastLib.returnIsCalled()
     if wasCalled = False then
           msgStop("Stop!", "You cannot open this form directly. It must be
  opened from Diveplan.fsl.")
           close()
     else
           formName.attach()
     endIf
  endIf
endIf
endMethod
```

mouseRightUp

ABTMAST.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu. If you click on a Text or Bitmap object, the Help menu for the containing object is displayed because the objects have no code attached.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls a mastlib.createObjMenu() method that displays an object
; level help menu.

; prevent repeat method calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio) ; what is the object?
   ;if object not diveFlagBox bitmap,
   ;find ui container if object text or bitmap
   if uio.name <> "diveFlagBox" then
     while uio.class = "Text" or uio.class = "Bitmap"
           uio.attach(uio.ContainerName)
           endWhile
   endIf
  mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

keyPhysical

ABTMAST.keyPhysical

Intercepts *F1* to call helpShowContext. The value 20000 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method keyPhysical(var eventInfo KeyEvent)

; if F1 pressed, display help for divePlan

if eventInfo.isFirstTime()
  then
     if eventInfo.vChar() = "VK_F1" then
           disableDefault
           helpShowContext(helpfile, 20000)
  endIf
endIf

endMethod
```

ABTMAST Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
   callerFm Form
  uio UIObject
  formName Form
  mastLib Library
  wasCalled Logical
endVar
```

ABTMAST Const Window

Constants declared in the form's Const window are visible to all objects in the form.

```
Const
   helpfile = ":WORK:mastuser.hlp"
endConst
```

ABTMAST Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  createObjMenu(var formName Form,
                pageName String, uio UIObject)
  pageNameOf(ui UIObject) String
  returnIsCalled() Logical
endUses
```

pushButton

ABTMAST.abtPg.Button16.pushButton

Returns control to DIVEPLAN.

```
method pushButton(var eventInfo Event)
  formReturn()
endMethod
```

DIVEPLAN

Methods

close

keyPhysical

mouseEnter

mouseExit

mouseRightUp

menuAction

destqbe

Var, Const, and Uses Windows

Var window

Const window

Uses window

Methods and procedures from library MASTLIB.LSL

clearPageValues

createObjMenu

mastObjMsg

pageNameOf

setIsCalledTrue

setIsCalledFalse

close

DIVEPLAN.close

Restores the Paradox SpeedBar and removes the custom menu.

```
method close(var eventInfo Event)

if eventInfo.isPrefilter()
  then
   doDefault

else
   ; close Divebio
  if bioFm.isAssigned() then
      bioFm.close()
   endIf

   ; close Divesite
   if siteFm.isAssigned() then
     siteFm.close()
   endIf

   ; close Divewrck
   if wreckFm.isAssigned() then
     wreckFm.close()
   endIf

   ; close Mastord
   if bookFm.isAssigned() then
     mastlib.formCanClose()
     bookFm.close()
   endIf

  ;restore SpeedBar
     showSpeedBar()
  ;remove custom menu and restore default menu
  removeMenu()
  ;close open help
     helpQuit(helpfile)
  ; now close the form

endIf

endMethod
```

keyPhysical

DIVEPLAN.keyPhysical

Intercepts keystrokes and translates them to method calls, if appropriate.

```
method keyPhysical(var eventInfo KeyEvent)

; Form level keyphysical to trap hot keys.
; Will trap F1 (Help) calls.

var
  theKey String
  thePage String
endvar

if eventInfo.isPreFilter() then
   thePage = mastlib.pagenameOf(active)  ; tell me the page
   theKey = eventInfo.vChar()  ; tell me what key was pressed

  switch

     case thePage = "mainPage" :
     if eventInfo.isAltKeyDown() then
       switch
                 case theKey = "E" or theKey = "e" : disableDefault
                 mainPage.clearQueryBttn.pushButton()
                 case theKey = "M" or theKey = "m" : disableDefault
                 mainPage.bioButton.pushButton()
                 case theKey = "P" or theKey = "p" : disableDefault
                 mainPage.processButton.pushButton()
                 case theKey = "S" or theKey = "s" : disableDefault
                 mainPage.siteButton.pushButton()
                 case theKey = "W" or theKey = "w" : disableDefault
                 mainPage.wreckButton.pushButton()
                 case theKey = "X" or theKey = "x" : disableDefault
                 mainPage.exitButton.pushButton()
                 otherwise: doDefault
                 endswitch

     else
     switch
     case theKey = "VK_F1" : disableDefault
  helpShowContext(helpFile, 20000)
     otherwise : doDefault
     endSwitch
  endIf

     case thePage = "resultsPage" :
     if eventInfo.isAltKeyDown() then
      switch
                 case theKey = "B" or theKey = "b" : disableDefault
                 resultsPage.bookItBttn.pushButton()
                 case theKey = "M" or theKey = "m" : disableDefault
                 resultsPage.mainPageBttn.pushButton()
                 case theKey = "X" or theKey = "x" : disableDefault
```

```
                resultsPage.exitButton.pushButton()
                otherwise: doDefault
            endswitch

     else
      switch
                case theKey = "VK_F1" : disableDefault
     helpShowContext(helpFile, 20001)
     otherwise : doDefault
      endSwitch
       endIf

   endswitch

else
  doDefault
endIf

endMethod
```

mouseEnter

DIVEPLAN.mouseEnter

A form-level messaging routine that displays the names of objects in the status bar depending on the mouse cursor position. Uses the Mastlib library method, masLib.mastObjMsg.

```
method mouseEnter(var eventInfo MouseEvent)

; prevent repeat MouseEnter calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio)  ; what is the object?
   uioName = uio.Name
   ; exclude text and bitmap objects, and noise names
  if uio.name <> self.name AND  ; eliminate message if form
    (uio.class = "Field" OR uio.class = "Multirecord" OR uio.class =
  "Button")
     then
    mastLib.mastObjMsg(formName.name, mastlib.pageNameOf(uio), uioName)
  endIf
endIf

endMethod
```

mouseExit

DIVEPLAN.mouseExit

Clears the mouseEnter status bar message when the mouse cursor leaves an object's region.

```
method mouseExit(var eventInfo MouseEvent)

; prevent repeat mouseExit calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio) ; what is the object?
  ;exiting from an object that has displayed a mouseEnter message
  if uio.class <> "Text" and uio.class <> "Bitmap" then
     message("")
  endIf
endIf

endMethod
```

mouseRightUp

DIVEPLAN.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu.

```
method mouseRightUp(var eventInfo MouseEvent)

; prevent repeat method calls
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio) ; what is the object?
;if object not diveFlagBox bitmap,
;find ui container if object text or bitmap
if uio.name <> "diveFlagBox" and uio.name <> "diveFlagBox1" then
     while uio.class = "Text" or uio.class = "Bitmap"
           uio.attach(uio.ContainerName)
           endWhile
   endIf
  mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

destqbe

DIVEPLAN.destqbe

Searches QBEVAL.DB and assigns values from it to a query (destQBE) made on DEST.DB. The values in QBEVAL.DB are set when the user makes selections in any or all of the following files: DIVESITE.FSL, DIVEWRCK.FSL, and DIVEBIO.FSL. Matching destinations are written to POSSDEST.DB. Returns False if QBEVAL.DB cannot be opened. destQBE is triggered by choosing Process in mainPage.pageMenu or by choosing the Process button.

```
method destqbe() Logical

var
   i SmallInt
  destQBE Query
  qbeValTc TCursor
  destName, avgTemp, accom, nLife, siteName, siteHi, shipCat,
  shipInt, shipCond, skill, shipName, commonName, speciesName String
endvar

if not qbeValTc.Open("QbeVal.db") then
  msgStop("Problem", "Couldn't open Qbeval.db")
  return false
endIf

; Examine Qbeval.db record-by-record and assign the values in the
; Query Value field to variables that are included in destQBE.
Scan qbeValTc :
  switch
     case qbeValTc."Field Name" = "Accomodations"    :
           accom = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Site Highlight"   :
           siteHi = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Night Life"       :
           nLife = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Site Name"        :
           siteName = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Destination Name" :
           destName = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Category"         :
           shipCat = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Interest"         :
           shipInt = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Condition"        :
           shipcond = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Avg Temperature"  :
           avgTemp = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Ship Name"        :
           shipName = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Common Name"      :
           commonName = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Species Name"     :
           speciesName = qbeValTc."Query Value"
     case qbeValTc."Field Name" = "Skill Level"      :
           skill = qbeValTc."Query Value"
  endswitch
```

```
endScan

; Define query. Include in the query the variables that have been
; assigned from Qbeval.db.

Switch
  ;Shipwrck and biolife criteria chosen
  Case (not shipname.isBlank() or not shipCat.isBlank() or
           not shipInt.isBlank() or not shipCond.isBlank()) and
        (not commonName.isBlank() or not speciesName.isBlank()) :

     destQBE = Query

     answer: :work:possdest.db
 Dest | Destination No | Destination Name |  Avg Temp (F)  | Avg Temp (C) |
       | Check _destno  | Check ~destName  | Check ~avgTemp | Check
  |

 Dest | Spring Temp (F) | Spring Temp (C) | Summer Temp (F) | Summer Temp
  (C) |
      | Check           | Check           | Check           | Check
  |

 Dest | Fall Temp (F) | Fall Temp (C) | Winter Temp (F) | Winter Temp (C) |
      | Check         | Check         | Check           | Check           |

 Dest | Accomodations |  Night Life  | Body of Water | Travel Cost |
      | Check ~accom  | Check ~nLife | Check         | Check       |

 Sites |       Site No       | Destination No | Site Name | Site Highlight
  |
       | _shipsite, _biosite | _destno        | ~siteName | ~siteHi
  |

 Sites | Skill Level |
       | ~skill      |

 Shipwrck | Ship Name |  Site No  | Category | Interest |
          | ~shipName | _shipsite | ~shipCat | ~shipInt |

 Shipwrck | Condition |
          | ~shipCond |

 Biosite | Species No | Site No  |
         | _specno    | _biosite |

 Biolife | Species No | Common Name   | Species Name  |
         | _specno    | ~commonName   | ~speciesName  |

Endquery

  ;Shipwrck but not Biolife criteria chosen
  Case     (not shipname.isBlank() or not shipCat.isBlank() or not
  shipInt.isBlank() or
           not shipCond.isBlank()) and (commonName.isBlank() and
```

```
  speciesName.isBlank()) :

destQBE = Query

answer: :work:possdest.db

 Dest | Destination No | Destination Name |  Avg Temp (F)  | Avg Temp (C) |
      | Check _destno  | Check ~destName  | Check ~avgTemp | Check        |

 Dest | Spring Temp (F) | Spring Temp (C) | Summer Temp (F) | Summer Temp
  (C) |
      | Check           | Check           | Check           | Check
  |

 Dest | Fall Temp (F) | Fall Temp (C) | Winter Temp (F) | Winter Temp (C) |
      | Check         | Check         | Check           | Check           |

 Dest | Accomodations |  Night Life  | Body of Water | Travel Cost |
      | Check ~accom  | Check ~nLife | Check         | Check       |

 Sites |  Site No  | Destination No | Site Name | Site Highlight |
       | _shipsite | _destno        | ~siteName | ~siteHi        |

 Sites | Skill Level |
       | ~skill      |

 Shipwrck | Ship Name |  Site No  | Category | Interest |
          | ~shipName | _shipsite | ~shipCat | ~shipInt |

 Shipwrck | Condition |
          | ~shipCond |

Endquery

  ;Biolife but not Shipwrck criteria chosen
   Case shipname.isBlank() and shipCat.isBlank() and
        shipInt.isBlank() and shipCond.isBlank()
        and not commonName.isBlank() and not speciesName.isBlank() :


destQBE = Query

answer: :work:possdest.db

 Dest | Destination No | Destination Name |  Avg Temp (F)  | Avg Temp (C) |
      | Check _destno  | Check ~destName  | Check ~avgTemp | Check        |

 Dest | Spring Temp (F) | Spring Temp (C) | Summer Temp (F) | Summer Temp
  (C) |
      | Check           | Check           | Check           | Check
  |

 Dest | Fall Temp (F) | Fall Temp (C) | Winter Temp (F) | Winter Temp (C) |
      | Check         | Check         | Check           | Check           |
```

```
 Dest | Accomodations |  Night Life  | Body of Water | Travel Cost |
      | Check ~accom  | Check ~nLife | Check         | Check       |

 Sites |  Site No | Destination No | Site Name | Site Highlight |
       | _biosite | _destno        | ~siteName | ~siteHi        |

 Sites | Skill Level |
       | ~skill      |

 Biosite | Species No | Site No  |
         | _specno    | _biosite |

 Biolife | Species No | Common Name   | Species Name  |
         | _specno    | ~commonName   | ~speciesName  |

Endquery


   ;no Shipwrck or Biolife criteria
   Case shipname.isBlank() and shipCat.isBlank() and
        shipInt.isBlank() and shipCond.isBlank()
        and commonName.isBlank() and speciesName.isBlank() :


destQBE = Query

answer: :work:possdest.db

 Dest | Destination No | Destination Name |  Avg Temp (F)  | Avg Temp (C) |
      | Check _destno  | Check ~destName  | Check ~avgTemp | Check        |

 Dest | Spring Temp (F) | Spring Temp (C) | Summer Temp (F) | Summer Temp
  (C) |
      | Check           | Check           | Check           | Check
  |

 Dest | Fall Temp (F) | Fall Temp (C) | Winter Temp (F) | Winter Temp (C) |
      | Check         | Check         | Check           | Check           |

 Dest | Accomodations |  Night Life  | Body of Water | Travel Cost |
      | Check ~accom  | Check ~nLife | Check         | Check       |

 Sites | Destination No | Site Name | Site Highlight |
       | _destno        | ~siteName | ~siteHi        |

 Sites | Skill Level |
       | ~skill      |

endQuery

Endswitch

writeQBE(destQBE, "Diveqbe")
; Execute destQuery (defined above) and write the results to Possdest.db.
; Possdest.db is used by Mastord.fsl to book trips.
```

```
if not executeQBE(destQBE, ":work:Possdest.db") then
   msgInfo("Error","Query could not be processed!")
   return FALSE
endIf

return TRUE

endMethod
```

menuAction

```
method menuAction(var eventInfo MenuEvent)

; This prevents the user from exiting Paradox while DIVEPLAN is running.

  If not eventInfo.isPreFilter() then

     If eventInfo.reason() = menuNormal and
        eventInfo.id() = MenuCanClose and not okToExit then

                 eventInfo.setErrorCode(CanNotDepart)
                 disableDefault
                 message("Use the Exit button to exit DIVEPLAN.")

     endIf
  endIf
endmethod
```

DIVEPLAN Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
  uioName String
  uio UIObject
  formName, siteFm, wreckFm, bioFm, bookFm, aboutFm Form
  mastLib Library
  closeVCRLg Logical
endVar
```

DIVEPLAN Const Window

Constants declared in the form's Const window are visible to all objects in the form.

```
Const
  helpfile = ":WORK:mastuser.hlp"
  ; path to Help file must be hardcoded (or file must be on path)
endConst
```

DIVEPLAN Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  clearPageValues(formName String, pageName String)
  createObjMenu(var formName Form,
                pageName String, uio UIObject)
  mastObjMsg(formName String, pageName String, objID String)
  pageNameOf(ui UIObject) String
  setIsCalledTrue()
  setIsCalledFalse()
endUses
```

DIVEPLAN.mainPage

Methods

arrive

menuAction

open

Procedures

pageMenu

arrive

DIVEPLAN.mainPage.arrive

Sets *mainPage* title to "Main" and displays a custom pop-up menu by calling the mainPage.pageMenu procedure.

```
method arrive(var eventInfo MoveEvent)

  ; define page title
  setTitle("Main")
  ;display custom Speedbar for page
  pageMenu()

endMethod
```

menuAction

DIVEPLAN.mainPage.menuAction

Tests the string returned by a *mainPage* menu selection and calls the appropriate method. For example, choosing the Marine Life menu item calls bioButton.pushButton. Choosing Results moves you to the *resultsPage.*

```
method menuAction(var eventInfo MenuEvent)

var
 mc String
endvar

; what menu option has been chosen?
mc = eventInfo.menuChoice()
switch
  case mc = "&Sites"             : siteButton.pushbutton()
  case mc = "&Marine Life"       : bioButton.pushbutton()
  case mc = "&Wrecks"            : wreckButton.pushbutton()
  case mc = "&Process"           : processButton.pushbutton()
  case mc = "Cl&ear"             : clearQueryBttn.pushbutton()
  case mc = "E&xit"              : exitButton.pushbutton()
  case mc = "&Results"           : resultsPage.moveto()
  case mc = "&Contents"          : helpShowContext(helpFile, 10000)
  case mc = "&Using Help"        : helpOnHelp()
  case mc = "&About MAST"        : mastLib.setIsCalledTrue()
     if aboutFm.open("Abtmast.fsl") then
           aboutFm.Wait()  ; modal dialog opened
           aboutFm.Close() ; now close it
           mastLib.setIscalledFalse()
     endIf
endswitch

endMethod
```

open

DIVEPLAN.mainPage.open

```
; Paradox for Windows Sample Application 12/19/92

method open(var eventInfo Event)

  delayScreenUpdates(Yes)
  ; if the file is not in :WORK:, don't allow open
  if not isFile("diveplan.fsl") then
     msgInfo("Startup Error!", "The MAST application files must be " +
             "in the working directory.")
     close()
  endif

  ;open library (make sure that it is in the WORK directory)
  if not mastLib.open("MastLib", globalToDesktop) then
     msgStop("Failure", "Mastlib could not be opened")
  endif

  ; associate form UIObject name with formName var
  formName.attach()
  hideSpeedBar()
  maximize()
  clearQueryBttn.pushbutton()
  okToExit = False
  delayScreenUpdates(No)

endmethod
```

pageMenu

DIVEPLAN.mainPage.pageMenu

Called by mainPage.arrive to display the *mainPage* menu.

```
proc pageMenu()

Var
 mainMenu Menu
 mainPop1, mainPop2 popUpMenu
endVar

; this custom proc displays a menu for Main page
; define menu
mainPop1.addText("&Sites")
mainPop1.addText("&Marine Life")
mainPop1.addText("&Wrecks")
mainPop1.addSeparator()
mainPop1.addText("&Process")
mainPop1.addText("Cl&ear")
mainPop1.addText("E&xit")
mainMenu.addPopUp("Se&lections", mainPop1)

mainMenu.addText("&Results")

mainPop2.addText("&Contents")
mainPop2.addText("&Using Help")
mainPop2.addText("&About MAST")
mainMenu.addPopUp("&Help", mainPop2)

mainMenu.show()

endproc
```

pushButton

DIVEPLAN.mainPage.processButton.pushButton

Called by choosing Process in *pageMenu*. Calls destqbe to process the query. Displays any matches on the *resultsPage.*

```
method pushButton(var eventInfo Event)

; temporarily bind foundTF to Dest.db
resultsPage.foundTF.tableName = "Dest.db"

; remove Possdest.db if it is already part of the data model
; so that you can repeat the query
if DMHasTable("Possdest.db")
  then
  DMRemovetable("Possdest.db")
endIf

; run query here
Try
  If Not destqbe() Then
      Return
   endIf

onFail  ; query failed - quit method
  fail(1, "Couldn't open Qbeval.db")
Endtry

; query succesful - add query answer (possdest.db) to
; data model and bind foundTF to it
resultsPage.foundTF.tablename = "Possdest.db"

; remove dest.db from the data model
DMRemoveTable("Dest.db")

; move to results page to display chosen destinations
resultsPage.moveTo()

endMethod
```

pushButton

DIVEPLAN.mainPage.helpButton.pushButton

Displays Help by calling helpShowContext. The value 20000 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method pushButton(var eventInfo Event)
;provide help on this page
helpShowContext(helpFile, 20000)
endMethod
```

pushButton

DIVEPLAN.mainPage.exitButton.pushButton

Displays a Quit MAST dialog box. Calls close to close the form if Yes is chosen.

```
method pushButton(var eventInfo Event)

; ask user if they really wants to close diveplan
choice = msgQuestion("Quit M*A*S*T", "Do you want to quit the
  application?")

; if user clicks Yes button, close diveplan
if choice = "Yes"
  then
   close()
endIf

endMethod
```

pushButton

DIVEPLAN.mainPage.clearQueryBttn.pushButton

Clears current query in QBEVAL.DB by using a CHANGETO query to change the Query Value field in each record to blank.

```
method pushButton(var eventInfo Event)

var
  clearQuery           Query
endVar

; build the string

clearQuery=Query

qbeval.db | Query Value     |
          | changeto blank  |

EndQuery

executeQBE(clearQuery)

; empty Possdest.db
if isTable("Possdest.db")
  then
  empty("Possdest.db")
endIf

endMethod
```

pushButton

DIVEPLAN.mainPage.bioButton.pushButton

Opens DIVEBIO.FSL, maximizes it, and gives it focus.

```
method pushButton(var eventInfo Event)

; open Divebio.fsl

; Divebio already open -- display it and put it into wait state
if isAssigned(bioFM)
  then
  bioFm.bringtoTop()
  bioFm.wait()  ; put Divebio into wait state
  ; user returned -- hide Divebio
  bioFm.hide()
  maximize()
  mainPage.moveTo()

else
  ; Divebio not open -- open it
  mastLib.setIsCalledTrue()
  if bioFM.Open("Divebio", WinStyleMaximize)
     then
   bioFm.Wait()  ; put Divebio into wait state
   ; user returned -- hide Divebio
   bioFm.hide()
    maximize()
   mainPage.moveTo()
  else
   ; open failed
   msgInfo("Status", "Sorry, the Marine Life form is not available")
   mastlib.setIsCalledFalse()
  endIf

endIf

endMethod
```

pushButton

DIVEPLAN.mainPage.siteButton.pushButton

Opens DIVESITE.FSL, maximizes it, and gives it focus.

```
method pushButton(var eventInfo Event)

; open Divesite.fsl

; Divesite already open -- display it and put into wait state
if isAssigned(siteFM)
  then
  siteFm.bringtoTop()
  siteFm.wait()  ; put Divesite into wait state
  ; user returned -- hide Divesite
  siteFm.hide()
  maximize()
  mainPage.moveTo()
else
  ; Divesite not open -- open it
  mastLib.setIsCalledTrue()
  if siteFM.Open("Divesite", WinStyleMaximize)
    then
     siteFm.Wait()  ; put Divesite into wait state
    ; user returned -- hide Divesite
     siteFm.hide()
    maximize()
    mainPage.moveTo()
  else
    msgInfo("Status", "Sorry, the Sites form is not available")
    mastlib.setIsCalledFalse()
  endIf
endIf

endMethod
```

pushButton

DIVEPLAN.mainPage.wreckButton.pushButton

Opens DIVEWRCK.FSL, maximizes it, and gives it focus.

```
method pushButton(var eventInfo Event)

; open Divewrck.fsl

; Divewrck already open -- display it and put it into wait state
if isAssigned(wreckFM)
  then
  wreckFm.bringtoTop()
  wreckFm.wait()  ; put Divewrck into wait state
  ; user returned -- hide Divewrck
  wreckFm.hide()
  maximize()
  mainPage.moveTo()

else
  ; Divewrck not open -- open it
  mastLib.setIsCalledTrue()
  if wreckFM.Open("Divewrck", WinStyleMaximize)
      then
    wreckFm.Wait() ; put Divewrck into wait state
    ; user returned -- hide Divewrck
     wreckFm.hide()
    maximize()
    mainPage.moveTo()
  else
    ; Open failed
    msgInfo("Status", "Sorry, the Ship Wreck form is not available")
     mastlib.setIsCalledFalse()
  endIf

endIf

endMethod
```

DIVEPLAN.resultsPage

Methods

arrive

menuAction

Procedures

pageMenu

arrive

DIVEPLAN.resultsPage.arrive

Sets *resultsPage* title to "Results" and displays a custom pop-up menu by calling the resultsPage.pageMenu procedure.

```
method arrive(var eventInfo MoveEvent)

  ; display new page title & menu
  setTitle("Vacations")
  pageMenu()

endMethod
```

menuAction

DIVEPLAN.resultsPage.menuAction

Tests the string returned by a *resultsPage* menu selection and calls the appropriate method. For example, choosing the Book It! menu item calls bookItBttn.pushButton. Choosing Main moves you to the *mainPage*.

```
method menuAction(var eventInfo MenuEvent)

var
 mc String
endvar

; what menu option has been chosen?
mc = eventInfo.menuChoice()
switch
  case mc = "&Book It!"            : bookItBttn.pushbutton()
  case mc = "E&xit"                : exitButton.pushbutton()
  case mc = "&Main"                : mainPage.moveto()
  case mc = "&First\tCtrl+F11"     : active.action(DataBegin)
  case mc = "&Last\tCtrl+F12"      : active.action(DataEnd)
  case mc = "&Next\tF12"           : active.action(DataNextRecord)
  case mc = "&Previous\tF11"       : active.action(DataPriorRecord)
  case mc = "&Insert\tIns"         : active.action(DataInsertRecord)
  case mc = "&Delete\tCtrl+Del"    : active.action(DataDeleteRecord)
  case mc = "C&ancel Changes\tAlt+Bksp" : active.action(DataCancelRecord)
  case mc = "&Contents"            : helpShowContext(helpFile, 10000)
  case mc = "&Using Help"          : helpOnHelp()
  case mc = "&About MAST"          : mastLib.setIsCalledTrue()
     if aboutFm.open("Abtmast.fsl") then
           ; modal dialog opened
           aboutFm.Wait()
           ; now close it
           aboutFm.Close()
           mastLib.setIscalledFalse()
     endIf
endswitch

endMethod
```

pageMenu

DIVEPLAN.resultsPage.pageMenu

Called by resultsPage.arrive to display a custom pop-up menu.

```
proc pageMenu()
Var
 resultsMenu Menu
 dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar

; this custom proc displays a menu for Results page
dropMenu1.addText("&Book It!")
dropMenu1.addText("E&xit")
resultsMenu.addPopUp("Des&tination", dropMenu1)

resultsMenu.addText("&Main")

dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("&Previous\tF11")
dropMenu2.addSeparator()
dropMenu2.addtext("&Insert\tIns")
dropMenu2.addtext("&Delete\tCtrl+Del")
dropMenu2.addtext("C&ancel Changes\tAlt+Bksp")
resultsMenu.addPopUp("Rec&ord", dropMenu2)

dropMenu3.addText("&Contents")
dropMenu3.addText("&Using Help")
dropMenu3.addText("&About MAST")
resultsMenu.addPopUp("&Help", dropMenu3)

resultsMenu.show()

endproc
```

pushButton

DIVEPLAN.resultsPage.helpButton.pushButton

Displays Help by calling helpShowContext. The value 20001 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method pushButton(var eventInfo Event)
  helpShowContext(helpFile, 20001)
endMethod
```

pushButton

DIVEPLAN.resultsPage.bookItBttn.pushButton

```
method pushButton(var eventInfo Event)

; open Mastord.fsl

; Mastord already open -- display it and put into wait state
if isAssigned(bookFM) then
  bookFm.bringtoTop()
  bookFm.show()
  bookFm.wait()
  ; user returned -- hide Divesite
  bookFm.hide()
  mainPage.moveTo()
  maximize()
else
  ; Mastord not open -- so open it
  mastLib.setIsCalledTrue()
  if bookFM.Open("Mastord", WinStyleMaximize) then
     bookFm.show()
     bookFm.Wait()  ; put Divesite into wait state
    ; user returned -- hide Divesite
     bookFm.hide()
    maximize()
    mainPage.moveTo()
  else
    msgInfo("Status", "Sorry, the Sites form is not available")
    mastlib.setIsCalledFalse()
  endif

endif

endmethod
```

pushButton

DIVEPLAN.resultsPage.mainPageBttn.pushButton
Moves users to the *mainPage* by calling mainPage.moveTo.

```
method pushButton(var eventInfo Event)
mainPage.moveTo()
endMethod
```

pushButton

DIVEPLAN.resultsPage.exitButton.pushButton

Displays a Quit MAST dialog box. Calls close to close the form if Yes is chosen.

```
method pushButton(var eventInfo Event)

; ask user if they really wants to close diveplan
choice = msgQuestion("Quit M*A*S*T", "Do you want to quit the
  application?")

; if user clicks Yes button, close diveplan
if choice = "Yes"
  then
   close()
endIf

endMethod
```

DIVESITE

Methods

keyPhysical

menuAction

mouseEnter

mouseExit

mouseRightUp

Var, Const, and Uses windows

Var window

Const window

Uses window

Methods and procedures from library MASTLIB.LSL

clearPageValues

createObjMenu

mastObjMsg

setPageValues

pageNameOf

returnIsCalled

setIsCalledTrue

setIsCalledFalse

keyPhysical

DIVESITE.keyPhysical

Intercepts keystrokes and translates them to method calls, if appropriate.

```
method keyPhysical(var eventInfo KeyEvent)

; Form level keyPhysical to trap hot keys.
; Will trap F1 (Help) calls.

var
  theKey String
  thePage String
endvar

if eventInfo.isPreFilter() then
   thePage = mastlib.pagenameOf(active)
   theKey = eventInfo.vChar()  ; tell me what key was pressed

  switch

     case thePage = "sitesPage" :
     if eventInfo.isAltKeyDown() then
       switch
                 case theKey = "A" or theKey = "a" : disableDefault
                       sitesPage.acceptBttn.pushButton()
                 case theKey = "C" or theKey = "c" : disableDefault
                       sitesPage.cancelBttn.pushButton()
                 case theKey = "D" or theKey = "d" : disableDefault
                       sitesPage.destsPageBttn.pushButton()
                 case theKey = "E" or theKey = "e" : disableDefault
                       sitesPage.clearMyQueryBttn.pushButton()
                 case theKey = "F" or theKey = "f" : disableDefault
                       sitesPage.reefs.moveTo()
                 case theKey = "M" or theKey = "m" : disableDefault
                       sitesPage.cheap.moveTo()
                 case theKey = "N" or theKey = "n" : disableDefault
                       sitesPage.sleepy.moveTo()
                 case theKey = "S" or theKey = "s" : disableDefault
                       sitesPage.sitesField.moveTo()
                 case theKey = "V" or theKey = "v" : disableDefault
                       sitesPage.skillFld.moveTo()
                 case theKey = "W" or theKey = "w" : disableDefault
                       sitesPage.subTropical.moveTo()
                 otherwise: doDefault
     endswitch

     else
           switch
                 case theKey = "VK_F1" : disableDefault
                       helpShowContext(helpFile, 20004)
           otherwise : doDefault
           endSwitch
       endIf
```

```
      case thePage = "destsPage" :
      if eventInfo.isAltKeyDown() then
       switch
                  case theKey = "A" or theKey = "a" : disableDefault
                        destsPage.acceptBttn.pushButton()
                  case theKey = "C" or theKey = "c" : disableDefault
                        destsPage.cancelBttn.pushButton()
                  case theKey = "D" or theKey = "d" : disableDefault
                        destsPage.dest.destination_Name.moveTo()
                  case theKey = "E" or theKey = "e" : disableDefault
                        destsPage.clearMyQueryBttn.pushButton()
                  case theKey = "I" or theKey = "i" : disableDefault
                        destsPage.sites.site_Name.moveTo()
                  case theKey = "T" or theKey = "t" : disableDefault
                        destsPage.addToChoicesBttn.pushButton()
                  case theKey = "S" or theKey = "s" : disableDefault
                        destsPage.sitesPageBttn.pushButton()
                  case theKey = "R" or theKey = "r" : disableDefault
                        destsPage.clearChoicesBttn.pushButton()
                  otherwise: doDefault
      endswitch

      else
       switch
      case theKey = "VK_F1" : disableDefault
                  helpShowContext(helpFile, 20004)
      otherwise : doDefault
       endSwitch
        endIf

   endswitch

else
  doDefault
endIf

endMethod
```

menuAction

DIVESITE.menuAction

Prevents you from using the control menu to close the form.

```
method menuAction(var eventInfo MenuEvent)

if eventInfo.isPreFilter()
  then
     doDefault
  else
    ; block user from closing form with System menu
    if eventInfo.ID() = MenuControlClose then
      eventInfo.setErrorCode(1)
       ; if Videobar is displayed, hide it
       if isAssigned(myBar) then
         myBar.hide()
       endIf
      formReturn(True)
     endIf
endIf

endMethod
```

mouseEnter

DIVESITE.mouseEnter

A form-level messaging routine that displays names of objects in the status bar depending on mouse cursor position. Uses the Mastlib library method masLib.mastObjMsg.

```
method mouseEnter(var eventInfo MouseEvent)

; prevent repeat methods calls
if eventInfo.isPreFilter()
  then
  eventInfo.getTarget(uio) ; what is the object?
   uioName = uio.Name
   ; exclude text and bitmap objects, and noise names
  if uio.name <> self.name AND  ; eliminate message if form
  (uio.class = "Field" OR uio.class = "Multirecord" OR uio.class =
  "Button")
     then
     mastLib.mastObjMsg(formName.name, mastlib.pageNameOf(uio), uioName)
  endIf
endIf

endMethod
```

mouseExit

DIVESITE.mouseExit

Clears the mouseEnter status bar message when the mouse cursor leaves an object's region.

```
method mouseExit(var eventInfo MouseEvent)

; prevent repeat mouseExit
if eventInfo.isPreFilter() then
  eventInfo.getTarget(uio)  ; what is the object?
  if uio.class <> "Text" and uio.class <> "Bitmap" then
   message("")
  endIf
endIf
endMethod
```

mouseRightUp

DIVESITE.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls a mastlib.createObjMenu() method that displays an object
; level help menu.

; prevent repeat method calls
if eventInfo.isPreFilter()
  then
   eventInfo.getTarget(uio) ; what is the object?
   ; if object not diveFlagBox bitmap,
   ; find ui container if object text or bitmap
   if uio.name <> "diveFlagBox1"  and uio.name <> "smlDiveFlagBox" then
     while
      uio.class = "Text" or uio.class = "Bitmap"
      uio.attach(uio.ContainerName)
     endWhile
   endIf
  mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

DIVESITE Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
  formName, aboutFm Form
  wasCalled Logical
  uio UIObject
  mastLib Library
endVar
```

DIVESITE Const Window

Constants declared in the form's Const window are visible to all objects in the form.

```
Const
  helpFile = ":WORK:mastuser.hlp"
  ; path to Help file must be hardcoded (or file must be on path)
endConst
```

DIVESITE Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  clearPageValues(formName String, pageName String)
  createObjMenu(var formName Form,
                 pageName String, uio UIObject)
  mastObjMsg(formName String, pageName String, objID String)
  setPageValues(formName String, pageName String) Logical
  pageNameOf(ui UIObject) String
  returnIsCalled() Logical
  setIsCalledTrue()
  setIsCalledFalse()
endUses
```

DIVESITE.destsPage

Methods

close

arrive

depart

menuAction

setFocus

Procedures

pageMenu

Var, Const, and Uses windows

Var window

Uses window

close

DIVESITE.destsPage.close

Closes the form and the custom SpeedBar.

```
method close(var eventInfo Event)

  ; Videobar is open
  if mastlib.isVCROpen()
    then
      ; close videobar
     myBar.close()
     mastlib.VCRisClosed()
   endIf

endMethod
```

arrive

DIVESITE.destsPage.arrive

Sets *destsPage* title to "Destinations" and sets up a custom menu by calling the destsPage.pageMenu procedure.

```
method arrive(var eventInfo MoveEvent)

  ; change page title
  setTitle("Destinations")

  ; display custom menu
  pageMenu()

endMethod
```

depart

DIVESITE.destsPage.depart
Hides the custom SpeedBar.

```
method depart(var eventInfo MoveEvent)

  ; Hide videobar
  myBar.Hide()

endMethod
```

menuAction

DIVESITE.destsPage.menuAction

Tests the string returned by a *destsPage* menu selection and calls the appropriate method. For example, choosing the Sites item calls sitesPageBttn.pushButton.

```
method menuAction(var eventInfo MenuEvent)

var
 mc String
endvar

; what menu option has been chosen?
mc = eventInfo.menuChoice()
switch
  case mc = "&Accept"          : acceptBttn.pushbutton()
  case mc = "C&ancel"          : cancelBttn.pushbutton()
  case mc = "Cl&ear"           : clearMyQueryBttn.pushbutton()
  case mc = "&Sites"           : sitesPageBttn.pushbutton()
  case mc = "&First\tCtrl+F11" : myBar.firstButton.pushButton()
  case mc = "&Last\tCtrl+F12"  : myBar.lastButton.pushButton()
  case mc = "&Next\tF12"       : myBar.nextButton.pushButton()
  case mc = "&Previous\tF11"   : myBar.priorButton.pushButton()
  case mc = "&Contents"        : helpShowContext(helpFile, 20003)
  case mc = "&Using Help"      : helpOnHelp()
  case mc = "&About MAST"      : mastLib.setIsCalledTrue()
     if aboutFm.open("Abtmast.fsl") then
           ; modal dialog opened
           aboutFm.Wait()
           ; now close it
           aboutFm.Close()
           mastLib.setIscalledFalse()
     endIf
endswitch

endMethod
```

setFocus

DIVESITE.destsPage.setFocus

Displays the custom SpeedBar VIDEOBAR.FSL, opening it first, if necessary.

```
method setFocus(var eventInfo Event)

  var
   x, y, w, h LongInt
  endvar

  ; videoBar is already open
  if MastLib.isVCROpen()
    then
     if myBar.isAssigned()
       then
           ; Videobar open -- attach it to this form
         myBar.bringToTop()
           myBar.setCallFormName(getTitle(), 5500, 400)
     else
           ; attach to Videobar
           myBar.attach(mastlib.vcrName())
         myBar.bringToTop()
           myBar.setCallFormName(getTitle(), 5500, 400)
     endIf
  else
     ; Videobar not open -- open it
     myBar.Open("Videobar.fsl")
     myBar.setCallFormName(getTitle(), 5500, 400)
     mastLib.VCRisOpen()
  endIf

endMethod
```

pageMenu

DIVESITE.destsPage.pageMenu

Called by **destsPage.arrive** to display the Select Destinations menus.

```
proc pageMenu()

Var
 pageMenu Menu
 dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar

; this custom proc displays a menu for Sites page
; define menu
dropMenu1.addText("&Accept")
dropMenu1.addText("&Cancel")
dropMenu1.addText("Cl&ear")
pageMenu.addPopUp("Se&lections", dropMenu1)

pageMenu.addText("&Sites")

dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("&Previous\tF11")
pageMenu.addPopUp("Rec&ord", dropMenu2)

dropMenu3.addText("&Contents")
dropMenu3.addText("&Using Help")
dropMenu3.addText("&About MAST")
pageMenu.addPopUp("&Help", dropMenu3)

pageMenu.show()

endproc
```

DIVESITE.destsPage Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
var
  DestNo Number
  DestTc, TempDestTc   TCursor
endvar
```

DIVESITE.destsPage Uses Window

The *destPage* Uses window declares external routines used by this page. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
  setCallFormName(const CallerTitle String,
                              xPos LongInt, yPos LongInt)
  vcrName() String
  VCRisOpen()
  VCRisClosed()
  isVCROpen() Logical
endUses
```

pushButton

DIVESITE.destsPage.clearChoicesBttn.pushButton

Clears the current destination choices and sets the list count to 0, which empties the list.

```
method pushButton(var eventInfo Event)
  ; empty destList
  destlist.choices.list.count = 0
endMethod
```

pushButton

DIVESITE.destsPage.AddtoChoicesBttn.pushButton

Called by choosing the Add to List button *(AddToChoicesBttn).* Adds selected destinations from DEST.DB to *deslist.* These destinations are added to QBEVAL.DB when the Accept button is chosen.

```
method pushButton(var eventInfo Event)

var
  destVal String
  i, listCount SmallInt
endVar

if active.name = "Destination_Name"
  then
  listCount = destList.choices.list.count
  destVal = dest.Destination_Name.value

; make sure that the destination is not already in destlist
  for i from 1 to listCount
     destList.choices.list.selection = i
     if destList.choices.list.value = destVal
        then
           msgInfo("Already in the list.", destVal)
           return
     endIf
  endFor

; add item to destlist
  listCount = listCount + 1
  destList.choices.list.count = listCount
  destList.choices.list.selection = listCount
  destList.choices.list.value = destVal
endIf

endMethod
```

open

DIVESITE.destsPage.choices.open

Sets list.count property of the list object to 0, which empties the list.

```
method open(var eventInfo Event)
self.list.count = 0
endMethod
```

pushButton

DIVESITE.destsPage.sitesPageBttn.pushButton
Moves focus to *sitesPage*.

```
method pushButton(var eventInfo Event)
sitesPage.moveTo()
endMethod
```

pushButton

DIVESITE.destsPage.clearMyQueryBttn.pushButton

Clears current query values using MastLib.clearPageValues.

```
method pushButton(var eventInfo Event)

  clearChoicesBttn.pushButton()
  eventInfo.getTarget(uio)  ; what is the object?
  mastLib.clearPageValues(formName.name, mastlib.pageNameOf(uio))

endMethod
```

pushButton

DIVESITE.destsPage.cancelBttn.pushButton

Hides the custom SpeedBar and returns to DIVEPLAN.FSL with no further processing.

```
method pushButton(var eventInfo Event)

  ; Hide videobar
  myBar.Hide()
  ; return to Diveplan without any processing
  formreturn(True)

endMethod
```

pushButton

DIVESITE.destsPage.helpButton.pushButton

Displays Help by calling helpShowContext. The value 20003 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values.

```
method pushButton(var eventInfo Event)
helpShowContext(helpfile, 20003)
endMethod
```

pushButton

DIVESITE.destsPage.acceptBttn.pushButton

Called by choosing the Accept button. Builds the destination string from current selected destinations and passes it as Query Value to QBEVAL.DB.

```
method pushButton(var eventInfo Event)

var
   destString String
  tcDest, tc     TCursor
  i, listCount SmallInt
endVar

destString = "" ; initialize destination string

eventInfo.getTarget(uio)
mastlib.clearPageValues(formName.Name, Mastlib.pageNameOf(uio))

; we must now fill assign destinations to QBEVAL.DB
listCount = destList.choices.list.count
IF listCount > 0
  then     ; there are destinations
  tc.Open("QBEVal.db")
  if not tc.Locate("Query ID", 5)
     then msgStop("Problem", "Couldn't locate Destination field in QBEVal")
  else
     tc.Edit() ; edit QBEVAL.DB
     if listCount > 1
        then
        for i from 1 to listCount
                  ; build a destination list string
                  destList.choices.list.selection = i
                  if destString <> ""
               then
                        destString = destList.choices.list.value +" OR " +
  destString
                  else
                        destString = destList.choices.list.value
                  endIf
           endfor
     else
           destString = destList.choices.list.value
     endIf
; assign destinations string to Query Value
     tc."Query Value".value = destString
     tc.endEdit()
     tc.Close()
  endIf
endIf

; hide VideoBar
myBar.Hide()

formReturn(True)
```

```
endMethod
```

DIVESITE.sitesPage

Methods

arrive

menuAction

open

clearPageObjects

Procedures

pageMenu

Uses window

Uses

arrive

DIVESITE.sitesPage.arrive

Sets *sitesPage* title to "Sites" and displays a custom pop-up menu by calling the sitesPage.pageMenu procedure.

```
method arrive(var eventInfo MoveEvent)

  ; define title
  setTitle("Sites")
  ; display custom menu
  pageMenu()

endMethod
```

menuAction

DIVESITE.sitesPage.menuAction

Tests the string returned by a *sitesPage* menu selection and calls the appropriate method. For example, choosing the Accept item calls acceptBttn.pushButton.

```
method menuAction(var eventInfo MenuEvent)

  var
  mc String
  endvar

  ; what menu option has been chosen?
  mc = eventInfo.menuChoice()
  switch
    case mc = "&Accept"            : acceptBttn.pushbutton()
    case mc = "&Cancel"            : cancelBttn.pushbutton()
    case mc = "Cl&ear"             : clearMyQueryBttn.pushbutton()
    case mc = "&Destinations"      : destsPageBttn.pushbutton()
    case mc = "&Contents"          : helpShowContext(helpFile, 20004)
    case mc = "&Using Help"        : helpOnHelp()
    case mc = "&About MAST"        : mastLib.setIsCalledTrue()
           if aboutFm.open("Abtmast.fsl") then
                 ; modal dialog opened
                 aboutFm.Wait()
                 ; now close it
                 aboutFm.Close()
                 mastLib.setIscalledFalse()
           endIf

  endswitch

endMethod
```

open

DIVESITE.sitesPage.open

This form must be called from DIVEPLAN.FSL; it cannot be run directly. Opens the MASTLIB.LSL library if found in the WORK directory.

```
method open(var eventInfo Event)

; This open method will not open Divesite.fsl unless it
; has been called from Diveplan.fsl. The variable isCalled in
; Mastlib.lsl has to be set to True for Divesite to be opened.
; isCalled is set to True by mastlib.setIsCalledTrue() which is
; invoked by the pushbutton method on Diveplan.

delayScreenUpdates(Yes)
;open library (make sure that it is in the WORK directory)
if not mastLib.Open("MastLib", GlobalToDesktop) then
  msgStop("Failure", "Mastlib could not be opened.")
  formReturn(false)

else
  wasCalled = mastLib.returnIsCalled()
  if wasCalled = False then
  msgStop("Stop!", "You cannot open this form directly. It must be opened
  from Diveplan.fsl.")
  close()

  else
   ; associate form UIObject name with formName var
   formName.attach()

   ; Make sure "divers" alias is defined, otherwise
   ; dataSource property will use Sites.db in model. This would present
   ; a problem as sites in data model is detail table, which
   ; would constrain the dataSource property of sitesField.sitesList.
   addAlias("divers", "standard", getAliasPath("Work"))
  endIf

endIf
delayScreenUpdates(No)

endMethod
```

clearPageObjects

DIVESITE.sitesPage.clearPageObjects

Clears all field objects on this page by assigning them a value of "" (empty string).

```
method clearPageObjects(pageName String) Logical

; This custom method clears all objects on that page that have been
; selected.

var
  pageObjTc TCursor
  fieldObj UIObject
endVar

; list all the selectable fields on the page
mastLib.listPageObjects(formName.name, pageName)

; open tcursor on list of selectable fields
pageObjTc.open("Objlist.db")

; clear all selected fields
scan pageObjTc :
   fieldObj.attach(pageObjTc."Object Path")
  if fieldObj  <> ""
    then
     fieldObj = ""
  endIf
endScan

; clear form.page values in Qbeval.db
mastLib.clearPageValues(formName.Name, pageName)

RETURN TRUE

endMethod
```

pageMenu

DIVESITE.sitesPage.pageMenu

Called by sitesPage.arrive to display a custom pop-up menu.

```
proc pageMenu()

Var
 pageMenu Menu
 dropMenu1, dropMenu2 popUpMenu
endVar

; this custom proc displays a menu for Sites page
; define menu
dropMenu1.addText("&Accept")
dropMenu1.addText("&Cancel")
dropMenu1.addText("Cl&ear")
pageMenu.addPopUp("Se&lections", dropMenu1)

pageMenu.addText("&Destinations")

dropMenu2.addText("&Contents")
dropMenu2.addText("Using Help")
dropMenu2.addText("&About MAST")
pageMenu.addPopUp("&Help", dropMenu2)

pageMenu.show()

endproc
```

DIVESITE.sitesPage Uses Window

The *sitesPage* Uses window declares external routines used by this page. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal
   listPageObjects(formName String, pageName String) Logical
endUses
```

open

DIVESITE.sitesPage.sitesList.open

Fills this list with items from the Site Name field of SITES.DB.

```
method open(var eventInfo Event)

; Field filled from non-linked Sites.db. The alias :WORK: is
; used to define path of Sites.db, otherwise Sites.db in data model
; would be used.
self.dataSource = "[:WORK:Sites.Site Name]"

endMethod
```

pushButton

DIVESITE.sitesPage.acceptBttn.pushButton

Called using menuAction by choosing the Accept Button. Uses Mastlib.setPageValues to post the selected site values (Accommodations, Site Highlight, and Night Life) to the appropriate fields in QBEVAL.DB. The Average Temperature field requires special handling to map, for example, "cold" to "<=69".

```
method pushButton(var eventInfo Event)

var
   tempVal String
  Tc TCursor
endVar

; We need to do some special processing to post to the Accomodations,
; Site Highlight, Night Life and Avg Temperature fields in QBEVAL.DB.
; With the exception of Avg Temperature - which requires special treatment
  -
; the setPageValues() method in Mastlib.lsl is called to do the processing.

eventInfo.getTarget(uio)
MastLib.setPageValues(formName.name, mastlib.pageNameOf(uio))

tc.Open("QBEVal.db")
tc.Edit()

; locate QBEVAL.DB Avg Temperature record
if not tc.Locate("Query ID", 9)
  then
   msgStop("Problem", "Couldn't locate Avg Temperature record in QBEVal")
else
  tempVal = ""
  if cold <> ""
     then tempVal = "<=69"
  endIf
  if subTropical <> ""
     then
     if tempVal <> "" THEN
           tempVal = "<=79"
     else
           tempVal = ">=70, <=79"
     endIf
  endIf
  if tropical <> ""
    then
     if tempVal = "" THEN
           tempVal = ">=80"
     else
           tempVal= tempVal+" OR >=80"
     endIf
  endIf
  ; set the value
  tc."Query Value".value = tempVal
endIf
```

```
tc.endEdit()
tc.close()
; return to DivePlan
formReturn(True)

endMethod
```

pushButton

DIVESITE.sitesPage.clearMyQueryBttn.pushButton

Clears the current query values using MastLib.clearPageValues.

```
method pushButton(var eventInfo Event)

  ; what is the object?
  eventInfo.getTarget(uio) ; what is the object?

  ; clear field objects and also Qbeval.db values
  clearPageObjects(Mastlib.pageNameOf(Uio))

endMethod
```

pushButton

DIVESITE.sitesPage.cancelBttn.pushButton

Returns to DIVEPLAN.FSL with no further processing.

```
method pushButton(var eventInfo Event)

  ; return to diveplan without any processing
  formreturn(True)

endMethod
```

pushButton

DIVESITE.sitesPage.destsPageBttn.pushButton
Moves user to the *destsPage.*

```
method pushButton(var eventInfo Event)

  destsPage.moveto()

endMethod
```

pushButton

DIVESITE.sitesPage.helpButton.pushButton

Provides Help by calling helpShowContext. The value 20004 is a context ID for an item in the help file. In a larger application, it's a good idea to define constants for such values instead of hard-coding them.

```
method pushButton(var eventInfo Event)
helpShowContext(helpfile, 20004)
endMethod
```

lookList

Methods

open

close

menuAction

keyPhysical

mouseRightUp

showMe

Var, Const, and Uses windows

Var window

Uses window

Methods and procedures from MASTLIB.LSL

getLookUpTitle

getDataSource

setDataSource

open

lookList.open

```
; Paradox for Windows Sample Application 12/19/92

method open(var eventInfo Event)

var
  callForm Form
endVar

  ; This method makes sure LOOKLIST is opened by another form; if it
  ; isn't, then an error message appears and the form closes.

  if not eventInfo.isPreFilter() then ; Execute only for the form.
     If not formCaller(callForm) then
           beep()
           msgStop("Open Error", "You can't open this form directly." +
                   "  It must be opened by an appropriate DIVEPLAN " +
                   "form.")
           close()
     else
           mastLib.open("mastlib.lsl", GlobalToDesktop)
           formName.attach()
     endIf
  endif

endmethod
```

close

lookList.close

Closes the MASTLIB.LSL library.

```
method close(var eventInfo Event)
if eventInfo.isPreFilter()
  then
    ;code here executes for each object in form
  else
    ;code here executes just for form itself
    mastLib.close()
endIf
endMethod
```

menuAction

lookList.menuAction

This code executes when you use the control menu to close the form. It disables the default response and calls cancelButton.pushButton instead. As a result, closing the form via the control menu has the same effect as choosing the Cancel button.

```
method menuAction(var eventInfo MenuEvent)
if eventInfo.isPreFilter() then
  ; This code executes for each object on the form.
  if eventInfo.id() = menuControlClose then
     disableDefault
     cancelBttn.pushButton()
  endIf
else
  ; This code executes only for the form.

endIf
endMethod
```

keyPhysical

lookList.keyPhysical

Intercepts the *Esc* key to call cancelBttn.pushButton. Intercepts *Alt+A* to move to the List All button.

```
method keyPhysical(var eventInfo KeyEvent)
if eventInfo.isFirstTime() then
   if eventInfo.vCharCode() = VK_ESCAPE then
     disableDefault
     cancelBttn.pushButton()
  endIf

  if eventInfo.isAltKeyDown() then
     if eventInfo.vChar() = "A" then
           listAllBttn.moveTo()
     endIf
  endIf
endIf
endMethod
```

mouseRightUp

lookList.mouseRightUp

Calls the Mastlib library **createObjMenu** method when the right mouse button is released. This creates the Code Help menu. If you click on a Text or Bitmap object, the Help menu for the containing object is displayed because the objects have no code attached.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls mastlib.createObjMenu() to display a pop-up menu
; for the object you just right-clicked.

if eventInfo.isPreFilter()  then
   eventInfo.getTarget(uio) ; Which object was clicked?
                            ; If it's not the diveFlagBox bitmap,
                            ; find its container if it's a text object or a
  bitmap
    if uio.name <> "diveFlagBox"  then
      while
         uio.class = "Text" or uio.class = "Bitmap"
         uio.attach(uio.ContainerName)
      endWhile
   endIf
  mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
endIf

endMethod
```

showMe

lookList.showMe

Initializes the LookList dialog box form and makes it visible.

```
method showMe(const theTitle String, const listDataSource String)

  setTitle(theTitle)
  lookPg.lookUpFld.lookUpList.dataSource = listDataSource
  lookPg.lookUpFld.lookUpList.list.selection = 1 ; highlight first value in
  list

  if listDataSource = "[possdest.Destination Name]" then
     listAllBttn.visible = True
  else
     listAllBttn.visible = False
  endIf

  lookUpFld.moveTo()
  bringToTop()
endMethod
```

lookList Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
  mastLib Library
  uio UIObject
  formName Form
endVar
```

lookList Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPAL
  getLookUpTitle() String
  getDataSource() String
  setDataSource(const dataSrc String)
  createObjMenu(var formName Form,  pageName String, uio UIObject)
  pageNameOf(ui UIObject) STRING
endUses
```

pushButton

lookList.lookPg.listAllBttn.pushButton

This button is displayed as a check box. When the button is checked, *lookUpList* displays all the destination fields in DEST.DB; otherwise the listing is restricted to the selected destination fields in POSSDEST.DB.

```
method pushButton(var eventInfo Event)
  doDefault
  switch
     case self.value = True :
           lookupFld.lookupList.dataSource = "[dest.Destination Name]"
     case self.value = False :
           lookupFld.lookupList.dataSource = "[possdest.Destination Name]"
  endSwitch
endMethod
```

pushButton

lookList.lookPg.cancelBttn.pushButton

Returns control to the calling method with no further processing.

```
method pushButton(var eventInfo Event)
     formReturn("")
endMethod
```

pushButton

lookList.lookPg.okBttn.pushButton

Returns control to the calling method with the selected field value.

```
method pushButton(var eventInfo Event)
  formReturn(lookupFld.lookupList.list.value)
endMethod
```

lookList.lookPg.lookupFld

Methods

keyPhysical

newValue

keyPhysical

lookList.lookPg.lookupFld.keyPhysical

Intercepts the *Return* keystroke and call okBttn.pushButton.

```
method keyPhysical(var eventInfo KeyEvent)
if eventInfo.vCharCode() = VK_RETURN then
  disableDefault
  okBttn.pushButton()
endIf
endMethod
```

newValue

lookList.lookPg.lookupFld.newValue

Displays the new value of this field by calling message(self.value).

```
method newValue(var eventInfo Event)
message(self.value)
endMethod
```

mouseDouble

lookList.lookPg.lookupList.mouseDouble
Converts a double click to **okBttn.pushButton.**

```
method mouseDouble(var eventInfo MouseEvent)
  okBttn.pushButton() ; make double-click = push OK
endMethod
```

videoBar

Methods

open

keyChar

mouseRightUp

keyPhysical

setCallFormName

Var, Const, and Uses windows

Var window

Const window

Uses window

open

videoBar.open

```
; Paradox for Windows Sample Application 9/16/93

method open(var eventInfo Event)

var
  callForm form
  formList Array[] String
endVar

  if not eventInfo.isPreFilter() then
     if not mastLib.open("MastLib", globalToDesktop) then ;open library
                 msgStop("Failure", "Mastlib could not be opened.")
           close()
     else
        formName.attach()
           enumFormNames(formList)
           If not formList.contains("Main") then
                 beep()
                 msgStop("Open Error", "You can't open this form directly."
  +
                         "  It must be opened by an appropriate DIVEPLAN "
  +
                         "form.")
                 close()
           endIf
     endif
  endif

endmethod
```

keyChar

videoBar.keyChar

Intercepts the keystrokes "f", "p", "n", and "l" to invoke the pushButton methods of firstButton, priorButton, nextButton, and lastButton respectively.

```
method keyChar(var eventInfo KeyEvent)
switch
case eventInfo.char() = "f" : firstButton.pushButton()
case eventInfo.char() = "p" : priorButton.pushButton()
case eventInfo.char() = "n" : nextButton.pushButton()
case eventInfo.char() = "l" : lastButton.pushButton()
endswitch
endMethod
```

mouseRightUp

videoBar.mouseRightUp

Calls the Mastlib library createObjMenu method when the right mouse button is released. This creates the Code Help menu. If you click on a Text or Bitmap object, the Help menu for the containing object is displayed because the objects have no code attached.

```
method mouseRightUp(var eventInfo MouseEvent)

; Calls a mastlib.createObjMenu() method that displays an object
; level help menu.

; prevent repeat method calls
if eventInfo.isPreFilter() then
  ; set focus to Videobar
  bringToTop()
     eventInfo.getTarget(uio) ; what is the object?
  ;find ui container if object text or bitmap
     while uio.class = "Text" or uio.class = "Bitmap"
     uio.attach(uio.ContainerName)
     endWhile
     mastLib.createObjMenu(formName, mastlib.pageNameOf(uio), uio)
  ; return focus to caller form
  callerFm.bringToTop()
endIf

endMethod
```

keyPhysical

VIDEOBAR.keyPhysical

Intercepts keystrokes and translates them to method calls, if appropriate. The values 20002, 20003, and 20005 are context IDs for items in the help file. In a larger application, it's a good idea to define constants for such values.

```
method keyPhysical(var eventInfo KeyEvent)

; if F1 pressed, display help for calling form (callerFm)

if eventInfo.isFirstTime()
  then
     if eventInfo.vChar() = "VK_F1"
           then
           disableDefault
                 ; what form was videoBar opened by?
                 Switch
                 Case callerFm.name = "diveBio" : helpShowContext(helpfile,
  20002)
                 Case callerFm.name = "diveSite" :
  helpShowContext(helpfile, 20003)
                 Case callerFm.name = "diveWrck" :
  helpShowContext(helpfile, 20005)
                 Endswitch
  endIf
endIf

endMethod
```

setCallFormName

videoBar.setCallFormName

Sets the position of the custom SpeedBar depending on the name of the calling form.

```
method setCallFormName(const callerTitle String, xPos LongInt, yPos
  LongInt)

   ; position VCR depending on calling form
  var
     x, y, w, h LongInt
  endvar

  ; attach calling form to VCR bar
  callerFm.attach(callerTitle)

  ; get videobar position
     getPosition(x, y, w, h)
  ; reposition videobar per page parameters
   setPosition(xPos, yPos, w, h)

   ;  set focus on caller form
   callerFm.bringToTop()

endMethod
```

videoBar Var Window

Variables declared in the form's Var window are visible to all objects in the form.

```
Var
  callerFm Form
  uio UIObject
  formName Form
  mastLib Library
  wasCalled Logical
endVar
```

videoBar Const Window

Constants declared in the form's Const window are visible to all objects in the form.

```
Const
  helpFile = ":work:mastuser.hlp"
  ; path to Help file must be hard-coded (or file must be on path)
endConst
```

videoBar Uses Window

The Uses window declares external routines used by this form. The routines can be stored in a DLL, an ObjectPAL library, or another form. The following code declares methods stored in the ObjectPAL library MASTLIB. The first line includes the **ObjectPAL** keyword to indicate that the methods are written in ObjectPAL, rather than in some other programming language used to create a DLL.

```
Uses ObjectPal

  createObjMenu(var formName Form,
                 pageName String, uio UIObject)
  pageNameOf(ui UIObject) String
  returnIsCalled() Logical
   VCRisClosed()

endUses
```

pushButton

videoBar.videoPg.lastButton.pushButton

Selects the last record using callerFm.action(DataEnd). For example, if DIVEWRCK.FSL is the calling form, this method displays the last record of SHIPWRCK.DB.

```
method pushButton(var eventInfo Event)

  callerFm.bringToTop()
  callerFm.active.action(DataEnd)

endMethod
```

pushButton

videoBar.videoPg.nextButton.pushButton

Displays the next record using callerFm.action(DataNextRecord). For example, if DIVEBIO.FSL is the calling form, selects the next record of BIOLIFE.DB using callerFm.action(DataNextRecord) and displays all the information for that record.

```
method pushButton(var eventInfo Event)

  callerFm.bringToTop()
  callerFm.active.action(DataNextRecord)

endMethod
```

pushButton

videoBar.videoPg.priorButton.pushButton

Selects the previous record using callerFm.action(DataPriorRecord). For example, if DIVEBIO.FSL is the calling form, this method selects the previous record of BIOLIFE.DB using callerFm.action(DataNextRecord) and displays all the information for that record.

```
method pushButton(var eventInfo Event)

  callerFm.bringToTop()
  callerFm.active.action(DataPriorRecord)

endMethod
```

pushButton

videoBar.videoPg.firstButton.pushButton

Selects the first record using callerFm.action(DataBegin). For example, if DIVEBIO.FSL is the calling form, this method selects the first record of BIOLIFE.DB using callerFm.action(DataNextRecord) and displays all the information for that record.

```
method pushButton(var eventInfo Event)

  callerFm.bringToTop()
  callerFm.active.action(DataBegin)

endMethod
```

mastOrd

Methods

mouseEnter

mouseExit

mouseRightUp

keyPhysical

menuAction

Var, Const and Uses windows

Var window

Const window

Uses window

Methods and Procedures from MASTLIB.LSL

MASTObjMsg

Objects contained by MASTORD

#Page2

mastOrd Uses window

```
Uses objectPAL

  ; Custom method attached to LOOKLIST.FSL (form)

  showMe(const theTitle String, const listDataSource String)

  ; Custom methods contained in MASTLIB.LIB (library)

  mastOBJMsg(formName String, pageName String, objectID String)
  createObjMenu(var formName Form,  pageName String, uio UIObject)

endUses
```

See also
MASTObjMsg

mastOrd Var window

```
Var

  mastLIB Library  ; Pointer to MASTLIB.LIB, the library
                   ; of ObjectPAL routines that are shared
                   ; between this form, DIVEPLAN, and other
                   ; forms in this application directory

  okToExit Logical ; Indicates whether or not the form can
                   ; be closed.

endVar
```

See also

MAST Library

mastOrd Const window

```
Const

  ; Define the Help file constants

  helpFile   = ":WORK:mastuser.hlp" ; path to Help file
  headerHelp = LongInt(20006)
  detailHelp = LongInt(20007)


endConst
```

mouseEnter

mastOrd.mouseEnter

```
method mouseEnter(var eventInfo MouseEvent)

; This method displays a help message in the status line for each
; object as the user moves the pointer over it.

var

  uiTarget uiObject  ; Holds the name of the object that the pointer
                     ; is currently pointing to.

endVar

  if eventInfo.isPreFilter() then  ; Execute for each object on form.

     eventInfo.getTarget(uiTarget)
     mastLib.mastObjMsg("mastOrd", "thePage", uiTarget.Name)

  else                             ; Execute only for the form itself.

  endIf

endMethod
```

See also

MASTObjMsg

mouseExit

mastOrd.mouseExit

```
method mouseExit(var eventInfo MouseEvent)

; This method clears a pointer help message from the status bar

  if eventInfo.isPreFilter() then  ; Execute for each object on form.

     message("")

  else                             ; Execute only for the form itself.
                                   ; (This branch here because it was
  endIf                            ; automatically entered by Paradox
                                   ; when this method was created.  It
                                   ; remains for clarity.)

endMethod
```

mouseRightUp

mastOrd.mouseRightUp

```
method mouseRightUp(var eventInfo MouseEvent)

; This method displays an "object inspector" menu for
; several objects on the form.  These menus let the user
; browse the ObjectPAL source code, via a compiled WINHELP
; Help file.  To accomplish this, the method calls a custom
; method (createObjMenu()) in MASTLIB.

var

  uiTarget uiObject ; The object that was right-clicked
  thisForm Form     ; Form variable for MASTLIB.createObjMenu()
  Obj_Name String   ; Used to make sure we search for the
                    ; right object when we call createObjMenu.

endVar


  if eventInfo.isPreFilter() then  ; Executes for each object on the form

     ; First, get the object that was clicked

     eventInfo.getTarget(uiTarget)

     ; Because fields and buttons are made up of other objects, the object
     ; that received the mouse click may not be the one the user thinks it
     ; is.  So, we'll see if the name of the target object is one we gave
     ; it.  If not, the name will start with a pound sign (#) and we need
     ; to work with its container.

     obj_Name = uiTarget.Name
     While obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
           uiTarget.attach(uiTarget.containerName)
           obj_Name = uiTarget.Name
     endWhile

     ; uiTarget is now attached to a named object (or the form), so
     ; attach the form variable to the form that's currently running
     ; (namely, this one) and then call createObjMenu (to display the
     ; object inspector menu).

     thisForm.attach()
     mastLib.createObjMenu(thisForm, "thePage", uiTarget)

     ; If the user right-clicked an object that can be selected,
     ; then place an event in the queue that selects it.

     if uiTarget.class = "Field" then
           If uiTarget.tabStop then
                 self.postAction(uiTarget.moveTo())
           endIf
```

```
    endIf
  else                              ; Execute only for the form itself.
  endIf
endMethod
```

See also
createObjMenu

keyPhysical

mastOrd.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method contains custom code for selected keystrokes
; the user may hit, including the <Alt> shortcut keys (also
; known as "accelerators") and custom Help system created
; for this application.

var

  keyPress String ; Holds the key a user pressed as an
                  ; <Alt>+ shortcut key.

endVar


if eventInfo.isPreFilter() then
  ; This code executes for each object on the form.

  if eventInfo.isAltKeyDown() then
     disableDefault
     keyPress = eventInfo.vChar()
     switch
           case keyPress = "A" : add_Cust.pushButton()
           case keyPress = "B" : equipFrm.Rental_Sale.moveTo()
           case keyPress = "C" : editCust.pushButton()
           case keyPress = "D" : diveOrds.Destination.moveTo()
           case keyPress = "I" : equipFrm.Item_No.moveTo()
           case keyPress = "N" : diveOrds.Return_Date.moveTo()
           case keyPress = "P" : printBtn.pushButton()
           case keyPress = "Q" : equipFrm.Qty.moveTo()
           case keyPress = "S" : diveOrds.Ship_Via.moveTo()
           case keyPress = "T" : diveOrds.Depart_Date.moveTo()
           case keyPress = "U" : nameList.moveTo()
           case keyPress = "V" : diveOrds.No_Of_People.moveTo()
           case keyPress = "X" : exit_Btn.pushButton()
           case keyPress = "Y" : diveOrds.Payment_Method.moveTo()
           otherwise :
                 enableDefault
                 return
     endSwitch
     sleep()
  else
     If eventInfo.vCharCode() = VK_F1 then
           disableDefault
           helpShowContext(helpFile, headerHelp)
     endIf
  endIf
endIf
endMethod
```

menuAction

mastOrd.menuAction

```
method menuAction(var eventInfo MenuEvent)

; This method prevents users from minimizing (or maximizing) MASTORD.

  If not eventInfo.isPreFilter() then  ; Execute when the form receives
                                       ; the event.

     If eventInfo.id() = MenuControlMaximize or
        eventInfo.id() = MenuControlMinimize then

                disableDefault
                beep()
                message("Form cannot be resized.")
     else
           If eventInfo.reason() = menuNormal and
              eventInfo.id() = MenuCanClose and not okToExit then

                     eventInfo.setErrorCode(CanNotDepart)
                     disableDefault
                     message("Use the Exit button to exit MASTORD.")
           endIf
     endIf
  endIf

endmethod
```

MASTORD.#Page2

Methods

open

arrive

menuAction

Procedures

findOrds

Var and Const windows

Var window

Const window

Methods and Procedures from MASTLIB.LSL

open

Objects contained by #Page2

diveOrds

help_button

exit_button

printBtn

namelist

editCust

name

add_Cust

eq_Total

invTotal

vacTotal

equipFrm

MASTORD.#Page2 Var window

```
Var

  cust_TC,          ; Pointer to the customer table (DIVECUST)
  orderTC,          ; Pointer to the orders table (DIVEORDS)
  equipTC,          ; Pointer to the equipment table (DIVEITEM)
  stockTC  tCursor ; Pointer to the stock table (DIVESTOK)

endVar
```

MASTORD.#Page2 Const window

```
; These constants are for the menu commands and the Help file.
; The command matching each menu constant is given after the
; constant is defined.


Const


  rentRate = .15 ; Used to calculate the rental price of an
                 ; item (15% of Sale Price).

  ; Define the menu constants

   Ords_New = 101 ; Orders | New Order command
  Ords_Prt = 102 ; Orders | Print command
  OrdsExit = 103 ; Orders | Exit command

  Edit_Cut = 201 ; Edit | Cut command
  EditCopy = 202 ; Edit | Copy command
  EditPast = 203 ; Edit | Paste command

  RecFirst = 301 ; Record | First command
  Rec_Prev = 302 ; Record | Previous command
  Rec_Next = 303 ; Record | Next command
  Rec_Last = 304 ; Record | Last command
  Rec_Plus = 305 ; Record | Insert command
  Rec_Nuke = 306 ; Record | Delete command
  Rec_Canc = 307 ; Record | Cancel Changes command
  Rec_Look = 308 ; Record | LookUp Help command

  HelpHelp = 401 ; Help | Using Help command
  HelpList = 402 ; Help | Help Index command
  Help_Use = 403 ; Help | System Help command

  Find_Ord = 501 ; Orders | Locate | Order Number
  FindDest = 502 ; Orders | Locate | Destination
  FindMeth = 503 ; Orders | Locate | Payment Method
  FindShip = 504 ; Orders | Locate | Ship Via
  FindCust = 505 ; Orders | Locate | Customer Number
  FindName = 506 ; Orders | Locate | Customer Name
  FindCity = 507 ; Orders | Locate | City
  FindStat = 508 ; Orders | Locate | State
  FindItem = 509 ; Orders | Locate | Item No
  FindSale = 510 ; Orders | Locate | Rental/Sale
  Find_Qty = 511 ; Orders | Locate | Qty

endConst
```

open

mastOrd.#Page2.open

```
method open(var eventInfo Event)

  ; This method performs preliminary initialization and setup for this
  ; application.  It ensures the files needed by the application can be
  ; found, opens the required tCursors, and places the tables in Edit
  ; mode.

  ; First, place a message in the status bar and delay updates to the
  ; screen.

  Message("Loading MASTORD.  One moment, please...")

  delayScreenUpdates(Yes)

  ; Make sure the form has been opened in the directory
  ; its files are stored in.  This offers some assurance that all
  ; necessary files are in the working directory, but
  ; a more complex text would be needed to make absolutely
  ; sure.

  if not isFile("mastord.fsl") then
     disableDefault
     msgStop("Directory Error", "The Paradox for Windows " +
             "working directory must be set to the directory " +
             "containing this form, for example: " +
              "C:\\PDOXWIN\\DIVEPLAN.")
     formReturn(0)
  endIf


  ; Set the default date format to Windows Short, tell Paradox
  ; to treat blank values as zeros, and open MASTLIB.

  If formatExist("Windows Short") then
     formatSetDateDefault("Windows Short")
  else
     disableDefault
     beep()
     msgStop("Can't Find Date Format", "You must have " +
             "the Windows Short date format defined " +
             "before you can use this application.")
     close()
  endIf

  blankAsZero(Yes)
  mastLib.open("MASTLIB", globalToDesktop)

  ; Open the tCursors for diveOrds and diveCust.  For
  ; information about how these are used, see the
  ; Var method of the form and the action methods for
  ; the Vacation Information panel (diveOrds) and the
```

```
  ; Customer Info panel (diveCust).

  orderTC.attach(Order_No)
  cust_TC.open(":WORK:DIVECUST.DB")
  equipTC.open(":WORK:DIVEITEM.DB")
  stockTC.open(":WORK:DIVESTOK.DB")

  ; Perform normal processing

  doDefault

  ; This next line places an event at the end of the
  ; current event queue.  This event places the form
  ; in Edit mode.  We place it at the end of the
  ; queue so that it doesn't interfere with the actions of this event.

  self.postAction(dataBeginEdit)
  delayScreenUpdates(No)

endMethod
```

See also

open

arrive

mastOrd.#Page2.arrive

```
method arrive(var eventInfo MoveEvent)

; This method creates and displays the menus used for this
; form.

var

  mainMenu Menu       ; The main menu
  OrdsMenu,           ; The Orders popup menu
  EditMenu,           ; The Edit popup menu
  Rec_Menu,           ; The Record menu
  HelpMenu,           ; The Help menu
  FindMenu popupMenu  ; The Orders | Locate menu

endVar

  ; First, let's define each menu:

  FindMenu.addText("&Order Number", menuEnabled, userMenu + find_ord)
  FindMenu.addText("&Destination", menuEnabled, userMenu + findDest)
  FindMenu.addText("&Payment Method", menuEnabled, userMenu + findMeth)
  FindMenu.addText("&Ship Via", menuEnabled, userMenu + findShip)
  FindMenu.addSeparator()
  FindMenu.addText("&Customer Number", menuEnabled, userMenu + findCust)
  FindMenu.addText("Customer &Name", menuEnabled, userMenu + findName)
  FindMenu.addText("Ci&ty", menuEnabled, userMenu + findCity)
  FindMenu.addText("St&ate", menuEnabled,userMenu + findStat)
  FindMenu.addSeparator()
  FindMenu.addText("&Item Number", menuEnabled, userMenu + findItem)
  FindMenu.addText("&Rental/Sale", menuEnabled, userMenu + findSale)
  FindMenu.addText("&Qty", menuEnabled, userMenu + find_Qty)

  OrdsMenu.addPopUp("&Locate", FindMenu)
  OrdsMenu.addText("&Print", menuEnabled, userMenu + ords_Prt)
  OrdsMenu.addSeparator()
   OrdsMenu.addText("&New Order", menuEnabled, userMenu + ords_New)
  OrdsMenu.addSeparator()
  OrdsMenu.addText("E&xit", menuEnabled, userMenu + ordsExit)
  MainMenu.addPopUp("&Orders", OrdsMenu)

  EditMenu.addText("Cu&t\tShift+Del", menuDisabled + menuGrayed, userMenu +
  edit_Cut)
  EditMenu.addText("&Copy\tCtrl+Ins", menuDisabled + menuGrayed, userMenu +
  editCopy)
  EditMenu.addText("&Paste\tShift+Ins", menuDisabled + menuGrayed, userMenu
  + editPast)
  MainMenu.addPopUp("&Edit", EditMenu)

  Rec_Menu.addText("&First\tCtrl+F11", menuEnabled, userMenu + recFirst)
  Rec_Menu.addText("&Previous\tF11", menuEnabled, userMenu + rec_Prev)
  Rec_Menu.addText("&Next\tF12", menuEnabled, userMenu + rec_Next)
```

```
  Rec_Menu.addText("&Last\tCtrl+F12", menuEnabled, userMenu + rec_Last)
  Rec_Menu.addSeparator()
  Rec_Menu.addText("Lookup &Help\tCtrl+Space", menuEnabled, userMenu +
  rec_Look)
  Rec_Menu.addText("&Insert\tInsert", menuEnabled, userMenu + rec_Plus)
  Rec_Menu.addText("&Delete\tCtrl+Del", menuEnabled, userMenu + rec_Nuke)
  Rec_Menu.addSeparator()
  Rec_Menu.addText("&Cancel Changes\tAlt+BkSp", menuEnabled, userMenu +
  rec_Canc)
  MainMenu.addPopUp("&Record", Rec_Menu)

  HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu +
  helpHelp)
  HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu +
  helpList)
  HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu +
  help_Use)
  MainMenu.addPopUp("&Help", HelpMenu)

  ; Display the main menu.

  mainMenu.show()

endMethod
```

menuAction

mastOrd.#Page2.menuAction

```
method menuAction(var eventInfo MenuEvent)

; This method displays, updates, and handles the custom menus
; used in this form.

var

  menu_Cmd smallInt ; The constant returned by a menu command
  editTest Logical  ; Flag variable used to test the type of
                    ; object that is active
  findTest String   ; Flag variable that holds the results of
                    ; a search attempt
  calledBy Form     ; Holds the handle of the form that opened
                    ; this form



endVar

  menu_Cmd = eventInfo.id()
  switch

     ; First, trap for the instant before the menu appears, so we can
     ; change the display attributes of certain menu commands when
  appropriate.

     case menu_Cmd = menuInit :

           try
                 editTest = active.Editing
           onFail
                 editTest = False
           endTry

           if editTest then
                 if active.selectedText <> "" then
                       setMenuChoiceAttributeByID(userMenu + edit_Cut,
  menuEnabled)
                       setMenuChoiceAttributeByID(userMenu + editCopy,
  menuEnabled)
                 else
                       setMenuChoiceAttributeByID(userMenu + edit_Cut,
  menuDisabled + menuGrayed)
                       setMenuChoiceAttributeByID(userMenu + editCopy,
  menuDisabled + menuGrayed)
                 endIf
           endIf

     case menu_Cmd = MenuControlClose :

           disableDefault
```

```
         exit_Btn.pushButton()

   case menu_Cmd = MenuControlMaximize or
        menu_Cmd = MenuControlMinimize :

         If formCaller(calledBy) then
               disableDefault
               beep()
               msgInfo("Oops!", "Because this form was opened by the " +
                       calledBy.getTitle() + "form, the Maximize and " +
                       "Minimize buttons are disabled.  Sorry...")
         endIf

   case menu_Cmd = userMenu + Find_Ord : FindTest =
FindOrds(":WORK:DIVEORDS", "Order No")
   case menu_Cmd = userMenu + FindDest : FindTest =
FindOrds(":WORK:DIVEORDS", "Destination")
   case menu_Cmd = userMenu + FindMeth : FindTest =
FindOrds(":WORK:DIVEORDS", "PaymentMethod")
   case menu_Cmd = userMenu + FindShip : FindTest =
FindOrds(":WORK:DIVEORDS", "Ship Via")
   case menu_Cmd = userMenu + FindCust : FindTest =
FindOrds(":WORK:DIVEORDS", "Customer No")
   case menu_Cmd = userMenu + FindName : FindTest =
FindOrds(":WORK:DIVECUST", "Name")
   case menu_Cmd = userMenu + FindCity : FindTest =
FindOrds(":WORK:DIVECUST", "City")
   case menu_Cmd = userMenu + FindStat : FindTest =
FindOrds(":WORK:DIVECUST", "State/Prov")
   case menu_Cmd = userMenu + FindItem : FindTest =
FindOrds(":WORK:DIVEITEM", "Item No")
   case menu_Cmd = userMenu + FindSale : FindTest =
FindOrds(":WORK:DIVEITEM", "Rental/Sale")
   case menu_Cmd = userMenu + Find_Qty : FindTest =
FindOrds(":WORK:DIVEITEM", "Qty")

    case menu_Cmd = userMenu + Ords_New :
         diveOrds.Destination.moveTo()
      If diveOrds.Destination.action(dataPostRecord) then
               diveOrds.Destination.postAction(dataInsertRecord)
         else
               beep()
               msgInfo("Can't Insert New Order", "Make sure the current
order" +
                       "can be saved.  (To cancel any changes, choose
Record | " +
                       "Cancel Changes.)")
         endIf

   case menu_Cmd = userMenu + Ords_Prt : printBtn.pushButton()
   case menu_Cmd = userMenu + OrdsExit : exit_Btn.pushButton()
   case menu_Cmd = userMenu + Edit_Cut :

         active.menuAction(menuEditCut)
         setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
```

```
      case menu_Cmd = userMenu + EditCopy :

            active.menuAction(menuEditCopy)
            setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)

      case menu_Cmd = userMenu + EditPast : active.menuAction(menuEditPaste)
      case menu_Cmd = userMenu + RecFirst : active.postAction(dataBegin)
      case menu_Cmd = userMenu + Rec_Prev :
   active.postAction(dataPriorRecord)
      case menu_Cmd = userMenu + Rec_Next :
   active.postAction(dataNextRecord)
      case menu_Cmd = userMenu + Rec_Last : active.postAction(dataEnd)
      case menu_Cmd = userMenu + Rec_Plus :
   active.postAction(dataInsertRecord)
      case menu_Cmd = userMenu + Rec_Nuke :
   active.postAction(dataDeleteRecord)
      case menu_Cmd = userMenu + Rec_Canc :
   active.postAction(dataCancelRecord)
      case menu_Cmd = userMenu + Rec_Look : active.postAction(dataLookup)
      case menu_Cmd = userMenu + HelpHelp : helpOnHelp()
      case menu_Cmd = userMenu + HelpList : helpShowIndex(helpFile)
      case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile,
   headerHelp)

   endSwitch

   If FindTest.isAssigned() then
      If FindTest = "notFound" then
            msgInfo("Search Unsuccessful", "Make sure you " +
                    "entered a search value valid for the " +
                    "field you chose to search.")
      else
            diveOrds.Destination.locate("Order No", findTest)
      endIf
   endIf

endMethod
```

findOrds

mastOrd.#Page2.findOrds

```
proc FindOrds(tbl_Name String, fld_Name String) String

; This procedure lets a user locate an order using a
; number of different criteria.  It asks the user for
; the search value, opens the appropriate table, and
; performs the locate.  If the locate is successful,
; the order number is returned; otherwise, the string
; "notFound" is returned.

; This procedure expects two parameters:
; tbl_Name is a String containing the name of the
; table to search.
; fld_Name is a String containing the name of the
; field to be searched.

var

  Find_Val anyType  ; The value the user wants to
                    ; locate

  theOrder LongInt  ; The order number of the located
                    ; record.

  Response Logical  ; The result of the locate
  searchTC tCursor  ; tCursor for the table being
                    ; searched

endVar

  If not searchTC.open(tbl_Name) then
     return "badTable"
  endIf

  find_Val = ""
  find_Val.view("Enter " + Fld_Name + " to locate")

  If tbl_Name = ":WORK:DIVECUST" then
     if searchTC.locate(fld_Name, find_Val) then
           find_Val = String(smallInt(searchTC."Customer No"))
           fld_Name = "Customer No"
           searchTC.open(":WORK:DIVEORDS")
     else
           return "notFound"
     endIf
  endIf


  If not searchTC.locate(fld_Name, find_Val) then
     return "notFound"
  Else
     Return searchTC."Order No"
```

```
  endIf
endProc
```

173

MASTORD.#Page2.diveOrds

diveOrds has no methods attached to it.

Var window
Var window

Objects contained by diveOrds
#Record17

MASTORD.#Page2.diveOrds Var window

```
Var

  allDests,          ; Contains all the possible destinations
                     ; a traveler can visit (used in the
                     ; newValue method of destination)
  vacDests tCursor   ; Contains the destinations the user
                     ; can choose from

endVar
```

MASTORD.#Page2.#Record17

Methods

open

arrive

canDepart

action

Var window

Var window

Objects contained by Record17

weeksOut

destCost

Ship_Cost

Ship_Via

cardInfo

Payment_Method

Box31

MASTORD.#Page2.#Record17 Var window

```
Var

  userNuke Logical  ; Flag to keep the application from
                    ; verifying vacation data when a user
                    ; chooses to delete an order record.

endVar
```

open

mastOrd.#Page2.#Record17.open

```
method open(var eventInfo Event)

  doDefault
  if not vacDests.open("POSSDEST") then
     vacDests.open("SITES")
  endIf
  if vacDests.nRecords() < 1 then
     vacDests.open("DEST")
  endIf
  allDests.open("DEST")

endMethod
```

arrive

mastOrd.#Page2.#Record17.arrive

```
method arrive(var eventInfo MoveEvent)

var

  reasonID smallInt ; Holds the id of the reason why this method was
                    ; called.  (Gives a slight edge in performance)
endVar

; When the user arrives on a new record in this multi-record object,
; this method performs initialization work.

  delayScreenUpdates(Yes)
  reasonID = eventInfo.reason()

  ; Check the reason this method was called and perform code appropriate
  ; for different reasons.

  switch

     case reasonID = userMove or reasonID = palMove :

           ; The method was called by the user or in response to ObjectPAL
           ; code that executed because of a user action.  Some of the
           ; fields need to be recalculated to match the new record.

           nameList.Value = Name.Value
           eq_Total.postAction(dataRecalc)

           ; Now, see if we need to display a vertical scroll bar or not.
  The
           ; scroll bar appears if there are more than five records in the
           ; table frame
           ; otherwise, it disappears.

           if equipFrm.nRecords > equipFrm.nRows then
                 equipFrm.VerticalScrollBar = True
           else
                 equipFrm.VerticalScrollBar = False
           endIf

     case reasonID = startupMove :

           ; This method was called because the form was opened.  This
  block
           ; of code ensures a few other calculated fields get initialized
           ; correctly.

           destCost.postAction(dataRecalc)
           City.postAction(dataRecalc)
           Zip_Postal_Code.postAction(dataRecalc)
```

```
  endSwitch
  delayScreenUpdates(No)

endMethod
```

canDepart

mastOrd.#Page2.#Record17.canDepart

```
method canDepart(var eventInfo MoveEvent)

; This method validates the vacation data by checking for blank
; fields.

var

  badField String   ; The name of the bad field, as seen by the user on the
  form
  uiTarget uiObject ; The name of the bad field, as seen by Paradox

endVar

  ; We only want to perform this validation if the order has an
  ; order number (which indicates that it was deliberately created
  ; by the user and not accidentally added by moving past the
  ; bottom record) and the user _hasn't_ chosen to delete it.

  If not Order_No.isBlank() and not userNuke then

     ; We need to check the order, so look at each required field
     ; If there is a blank field that needs information, then
     ; store the name that the user sees to a variable for the error
     ; message and store the name of the field to a uiObject variable
     ; so we can make it active after the user clears the error message

     switch

           case Destination.isBlank() : ; Did the user enter a Destination?
                 badField = "Destination"
                 uiTarget.attach(diveOrds.Destination)

           case No_Of_People.isBlank() : ; Did the user enter the number of
  Travelers?
                 badField = "Travelers"
                 uiTarget.attach(diveOrds.No_of_People)

           case Depart_Date.isBlank() : ; Did the user enter a Depart Date?
                 badField = "Depart Date"
                 uiTarget.attach(diveOrds.Depart_date)

           case Return_Date.isBlank() : ; Did the user enter a Return Date?
                 badField = "Return Date"
                 uiTarget.attach(diveOrds.Return_Date)

           case Payment_Method.isBlank() : ; Did the user enter a Payment
  Method?
                 badField = "Payment Method"
                 uiTarget.attach(diveOrds.Payment_Method)

     endSwitch
```

```
    ; If badField has a value, then we know there was at least one
    ; field that was left blank.  Given that, display an error message,
    ; make the bad field active, and then exit the method.

    If badField.isAssigned() then
          msgInfo("Incomplete Vacation Order", "Enter " +
                  "a value for " + badField + ".")
          eventInfo.setErrorCode(-1)
          uiTarget.moveTo()
          return
    endIf

  else

    ; This prevents the user from entering data into records that are
    ; not explicitly inserted.  This code is needed because Paradox
    ; automatically appends a blank record at the bottom of a table when
    ; you move past the last record of the table.  While this is a time-
    ; saving feature in most cases, it can lead to records that do not
    ; have key values in this application.  So, we'll override this
    ; behavior by moving to the last record in the table.

    If diveOrds.Order_No.isBlank() then
          self.postAction(dataEnd)
    endIf
  endIf

endMethod
```

action

mastOrd.#Page2.#Record17.action

```
method action(var eventInfo ActionEvent)

; This method controls record-level actions for this group
; of fields.  For example, if the user chooses to insert a
; new order, this method calculates the new order number.
; If the user chooses to delete an order, we want to display
; a confirmation dialog.

var

  actionID smallInt  ; Holds the id of the user's event
  Response String    ; Holds the user's response to a dialog box

endVar

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  actionID = eventInfo.id()

  switch

     ; The user chose to insert a new order, so we need to
     ; calculate a new order number.

     case actionID = dataInsertRecord :

           doDefault
           Order_No = maxOrder

     ; The user chose to delete a record, so let's pop up
     ; a dialog making sure they really want to delete it.

     case actionID = dataDeleteRecord :

           Response = msgQuestion("Are you sure?", "If you " +
                      "choose Yes, this order will be permanently " +
                      "deleted.  Choose No to Cancel.")

           ; If the user chose not to delete the record, we
           ; want to prevent anything else from happening.
           ; Otherwise, we'll set a flag which disables the
           ; validity checking in the canDepart method.

           If Response = "No" then
                 eventInfo.setErrorCode(-1)
           else
                 userNuke = True
           endIf

     ; If the user chose to unlock a record, we want to
```

```
   ; follow it if it flies away.  The easiest way to
   ; do this, in this form, is to force a dataPostRecord,
   ; which is similar to the Record | Post/Keep Locked
   ; command.

   case actionID = dataUnlockRecord or
        actionID = dataToggleLockRecord:

         ; Because this could be caused by the user locking
         ; the record (using Record | Lock), we want
         ; to post the record only if it's already locked.

         If active.locked then

               ; Make sure the total fields in DIVEORDS are current.
               ; If not, update them before unlocking or posting the
               ; record.

               If vacationCost.Value <> vacTotal.Value then
                     vacationCost.Value = vacTotal.Value
               endIf

               If subTotal.Value <> eq_Total.Value then
                     subTotal.Value = eq_Total.Value
               endIf

               If Total_Invoice.Value <> InvTotal.Value then
                     Total_Invoice.value = InvTotal.Value
               endIf

               ; Force the record to be posted.  This is an easy way
               ; to work with "flyAway."

               If not action(dataPostRecord) then
                     eventInfo.setErrorCode(-1)
               endIf
         endIf

   case eventInfo.id() = dataPostRecord :

         if self.touched then
               doDefault
               if errorCode() <> 0 then
                     If errorCode() = -1 then
                           msgInfo("Can't Save Current Order", "Reason: "
+
                                 "Missing or invalid information.")
                     else
                           msgInfo("Can't Save Current Order", "Reason: "
+
                                 errorMessage() + ".")
                     endIf
                     eventInfo.setErrorCode(-1)
               endIf
         endIf
```

```
     case actionID = dataArriveRecord :

           userNuke = False

  otherwise : doDefault

  endSwitch

  delayScreenUpdates(No)
  setMouseShape(mouseArrow)


endMethod
```

calcField

mastOrd.#Page2.weeksOut.calcField

```
iif((Depart_Date.isBlank()) or (Return_Date.isBlank()), 0,
  Ceil(Number(Return_Date - Depart_Date) / 7))
```

newValue

mastOrd.#Page2.destCost.newValue

```
method newValue(var eventInfo Event)

  vacTotal.postAction(dataRecalc)

endMethod
```

newValue

mastOrd.#Page2.Ship_Cost.newValue

```
method newValue(var eventInfo Event)

  if eventInfo.reason() <> startupValue then
     eq_Total.postAction(dataRecalc)
  endIf

endMethod
```

calcField

mastOrd.#Page2.cardInfo.calcField

```
[DIVEORDS.CcNumber] + IIF([DIVEORDS.CcNumber] = "", "", " - ") +
  [DIVEORDS.CcExpDate]
```

MASTORD.#Page2.Ship_Via

Methods

keyPhysical

action

keyPhysical

mastOrd.#Page2.Ship_Via.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)
; This method overrides the default behavior of certain
; keystrokes and selects a specific field object, depending
; on the keystroke pressed.

var

  keyPress smallInt ; Holds the key pressed by the user

endVar

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  keyPress = eventInfo.vCharCode()
  disableDefault
  switch

     case keyPress = VK_DOWN or
          keyPress = VK_TAB   : equipFrm.Item_No.moveTo()
     case keyPress = VK_RIGHT : nameList.moveTo()

     otherwise: enableDefault

  endSwitch
  delayScreenUpdates(No)
  setMouseShape(mouseArrow)

endMethod
```

action

mastOrd.#Page2.Ship_Via.action

```
method action(var eventInfo ActionEvent)

  If eventInfo.id() = dataLookup then
     doDefault
     nameList.moveTo()
  endIf

endMethod
```

MASTORD.#Page2.Payment_Method

Methods

open

action

changeValue

Var window

Var window

MASTORD.#Page2.Payment_Method Var window

```
Var

  payTypes popupMenu

endVar
```

open

MASTORD.#Page2.Payment_Method.open

```
method open(var eventInfo Event)

; This method defines the popup menu of payment types
; (used in this field's action statement.  If you change
; this list, be sure to also change this field's Picture
; valcheck in DIVEORDS; otherwise, you will not be able
; to save your new entries to the table.

  payTypes.addText("Cash")
  payTypes.addText("Check")
  payTypes.addText("Money Order")
  payTypes.addSeparator()
  payTypes.addText("AmEx")
  payTypes.addText("Diners Club")
  payTypes.addText("Discover")
  payTypes.addText("MasterCard")
  payTypes.addText("Visa")

endMethod
```

action

MASTORD.#Page2.Payment_Method.action

```
method action(var eventInfo ActionEvent)

; This method lets the user choose a payment method from a
; popup menu (defined in this field's Open method).  To
; display this menu when this field is active, press
; <Ctrl+Space>.


var

  Response String      ; Holds the user's choice

endVar

  ; If the user asked for data lookup (by pressing <Ctrl+
  ; Space>, then display the popup menu, save the user's
  ; response to a variable, and make sure that response
  ; isn't blank (meaning the menu was canceled) and that
  ; it isn't the same value that's already in the field.

  if eventInfo.id() = DataLookup then
     Response = String(payTypes.show())
     If Response <> "" and Response <> self.value then

           ; The user did choose a new value, so assign it
           ; to the field, then move to the next field in
           ; the tab order.  (This causes the new value to
           ; be committed and triggers the changeValue
           ; method on this field, which verifies the actual
           ; value and reacts accordingly.)

           self.value = Response
           self.action(FieldForward)
     endIf
  endIf


endMethod
```

changeValue

MASTORD.#Page2.Payment_Method.changeValue

```
method changeValue(var eventInfo ValueEvent)

; This method checks the new payment method and gets
; (or clears) the credit card information field as needed.
; We use this method because it is only called when a user
; enters a new value and lets us validate the new value
; before it gets saved to the table.

var

  new_Numb String   ; Holds the new credit card number
  new_ExpD Date     ; Holds the new expiration date
  new_Type String   ; Holds the type the user wants
                    ; posted to the field

endVar

  ; First, save the pending payment method to a variable.
  ; This improves performance because we only have to call
  ; eventInfo.newValue() once, yet we need to use the newValue
  ; several times.

  new_Type = eventInfo.newValue()

  ; Now, test the new value and react accordingly.

  If new_Type = "Cash" or new_Type = "Check" or
     new_Type = "Money Order" then

     doDefault  ; Save the new value

     ; User didn't select a credit card, so check to see
     ; if there is existing credit card information.  If
     ; so, clear it.

     If not ccNumber.isBlank() then
           ccNumber.value = blank()
     endIf

     If not ccExpDate.isBlank() then
           ccExpDate.value = blank()
     endIf

  Else

     ; User selected a credit card, so we need to get
     ; (and validate) a credit card number and expiration.
     ; If there are values in these fields already, we'll
     ; initialize the variables to those values.

     ; Because there are a lot of different ways this
```

```
   ; process could fail, we'll assume it will and
   ; then approve the change in those cases where
   ; all the verification checks pass.  (This leads
   ; to less code and better performance.)

   disableDefault

   ; Start with and get the credit card number.  If
   ; that checks out, continue with the expiration date.

   If not ccNumber.isBlank() then  ; If there's a card number
         new_Numb = ccNumber.value    ; already, use it as the default
   else                           ; otherwise, default to a blank
         new_Numb = ""                ; value.
   endIf

   new_Numb.view("Enter Credit Card Number")

   ; Here, we make sure the user entered a new credit card number.
   ; If they didn't, then we won't accept the new payment method.

   If new_Numb = ccNumber.value then
         msgInfo("Canceling new Payment Method", "Card number not
changed.")
         self.action(EditUndoField)
         return
   endIf

   If not new_Numb.isBlank() and new_Numb.size() < 21 then

         ; User entered a string that will fit in the field.
         ; Since we can't test it beyond that, we'll assume
         ; it's a good number and continue with the expiration date.

         If not ccExpDate.isBlank() then   ; Use existing expiration
               new_ExpD = ccExpDate.value     ; date as the default or
         else                              ; today's date, whichever
               new_ExpD = today()             ; is appropriate.
         endIf

         new_ExpD.view("Enter Expiration Date")

         ; Now, make sure the user entered a new expiration date.
         ; If not, then we won't accept the new payment method.

         If new_ExpD = ccExpDate.value then
               msgInfo("Canceling new Payment Method", "Expiration Date
not changed.")
               self.action(EditUndoField)
               return
         endIf

         ; Verify the user entered a date value.  If they
         ; didn't, then display an error and cancel the
         ; rest of this method.
```

```
            try
                  date(new_ExpD)
            onFail
                  msgInfo("Invalid Date Format", "Couldn't convert '" +
                          String(new_ExpD) + "' to a date value.  Be " +
                          "sure to use the correct format, e.g. " +
                          String(today()))
                  return
            endTry

            ; The user did enter a date value, so make sure the
            ; card hasn't already expired.  If so, display an
            ; error.

            If new_ExpD < today() then
                  msgInfo("Invalid Expiration Date", "Enter a " +
                          "date that is equal to or later than " +
                          "the current date (" + String(today()) +
                          ".)")
            else

                  ; The date is good, so post the new payment
                  ; method and update the card information.

                  doDefault                  ; Save payment method
                  ccNumber.Value = new_Numb  ; Save new card number
                  ccExpDate.Value = new_ExpD ; Save new expiration

            endIf
      endIf
   endIf

   ; Now, we need to update the calculated field that shows
   ; the credit card info.

   cardInfo.postAction(dataRecalc)


endMethod
```

MASTORD.#Page2.#Box31

Methods

setPrice

Objects contained by #Box31

No_Of_People

Destination

Return_Date

Depart_Date

setPrice

mastOrd.#Page2.#Box31.setPrice

```
method setPrice()

; This method updates the rental price of any ordered equipment items when
; the user changes the Depart or Return dates of a vacation order.

var

  equip_TC tCursor  ; A pointer to the equipment items for this order

  currItem Number   ; Holds the item number of each piece of equipment
  during the
                    ; SCAN...ENDSCAN loop.
endVar


  ; First, attach the tCursor to the equipment items shown for this
  ; order.

  equip_TC.attach(equipFrm.Item_No)

  ; Next, start a loop that performs the update calculations for each
  ; record in the restricted view (the equipment items shown in the
  ; equipFrm table frame.

  scan equip_TC :

     currItem = equip_TC."Item No"       ; Save the current item number
     stockTC.qLocate(equip_TC."Item No") ; Locate the item in DIVESTOK

      ; If this equipment item order is a rental, then update the sale
     ; price.

     If equip_TC."Rental/Sale" = "Rental" then
           equip_TC."Price" = weeksOut * rentRate * stockTC."Sale Price"
     endIf

  endScan

   ; Make sure the first record in the restricted view is current.

  equip_TC.Home()

  ; Update the table frame with the prices that were calculated in the
  ; tCursor.

  equipFrm.resync(equip_TC)

endMethod
```

MASTORD.#Page2.No_Of_People

Methods

keyPhysical

changeValue

keyPhysical

mastOrd.#Page2.No_Of_People.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method overrides the default behavior of the
; <Right Arrow> key for this field.  We want this key to
; select the Customer Name field, instead of displaying
; the next vacation record.

  If eventInfo.vCharCode() = VK_RIGHT then
     disableDefault
     nameList.moveTo()
  endIf

endMethod
```

changeValue

mastOrd.#Page2.No_Of_People.changeValue

```
method changeValue(var eventInfo ValueEvent)

  doDefault
  If eventInfo.errorCode() <> 0 then
     errorShow()
  else
     vacTotal.postAction(dataRecalc)
  endIf

endMethod
```

MASTORD.#Page2.Destination

Methods

action

changeValue

newValue

action

mastOrd.#Page2.Destination.action

```
method action(var eventInfo ActionEvent)

; This method provides lookup help for the Destination
; using a a custom form (LOOKLIST.FSL) and a method
; attached to that form (showMe()).

var

  lookForm Form   ; LOOKLIST.FSL is a custom data lookup form
  Response String ; Holds the user's response to the lookup

endVar


  ; If user pressed <Ctrl+Space> (or

  If eventInfo.id() = dataLookUp then


     ; First, see if LOOKLIST is already loaded into memory.
     ; If not, then

     lookForm.open("LOOKLIST.FSL")
     lookForm.showMe("Enter Vacation Destination",
                     String(vacDests.tableName() +
                     ".Destination Name"))

     Response = String(lookForm.wait())

     try
           lookForm.hide()
     onFail
           if errorCode() = peFormClosed then
                 ; If the user closed the dialog box, there's no need to
  hide it.
           endIf
     endTry

     If Response <> "" and Response <> self.Value then
           self.Value = Response
           self.postAction(fieldForward)
     endIf
  endIf

endMethod
```

changeValue

mastOrd.#Page2.Destination.changeValue

```
method changeValue(var eventInfo ValueEvent)

  ; If the new Destination isn't blank, then we want to grab the
  ; destination cost from DESTS.DB.

  If eventInfo.newValue() <> "" then

     ; If the new value is in DESTS, then assign that record's Travel
     ; Cost (a field) to destCost.

     If allDests.locate("Destination Name", eventInfo.newValue()) then
           destCost = allDests."Travel Cost"
     else

           ; We couldn't find the new destination, so blank destCost,
           ; display an error message, and then prevent other actions
           ; from taking place.

           msgInfo("Invalid Destination", "Press <Ctrl+" +
                   "Space> to display a list of possible " +
                   "destinations or choose Record | Cancel " +
                   "Changes to restore the former value.")
           eventInfo.setErrorCode(cannotDepart)
     endIf
  endIf

endMethod
```

newValue

mastOrd.#Page2.Destination.newValue

```
method newValue(var eventInfo Event)

; This method updates the Cost per Person per Week field (destCost)
; and triggers a recalc of the Vacation Cost field(s).

; We need to recalculate destCost when the app starts and when a
; new record appears in the diveOrds frame.

  If self.isBlank() then
     destCost.blank()
  else

     If allDests."Destination Name" <> self.value then
           allDests.locate("Destination Name", self.value)
           destCost.value = allDests."Travel Cost"
     endIf
  endIf

endMethod
```

MASTORD.#Page2.Return_Date

Methods

keyPhysical

changeValue

keyPhysical

mastOrd.#Page2.Return_Date.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method overrides the default behavior of the
; <Right Arrow> key for this field.  We want this key to
; select the Customer Name field, instead of displaying
; the next vacation record.

  If eventInfo.vCharCode() = VK_RIGHT then
     disableDefault
     nameList.moveTo()
  endIf

endMethod
```

changeValue

mastOrd.#Page2.Return_Date.changeValue

```
method changeValue(var eventInfo ValueEvent)

  ; Make sure the Return Date is later than the Depart Date.  If not,
  ; display an error message and prevent the user from leaving the field.

  ; Because the user may be using Field View to change this field,
  ; we need to cast the newValue as a Date value.

  If not Depart_Date.isBlank() then
     If Date(eventInfo.newValue()) <= Depart_Date.value then
           beep()
           msgInfo("Invalid Return Date: " + eventInfo.newValue(), "Enter a
  return date that is " +
                   "later than the Depart Date (" +
  String(Depart_Date.value) +
                   ")")
           eventInfo.setErrorCode(-1)
     else
           doDefault
           weeksOut.Action(dataRecalc)
           if equipFrm.nRecords > 0 then
                 setPrice()
           endIf
     endIf
  endIf
endMethod
```

MASTORD.#Page2.Depart_Date

Methods

keyPhysical

changeValue

keyPhysical

mastOrd.#Page2.Depart_Date.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method lets the user press <Space> to enter the
; current date as the default depart date.

  If eventInfo.vCharCode() = VK_SPACE and not self.touched then
     self.value = today()
     self.postAction(fieldForward)
     disableDefault
  endIf


endMethod
```

changeValue

mastOrd.#Page2.Depart_Date.changeValue

```
method changeValue(var eventInfo ValueEvent)

  ; Make sure the Depart Date is earlier than the Return Date.  If not,
  ; display an error message and prevent the user from leaving the field.

  ; Because the user may be using Field View to change this field,
  ; we need to cast the newValue as a Date value.

  If not Return_Date.isBlank() then
     If Date(eventInfo.newValue()) >= Return_Date.value then
           beep()
           msgInfo("Invalid Depart Date: " + eventInfo.newValue(), "Enter a
  return date that is " +
                   "earlier than the Return Date (" +
  String(Return_Date.value) +
                   ")")
           eventInfo.setErrorCode(-1)
     else
           doDefault
           weeksOut.Action(dataRecalc)
           if equipFrm.nRecords > 0 then
                 setPrice()
           endIf
     endIf
  endIf

endMethod
```

pushButton

mastOrd.#Page2.help_button.pushButton

```
method pushButton(var eventInfo Event)

  doDefault
  helpShowContext(helpFile, headerHelp)

endMethod
```

pushButton

mastOrd.#Page2.exit_button.pushButton

```
method pushButton(var eventInfo Event)

  doDefault
  okToExit = True
  formReturn(0)

endMethod
```

pushButton

mastOrd.#Page2.printBtn.pushButton

```
method pushButton(var eventInfo Event)

; This method handles this application's printing abilities.  It presents
; a dialog box to the user, lets them choose the report they want to print,
; then performs the actions required to print the desired report.  In some
; cases, this requires a query.

var

  Response AnyType                        ; Holds user's response to the
  PRINTRPT dialog
  rpt_Form Form                           ; Form variable for RPTSLIST.FSL
  rpt_Info reportPrintInfo                ; Report Information for the final
  report
  rptQuery Query                          ; Holds query for current
  invoice/customer reports
  rpt_Name Report                   ; Generates the final report


endVar

  ; First, let the button complete its "push" actions

  doDefault

  ; Turn the pointer to an hourglass, then open the form containing
  ; the report list, and let the user interact with it.

  setMouseShape(mouseWait)
  rpt_Form.open("RPTSLIST")
  rpt_Form.rptType.value = "quikInv"
  setMouseShape(mouseArrow)
  Response = String(rpt_Form.wait())
  setMouseShape(mouseWait)
  rpt_Form.close()
  rpt_Info.startPage = 1
  rpt_Info.endPage = 9999

  ; User has closed the RPTSLIST form, so we need to figure out what
  ; to do next.  The values in Response come from the RPTSTYPE field
  ; object on RPTSLIST.FSL.  The values returned by RPTSTYPE are
  ; different than the text labels seen by the user.  This minimizes
  ; code, but also makes maintaining the forms a little more difficult.

  Switch

     ; The user canceled the dialog box by choosing the Cancel
     ; button or by double-clicking the form's Control menu.  In
     ; either case, we don't need to do anything, so leave this method.

     case Response = "Cancel" : Return
```

```
   ; User chose a quick print of the current invoice.  This report
   ; is essentially a "screen dump" and we don't need to run the
   ; rest of the code after printing the screen.

   case Response = "quikInv" :

         menuAction(MenuFilePrint)
         Return

   ; Now, we need to do some work.  The user chose to print a full
   ; copy of the current invoice.

   case Response = "currInv" :

   ; First, define a query (using a query variable) that locates the
   ; current invoice and creates an ANSWER table (in the user's private
   ; directory) containing that record.  Notice the query expression in
   ; the Order No field.

         rptQuery = Query

            DIVEORDS.DB | Customer No | Order No                | Sale
Date |
                        | Check       | Check ~(Order_No.value) | Check
|

            DIVEORDS.DB | Ship Via | Ship Cost | Subtotal | Total Invoice
|
                        | Check    | Check     | Check    | Check
|

            DIVEORDS.DB | PaymentMethod | CcNumber | CcExpDate | No Of
People |
                        | Check         | Check    | Check     | Check
|

            DIVEORDS.DB | Depart Date | Return Date | Destination |
VacationCost |
                        | Check       | Check       | Check       | Check
|

         EndQuery

         ; Now, try to run the query.  If it succeeds, then we'll
continue
         ; setting up the report.  Otherwise, display a generic error
         ; message.

         If rptQuery.executeQBE() then

               ; Now, indicate the report we want to run and the master
               ; table the report will use.

               rpt_Info.Name = "PRINTINV"
```

```
            rpt_Info.masterTable = ":PRIV:ANSWER"
         else
            beep()
            msgInfo("Can't Perform Report Query", ErrorMessage())
            return
         endIf

   ; User chose to print a summary of orders for the current customer.
   ; So, we need a query to extract the customer information (and put
   ; it in :PRIV:ANSWER.DB).  We'll try to run that query.  If it
   ; succeeds, set up the final report; otherwise, display a generic
   ; error message and then quit the method.


   case Response = "sumCust" :

         ; Now, define a query (using a query variable) that locates the
         ; current customer and creates an ANSWER table (in the user's
private
         ; directory) containing that record.  Notice the query
expression in
         ; the Customer No field.

         rptQuery = Query

            DIVECUST.DB | Customer No                | Name   | Street |
                        | Check ~(Customer_No.value) | Check  | Check  |

            DIVECUST.DB | City   | State/Prov | Zip/Postal Code |
                        | Check  | Check      | Check           |

            DIVECUST.DB | Country | Phone  | First Contact |
                        | Check   | Check  | Check         |

         endQuery

         ; Now, try to run the query.  If it succeeds, then we'll
continue
         ; setting up the report.  Otherwise, display a generic error
         ; message.

         If rptQuery.executeQBE() then

               ; Now, indicate the report we want to run and the master
               ; table the report will use.

               rpt_Info.Name = "CUSTORDS"
               rpt_Info.masterTable = ":PRIV:ANSWER"
         else
               beep()
               msgInfo("Can't Perform Report Query", ErrorMessage())
               return
         endIf

   ; User chose to print a summary of all the orders in the system.
```

```
      ; Since the report is designed to do this, we don't need a query.
      ; So, we'll simply set up the name of the report.

      case Response = "sumOrds" :

            rpt_Info.Name = "CUSTORDS"

   endSwitch

   ; At this point, we're ready to run the actual report.  So, try to
   ; print it.  If it fails, then display a generic error message.

   If not rpt_Name.print(rpt_Info) then
      msgInfo("Can't Print Report", ErrorMessage())
   endIf

   setMouseShape(mouseArrow)

endMethod
```

MASTORD.#Page2.nameList

Methods

open

keyPhysical

action

newValue

open

mastOrd.#Page2.nameList.open

```
method open(var eventInfo Event)

  ; This method fills the drop down list with all the
  ; customer names in DIVECUST.

  doDefault

  ; The :WORK: alias tells Paradox to open the DIVECUST
  ; table from disk.  Otherwise, the list would only
  ; contain the name of the customer currently attached
  ; to this order.

  self.theList.dataSource = "[\":WORK:DIVECUST.DB\".Name]"

  ; Now, set this field to the customer attached to the
  ; current order.

  self.value = Name

endMethod
```

keyPhysical

mastOrd.#Page2.nameList.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method overrides the default behavior of certain
; keystrokes and selects a specific field object, depending
; on the keystroke pressed.

var

  keyPress smallInt ; Holds the key pressed by the user

endVar

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  keyPress = eventInfo.vCharCode()
  disableDefault
  switch

     case keyPress = VK_LEFT or
          keyPress = VK_UP or
          (keyPress = VK_TAB and
          eventInfo.isShiftKeyDown()) : No_Of_People.moveTo()

     case keyPress = VK_DOWN or
          keyPress = VK_RIGHT or
          keyPress = VK_TAB : Depart_Date.moveTo()

     otherwise: enableDefault

  endSwitch
  delayScreenUpdates(No)
  setMouseShape(mouseArrow)


endMethod
```

action

mastOrd.#Page2.nameList.action

```
method action(var eventInfo ActionEvent)

  If eventInfo.id() = dataDeleteRecord then
     beep()
     msgInfo("Can't Delete Customer Info Here", "Use the " +
             "Change Address or Add Customer dialog boxes " +
             "to delete customers from the system.")
     eventInfo.setErrorCode(-1)
  endIf

endMethod
```

newValue

mastOrd.#Page2.nameList.newValue

```
method newValue(var eventInfo Event)

  ; This method performs the work of putting the user's
  ; customer selection (the new name) into the linking
  ; field (Customer No) in DIVEORDS.

  ; The idea is to let the user work with things they
  ; are familiar with (for example, a customer's name)
  ; and let Paradox and the application do the work of
  ; maintaining the link.

  ; First, see why this method was called.  We want to
  ; change the customer number field only if the user
  ; changes the value of this field.

  if eventInfo.reason() = editValue then

     ; If the user chose (or entered) the same name,
     ; we don't need to do anything.

     If self.value <> Name.value then

           ; The user did enter a new name, so we need
           ; to make sure the customer is in DIVECUST.
           ; Because this field isn't bound to a table,
           ; we use a tCursor to locate the new name.

           If cust_TC.locate("Name", self.value) then

                 ; The new customer is in DIVECUST, so let's
                 ; try to update the Customer No.

                 try
                       Customer_No = cust_TC."Customer No"
                 onFail

                 ; We can't change the Customer No, so let's
                 ; reset the value of this field so it matches
                 ; the customer already attached to this order,
                 ; then, we'll alert the user and display an
                 ; appropriate message.  The disableDefault
                 ; keeps focus on this field object.

                       self.Value = Name.Value
                       beep()
                       msgInfo("Oops!", "Couldn't change the customer." +
                               "  Make sure you're editing the table(s)" +
                               " and that you chose an existing " +
                               " customer.")
                       disableDefault
                 endTry
```

```
            else

                  ; We couldn't locate the new name in DIVECUST,
                  ; so, we'll reset the name, alter the user,
                  ; and prevent focus from moving from this field.

                        self.value = Name.value
                        beep()
                        msgInfo("Oops!", "Couldn't change the customer." +
                                "  Make sure you're editing the table(s)" +
                                " and that you chose an existing " +
                                " customer.")
                        disableDefault
            endIf
      endIf
  endIf

endMethod
```

pushButton

mastOrd.#Page2.editCust.pushButton

```
method pushButton(var eventInfo Event)

; The method opens the DIVECUST form, locates the
; current customer, and waits for the user to close the
; form.  When that happens, we update the customer list
; and assign, if appropriate, the new customer to the
; current order.

var

  custForm Form     ; Variable to DIVECUST form
  oldID_No Number   ; Holds the current customer number
  newID_No,         ; Holds the return value from custForm
  Response String   ; Holds user's response to a dialog box
  uiTarget uiObject ; Pointer to the names shown in custForm


endVar

  doDefault
  delayScreenUpdates(Yes)

  oldID_No = Customer_No

  ; Open DIVECUST.FSL, change its title, and then
  ; add a record.

  custForm.openAsDialog("DIVECUST", winStyleHidden + winStyleDefault)
  custForm.setTitle("Edit Customer")
  custForm.Customer.locate("Customer No", Customer_No)
  delayScreenUpdates(No)

  ; Now, let the user interact with the form.

  custForm.show()
  custForm.modal = True
  newID_No = String(custForm.wait())

  ; Now, if the user didn't choose Cancel from the form,
  ; then we need to update the order (with the new
  ; customer information) and the drop-down list.  (The
  ; drop down field will be updated by the dataSource command.)

  nameList.theList.dataSource = "[\":WORK:DIVECUST.DB\".Name]"
   uiTarget.attach(custForm.Customer.Name)
  cust_TC.attach(uiTarget)

  If newID_No <> "Cancel" then

     If Number(newID_No) <> oldID_No then
           Response = msgQuestion("New Customer ID", "You chose a
```

```
  different" +
                    "customer (" + custForm.Customer.Name.Value + ").  " +
                    "Do you want to change the customer for this order?")

          If Response = "Yes" then
                Customer_No = Number(newID_No)
          endIf
     endIf

     If nameList.Value <> Name.Value then
          nameList.Value = Name.Value
     endIf

  endIf


  ; We're done with the form, so let's close it.

  custForm.close()

endMethod
```

newValue

mastOrd.#Page2.name.newValue

```
method newValue(var eventInfo Event)

  if eventInfo.reason() <> startUpValue then
     nameList = self.value
  endIf

endMethod
```

pushButton

mastOrd.#Page2.add_Cust.pushButton

```
method pushButton(var eventInfo Event)

; The method opens the DIVECUST form, inserts the record,
; and waits for the user to close the form.  When that
; happens, we update the customer list and assign, if
; appropriate, the new customer to the current order.

var

  custForm Form   ; Variable to DIVECUST form
  Response String ; Holds the return value from custForm

endVar

  doDefault
  delayScreenUpdates(Yes)

  ; First, open DIVECUST.FSL, change its title, and then
  ; add a record.

  custForm.openAsDialog("DIVECUST", winStyleHidden + winStyleDefault)
  custForm.setTitle("Add Customer")
  custForm.Customer.action(dataInsertRecord)
  delayScreenUpdates(No)

  ; Now, let the user interact with the form.

  custform.show()
  custForm.Modal = True

  Response = String(custForm.wait())

  ; The user exited the form, so let's close it.

  custForm.close()

  ; Now, if the user didn't choose Cancel from the form,
  ; then we need to update the order (with the new
  ; customer information) and the drop-down list.  (The
  ; drop-down field is updated by changing the DataSource property.)

  If Response <> "Cancel" then
     nameList.theList.dataSource = "[\":WORK:DIVECUST.DB\".Name]"
     Customer_No = Number(Response)
  endIf


endMethod
```

MASTORD.#Page2.custAddr

There are no methods attached to custAddr.

Objects contained by custAddr

Zip_Postal_Code

City

calcField

mastOrd.#Page2.Zip_Postal_Code.calcField

```
[DIVECUST.Zip/Postal Code] + IIF([DIVECUST.Zip/Postal Code] = "", "", "  ")
  + [DIVECUST.Country]
```

calcField

mastOrd.#Page2.City.calcField

```
[DIVECUST.City] + iif([DIVECUST.City] = "", "", ", ") +
  [DIVECUST.State/Prov]
```

calcField

mastOrd.#Page2.eq_Total.calcField

```
sum([DIVEITEM.Line Total]) + Ship_Cost
```

calcField

mastOrd.#Page2.invTotal.calcField

```
vacTotal + eq_Total
```

MASTORD.#Page2.vacTotal

Methods
newValue

Objects contained by vacTotal
calcField

calcField

mastOrd.#Page2.vacTotal.calcField

```
No_of_People * weeksOut * destCost
```

newValue

mastOrd.#Page2.vacTotal.newValue

```
method newValue(var eventInfo Event)

  invTotal.postAction(dataRecalc)

endMethod
```

MASTORD.#Page2.equipFrm

Methods
action

Objects contained by equipFrm
Record63

action

mastOrd.#Page2.equipFrm.action

```
method action(var eventInfo ActionEvent)

  If eventInfo.id() = dataRecalc then
     doDefault
     Price.action(1)
  endIf

endMethod
```

MASTORD.#Page2.#Record63

Methods

arrive

action

depart

Objects contained by #Record 63

itemNote

Price

Qty

Rental_Sale

Item_No

arrive

mastOrd.#Page2.#Record63.arrive

```
method arrive(var eventInfo MoveEvent)

  itemNote.Frame.Style = outside3DFrame

endMethod
```

depart

mastOrd.#Page2.#Record63.depart

```
method depart(var eventInfo MoveEvent)

  itemNote.Frame.Style = solidFrame

endMethod
```

action

mastOrd.#Page2.#Record63.action

```
method action(var eventInfo ActionEvent)

; This method contains the code that asks the user to confirm
; a record deletion and the code that handles record "flyaway"
; when an equipment item is added or changed.

var

  actionID smallInt ; Holds the id of the user's action.  This
                    ; improves performance because we don't
                    ; call eventInfo.id() for each branch in
                    ; the SWITCH statement.
  Response String   ; Holds the user's response to a dialog box.

endVar

  ; Turn the pointer to an hourglass and delay updates to
  ; the screen.  This lets the user know something is going
  ; on and slightly improves performance of interim operations.

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  ; Get the id of the action the user generated.

  actionID = eventInfo.id()

  ; Evaluate what the user did.  If it's something we're
  ; concerned about, execute our custom code; otherwise,
  ; perform the default actions.

  switch

     ; The user chose to delete a record, so let's pop up
     ; a dialog making sure they really want to delete it.

     case actionID = dataDeleteRecord :

           Response = msgQuestion("Are you sure?", "If you " +
                      "choose Yes, this equipment line will " +
                      "be permanently deleted.  Choose No " +
                  "to Cancel.")

           ; If the user chose not to delete the record, we
           ; want to prevent anything else from happening.
           ; Otherwise, we'll set a flag which disables the
           ; validity checking in the canDepart method.

           If Response = "No" then
                 eventInfo.setErrorCode(-1)
           endIf
```

```
    ; If the user chose to unlock a record, we want to
    ; follow it if it flies away.  The easiest way to
    ; do this, in this form, is to force a dataPostRecord,
    ; which is similar to the Record | Post/Keep Locked
    ; command.

    case actionID = dataUnlockRecord or
         actionID = dataToggleLockRecord:

          ; Because this could be caused by the user locking
          ; the record (using Record | Lock), we want
          ; to post the record only if it's already locked.

          If self.locked and self.touched then
                If not action(dataPostRecord) then
                      eventInfo.setErrorCode(-1)
                endIf
          endIf

    case eventInfo.id() = dataPostRecord :

          if self.touched then
                doDefault
                if errorCode() <> 0 then
                      msgInfo("Can't Save Current Equipment Item",
                              "Reason: " + errorMessage())
                      eventInfo.setErrorCode(-1)
                endIf
          endIf

  otherwise : doDefault

  endSwitch

  delayScreenUpdates(No)
  setMouseShape(mouseArrow)

endMethod
```

MASTORD.#Page2.itemNote

Methods

open

arrive

depart

mouseDown

mouseUp

calcField

mouseClick

Objects contained by itemNote

Line_Total

open

mastOrd.#Page2.itemNote.open

```
method open(var eventInfo Event)
  self.Frame.Style = solidFrame
endMethod
```

arrive

mastOrd.#Page2.itemNote.arrive

```
method arrive(var eventInfo MoveEvent)

  self.Frame.Style = SolidFrame
  self.Frame.Color = Black

endMethod
```

depart

mastOrd.#Page2.itemNote.depart

```
method depart(var eventInfo MoveEvent)

  self.Frame.Style = SolidFrame
  self.Frame.Color = Gray

endMethod
```

mouseDown

mastOrd.#Page2.itemNote.mouseDown

```
method mouseDown(var eventInfo MouseEvent)

  if self.Frame.Style = outside3dFrame then
     self.Frame.Style = inside3DFrame
  endIf

endMethod
```

MouseUp

mastOrd.#Page2.itemNote.mouseUp

```
method mouseUp(var eventInfo MouseEvent)

  If self.Frame.Style = inside3DFrame then
     self.Frame.Style = outside3DFrame
  endIf

endMethod
```

calcField

mastOrd.#Page2.itemNote.calcField

```
iif([DIVEITEM.Line Note] = "", "", "¶")
```

mouseClick

mastOrd.#Page2.itemNote.mouseClick

```
method mouseClick(var eventInfo MouseEvent)

; This method lets the user, when appropriate, add or edit
; a line note for the current piece of equipment.  The note
; itself can be up to 255 characters.

var
  tempMemo String   ; Holds the contents of the memo before
                    ; they're posted to the table
  uiTarget uiObject ; Holds the object that got the mouse
                    ; click; used to make sure the user
                    ; clicked the note "button" for the
                    ; current record in the tableFrame.
endVar

  ; Get the object that got clicked.

  eventInfo.getTarget(uiTarget)

  ; Perform the mouseUp method before getting into the
  ; rest of this.

  doDefault

  ; Verify the user clicked the note "button" for the
  ; current table frame record.  This is done by verifying
  ; the current frame style and comparing the name of
  ; the target object (the one that got clicked) with the
  ; name of the active object (the one running this method).

  if uiTarget.Frame.Style = Outside3DFrame and
     (uiTarget.fullName = self.fullName) then

     ; The user clicked the right memo "button," so
     ; initialize the temporary variable to the current
     ; line note, if any.  (Line_Note is a hidden field
     ; in the table frame.)

     tempMemo = Line_Note.value

     ; Let the user work with the memo variable

     tempMemo.view("Enter note for this item")

     ; If the variable was changed, then we need to make sure
     ; it can be posted and react accordingly.

     if tempMemo <> Line_Note then

           switch
```

```
                ; User either emptied the current memo or didn't
                ; add one.  In either case, clear the contents of
                ; the field.

                case tempMemo.size() = 0 :
                      Line_Note = ""

                ; User added too many characters.  So, display a
                ; message, then save as many characters as we can.

                case tempMemo.size() > 255 :
                      Message("Note is too long.  Truncating it " +
                              "to 255 characters...")
                      Line_Note = tempMemo.subStr(1, 255)


                ; User's memo is just fine, so save it as is.

                otherwise : Line_Note = tempMemo
          endSwitch
    endIf

    ; Now, recalculate the memo indicator.

    self.action(dataRecalc)
  else

    ; User can't edit the memo they clicked, so beep at
    ; them.

    beep()
  endIf

endMethod
```

newValue

mastOrd.#Page2.Line_Total.newValue

```
method newValue(var eventInfo Event)


  If eventInfo.reason() <> editValue and self.locked then
     doDefault
     eq_Total.postAction(dataRecalc)
  endIf

endMethod
```

MASTORD.#Page2.Price

Methods

action

setPrice

action

mastOrd.#Page2.Price.action

```
method action(var eventInfo ActionEvent)

  if eventInfo.id() = 1 then
     self.setPrice()
     eq_Total.postAction(dataRecalc)
  endIf

endMethod
```

setPrice

mastOrd.#Page2.Price.setPrice

```
method setPrice()

; This method sets the price for the ordered item.  It
; needs three fields to have values before it will work
; (Item_No, Rental_Sale, and Qty).

  ; If any of the three fields is empty, then
  ; set the field to zero.

  If Item_No.isBlank() or Rental_Sale.isBlank() or
      Qty.isBlank() then

     self.Blank()

  else

     ; If the customer wants to buy the equipment, then
     ; base the price on the Sale_Price

     If Rental_Sale = "Sale" then
           self.value = Sale_Price
     else

           ; Otherwise, set the price to equal 15% of the
           ; sale price * the number of weeks out.

           self.value = Sale_Price * rentRate * weeksOut

     endIf

  endIf

  ; Now, update the Line Total.

  Line_Total = self.value * Qty.Value

endMethod
```

MASTORD.#Page2.Qty

Methods

keyPhysical

changeValue

keyPhysical

mastOrd.#Page2.Qty.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method overrides the normal action of the UP key on this
; field.

var
  keyPress smallInt ; Holds the key the user pressed
endVar

  ; We want this action to happen only when the user presses <Up Arrow> on
   ; the first row of the table frame; otherwise, let the default behavior
  ; take place.

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  if self.rowNo = 1 then
     If eventInfo.vCharCode() = VK_UP then
                 doDefault
                 nameList.moveTo()
     endIf
  endIf

  delayScreenUpdates(No)
  setMouseShape(mouseArrow)

endMethod
```

changeValue

mastOrd.#Page2.Qty.changeValue

```
method changeValue(var eventInfo ValueEvent)

; This method verifies the quantity ordered for an item.
; It makes sure there are enough units in stock to handle
; the user's request and displays messages if the user
; orders too many items or if the number of units on Hand
; falls below the reorder point.


var

  orderAmt smallInt ; Holds the quantity entered by the user

endVar

  orderAmt = eventInfo.newValue()

  if orderAmt > On_Hand then
     msgInfo("Can't Fill Order", "There are only " +
             String(On_Hand) + " units in stock.  Enter a " +
             "smaller quantity.")
     eventInfo.setErrorCode(-1)
  else
     doDefault
     Price.setPrice()
     if On_Hand - orderAmt <= reOrder_Point then
           msgInfo("Time to Reorder Item No " + String(Item_No),
                   "Only " + String(smallInt(On_Hand - orderAmt)) +
                   " units left in stock.")
     endIf
  endIf

endMethod
```

MASTORD.#Page2.Rental_Sale

Methods

arrive

depart

keyPhysical

newValue

arrive

mastOrd.#Page2.Rental_Sale.arrive

```
method arrive(var eventInfo MoveEvent)

  self.saleList.Visible = True

endMethod
```

depart

mastOrd.#Page2.Rental_Sale.depart

```
method depart(var eventInfo MoveEvent)

  self.saleList.Visible = False

endMethod
```

keyPhysical

mastOrd.#Page2.Rental_Sale.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

var

  keyPress smallInt ; Holds the key pressed by the user

endVar

  keyPress = eventInfo.vCharCode()
  disableDefault
  switch

     case keyPress = VK_LEFT  : self.action(moveLeft)
     case keyPress = VK_RIGHT : self.action(moveRight)
     case keyPress = VK_UP    :
           if self.rowNo = 1 then
           enableDefault
           doDefault
           moveto(diveOrds.return_Date)
           else
                 self.action(dataPriorRecord)
           endIf
     case keyPress = VK_DOWN  : self.action(dataNextRecord)
     otherwise : enableDefault

  endSwitch

endMethod
```

newValue

mastOrd.#Page2.Rental_Sale.newValue

```
method newValue(var eventInfo Event)

  if eventInfo.reason() = editValue then
     doDefault
     Price.setPrice()
  endIf

endMethod
```

MASTORD.#Page2.Item_No

Methods

keyPhysical

action

changeValue

keyPhysical

mastOrd.#Page2.Item_No.keyPhysical

```
method keyPhysical(var eventInfo KeyEvent)

; This method overrides the normal action of dataPriorField
; for the tableFrame; it selects the Customer Name field.

var
  keyPress smallInt ; Holds the key the user pressed
endVar

  ; We want this action to happen only when the user
  ; presses <Up Arrow> or <Shift+Tab> on the first row
  ; of the table frame; otherwise, let the default behavior
  ; take place.

  setMouseShape(mouseWait)
  delayScreenUpdates(Yes)

  if self.rowNo = 1 then

     switch

           case (eventInfo.vCharCode() = VK_LEFT or
             (eventInfo.vCharCode() = VK_TAB and
             eventInfo.isShiftKeyDown())) :

                 doDefault
                 nameList.moveTo()

           case eventInfo.vCharCode() = VK_UP :

                 doDefault
                 diveOrds.Ship_Via.moveTo()

           otherwise : doDefault
     endSwitch
  endIf

  delayScreenUpdates(No)
  setMouseShape(mouseArrow)

endMethod
```

action

mastOrd.#Page2.Item_No.action

```
method action(var eventInfo ActionEvent)

  If eventInfo.id() = dataLookup then
     doDefault
     Price.Action(1)
  endIf

endMethod
```

changeValue

mastOrd.#Page2.Item_No.changeValue

```
method changeValue(var eventInfo ValueEvent)

  doDefault
  Price.Action(1)

endMethod
```

MASTLIB.LSL Library

Methods

clearPageValues

createObjMenu

formCanClose

getDataSource

getLookUpTitle

isCanClose

isVCROpen

listPageObjects

MASTObjMsg

open

pageNameOf

returnisCalled

setDataSource

setIsCalledTrue

setIsCalledFalse

setLookUpTitle

setPageValues

VCRisOpen

vcrName

VCRisClosed

Var and Const windows

Var window

Const window

clearPageValues

MASTLIB.clearPageValues
Clears any previous Query Value for the page (given by the *pageName* argument) of the form given by the *formName* argument. The two arguments map to a Query ID through QBEMAP.DB. The Query ID then locates the entry in QBEVAL.DB. Called prior to setPageValues.

See Also
MASTLIB.setPageValues

```
method clearPageValues(formName String, pageName String)

var
   myQuery Query
endVar

; clear Qbeval.db values for this page
myQuery=Query

qbemap.db | Form Name    | Page ID     | Query ID |
          | ~formName    | ~pageName   | _qID     |

qbeval.db | Query ID | Query Value     |
          | _qID     | changeto blank  |

EndQuery

executeQBE(myQuery)

endMethod
```

createObjMenu

MASTLIB.createObjMenu

Creates and displays an object-sensitive pop-up menu. Called with formName, pageName, and UIObject arguments passed by the formName.mouseRightUp method. The target object is determined by the focus (mouse cursor position, for example) but some objects (such as texts or bitmaps) are excluded by mouseRightUp for particular forms. An object menu is only created and displayed if an entry for the target object is located in MSGHELP.DB. The standard menu option is Code Help. Other options depend on the target object. For example, if Graphic has the focus, the About Fish option is added. createObjMenu also processes the menu choices.

See Also

DIVEWRCK.mouseRightUp,

DIVEBIO.mouseRightUp,

DIVEPLAN.mouseRightUp,

MASTORD.mouseRightUp

DIVESITE.mouseRightUp

```
method createObjMenu(var formName Form, pageName String, uio UIObject)

; This is generalized object menu system called by mouseRightDown methods.
; It provides object help by locating the object's name in MsgHelp.db
; and then offers the user a PopUpMenu with "Object Help" as the standard
; option. If the user selects "Object Help" from the menu, an info box
; is displayed containing the object's help text contained in Msghelp.

; STEPS:
; First, determine if MsgHelp is successfully opened and, if it is,
; locate the object in the table. Not all objects have help. If the
; object is found in MsgHelp, display a standard object help pop- up menu.
; Next, add to the PopUpMenu any additional items that only pertain to
; the object. Finally, process the menu choice that the user selects.

var
  p PopUpMenu
endvar

; opened object help table already --
; now locate matching formName.pageName.object
if objHelpTc.Locate("Form Name", formName.Name, "Page Name", pageName,
  "Object ID", uio.name)
  then
  ; build menu
  p.empty() ; First, empty old items
  p.addStaticText("Object Properties")
  p.addSeparator()
  p.addText("Code Help")

  ; add menu items to standard "About Object"
  switch
   case uio.name = "processButton" : p.addText("View Query")
   case uio.name = "diveFlagBox"   : p.addText("About MAST")
   case uio.name = "Graphic"       : p.addText("&About Fish")
```

```
   case uio.name = "shipWrecks"    : p.addText("&Choose Wreck")
  endswitch

  choice=p.Show()
  ; process menu choices
  switch
   case choice = "Code Help"  :
     helpShowContext(proghelp, objHelptc."Context ID")

   case choice = "View Query" :
     qRpt.open("Qval.rsl")
     qRpt.setTitle("View Query")

   case choice = "About MAST" :
     setIsCalledTrue()
     if aboutFm.open("Abtmast.fsl") then ; modal dialog opened
           aboutFm.Wait()
           aboutFm.Close() ; now close it
           setIscalledFalse()
     endIf

   case choice = "&About Fish" :
     formName.marineLifePage.notes.moveTo()

   case choice = "&Choose Wreck" :
     formName.wreckFld = self."Ship Name".value
     formName.wreckFld.moveto()

  endswitch
endIf

; that's it folks!

endMethod
```

formCanClose

MASTLIB.formCanCLose

Sets a flag to indicate that a form can be closed.

```
method formCanClose()
  canClose = True
endMethod
```

getDataSource

MASTLIB.getDataSource
Returns the item stored at the ListSource index of the *TitleAndSource* DynArray declared in MASTLIB.LSL.

See Also
MASTLIB.setDataSource

```
method getDataSource() String
  return TitleAndSource["ListSource"] ; return value stored in DynArray
endMethod
```

getLookUpTitle

MASTLIB.getLookUpTitle
Returns the item stored at the LookUpTitle index of *TitleAndSource* a DynArray declared in MASTLIB.LSL.

See Also
MASTLIB.setLookUpTitle

```
method getLookUpTitle() String
  return(TitleAndSource["LookUpTitle"])
endMethod
```

isCanClose

MASTLIB.isCanClose

Checks the value of the variable *canClose* to report whether a form can close.

```
method isCanClose() Logical

  ; is Diveplan asking for a close
  if isAssigned(canClose)
    then
     return canClose
  ; Diveplan hasn't asked for close -- cannot close form
  else
     return False
  endIf

endMethod
```

isVCROpen

MASTLIB.isVCROpen

Reports whether a custom SpeedBar is open.

```
method isVCROpen() Logical

  ; is VCR open?
  if isAssigned(VCROpenLg)
  ; Videobar has been opened
    then
     return VCROpenLg
  ; Videobar has not been opened
  else
     return False
  endIf

endMethod
```

listPageObjects

MASTLIB.listPageObjects

Gets the values of objects on the current page and writes them to a table.

```
method listPageObjects(formName String, pageName String) Logical

; This method runs a query (pageObjsQBE) to find all value setting
; objects on the current form & page, sending the results to Objlist.db.

var
  pageObjsQBE Query

endVar

pageObjsQBE = Query

            qbeMap.db | Form Name   | Page ID     | Object Path |
                      |  ~formName |  ~pageName | Check       |

            endQuery

executeQBE(pageObjsQBE, "Objlist.db")

RETURN TRUE

endMethod
```

MASTObjMsg

MASTLIB.MASTObjMsg

Opens MSGHELP.DB and tries to locate an entry matching the given *formName*, *pageName*, and *ObjID* arguments. If found, the Message Text value is displayed in a message.

```
method MASTObjMsg(formName String, pageName String, ObjID String)

;locate matching formName.pageName.object
If myMsgTc.locate("Form Name", formName, "Page Name", pageName, "Object
  ID", ObjID)
  then message(myMsgTc."Message Text".value)
endIf

endMethod
```

open

MASTLIB.open

Opens MSGHELP.DB, a table that stores help messages.

```
method open(var eventInfo Event)
if not objHelpTc.Open("Msghelp.db") then
  ; failed to open object help table -- inform user
    msgStop("Problem", "Couldn't open MsgHelp.db")
endIf

if not myMsgTC.open("Msghelp.db") then
    msgStop("Problem", "Couldn't open MsgHelp.db")
endIf

endMethod
```

pageNameOf

MASTLIB.pageNameOf

Returns the name of the page that contains the UIObject specified in the argument *ui*. A null string is returned if no containing page is found.

```
method pageNameOf(ui UIObject) String

; this method returns page name for the current object
var s string endvar

while ui.Class <> "Page" and ui.ContainerName <> ""
   ui.attach(ui.ContainerName)
endWhile

if ui.class = "Page" then s = ui.name
   else s = ""
endIf

return s

endMethod
```

returnisCalled

MASTLIB.returnisCalled

Returns the value (True or False) assigned to the MASTLIB variable *isCalled.* Returns False if no value has been assigned. *isCalled* is tested by the open methods of DIVESITE, DIVEWRCK, DIVEBIO, and ABTMAST to ensure that these forms are opened only by DIVEPLAN.

```
method returnIsCalled() Logical
  if not isCalled.isAssigned()
     then
           isCalled = False
  endIf
  return isCalled
endMethod
```

setDataSource

MASTLIB.setDataSource

Assigns the *lookSrc* string argument to the ListSource index of *TitleAndSource*, a DynArray declared in MASTLIB.LSL

See Also

MASTLIB.getDataSource

```
method setDataSource(const lookSrc String)
  TitleAndSource["ListSource"] = lookSrc
endMethod
```

setIsCalledTrue

MASTLIB.setIsCalledTrue

Sets the MASTLIB variable *isCalled* to True. *isCalled* is used to prevent the opening of certain forms directly rather than by DIVEPLAN.

See Also

MASTLIB.returnisCalled

```
method setIsCalledTrue()
  isCalled = True
endMethod
```

setIsCalledFalse

MASTLIB.setIsCalledFalse

Sets the MASTLIB variable *isCalled* to False. *isCalled* is used to prevent the opening of certain forms directly rather than by DIVEPLAN.

See Also

MASTLIB.returnisCalled

```
method setIsCalledFalse()
  isCalled = False
endMethod
```

setLookUpTitle

MASTLIB.setLookUpTitle

Assigns the *lookTtl* string argument to the LookUpTitle index in *TitleAndSource* DynArray (declared in MASTLIB.LSL).

See Also

MASTLIB.getLookUpTitle

```
method setLookUpTitle(const lookTtl String)
  TitleAndSource["LookUpTitle"] = lookTtl ; set value of DynArray item
endMethod
```

setPageValues

MASTLIB.setPageValues
Runs a query to find all value objects (vacation criteria) selected on the given *formName* and *pageName* arguments. Called by the acceptBttn methods of *wrecksPage*, *destsPage*, and *marineLifePage* to collect, in QBEVAL.DB, the selected criteria for possible destinations. The query results are sent to the TCursor *newValsTc*. QBEVAL.DB is searched by DIVEPLAN.destqbe to find matching destinations in DEST.DB.

See Also
MASTLIB.clearPageValues,

DIVEPLAN.destqbe

```
method setPageValues(formName String, pageName String) Logical

; This method runs a query (mapValQBE) to find all value objects on
; the current form & page, sending the results to a tCursor named
; newValsTc.


var
  newValsTC, valCursor TCursor
  mapValQBE Query
  fieldObj UIObject
endVar

clearPageValues(formName, pageName) ; clear QBEVAL.DB Query Value values

mapValQBE = Query

           qbeMap.db | Form Name   | Page ID     | Query ID   | Object Path |
                                 |  ~formName |  ~pageName | _qID       | Check
  |

           qbeVal.db | Query ID    | Field Name |
                    | _qID        | Check      |

           endQuery

executeQBE(mapValQBE, newValsTc)

valCursor.open("qbeVal.db")
valCursor.edit()

scan newValsTC :
   valCursor.locate("Field Name", newValsTC."Field Name")
   fieldObj.attach(newValsTC."Object Path")
  if not isBlank(fieldObj.value) then
     if isBlank(valCursor."Query Value") then
           valCursor."Query Value" = fieldObj.value
     else
           valCursor."Query Value" = valCursor."Query Value" + " OR " +
  fieldObj.value
     endIf
  endIf
```

```
endScan

valCursor.endEdit()
RETURN TRUE

endMethod
```

VCRisOpen

MASTLIB.VCRisOpen

Sets a flag to indicate that a custom SpeedBar is open.

```
method VCRisOpen()
  VCROpenLg = True
endMethod
```

vcrName

MASTLIB.vcrName

Returns the text in the title bar of a custom SpeedBar.

```
method vcrName() String
    ; return Videbar title
    return "Click button to change record"
endMethod
```

VCRisClosed

MASTLIB.VCRisCLosed

Sets a flag to indicate that a custom SpeedBar is closed.

```
method VCRisClosed()
  ; videobar closed
  VCROpenLg = False
endMethod
```

MASTLIB Var Window

These variables are visible to all methods and procedures in this library.

```
Var
  canClose, isCalled, VCROpenLg Logical
  TitleAndSource DynArray[] String ; 2 fields: LookUpTitle and ListSource
  myMsgTc, objHelpTc Tcursor
  qRpt Report
  aboutFm Form
endVar
```

MASTLIB Const Window

These constants are visible to all methods and procedures in this library.

```
Const
  proghelp = ":WORK:Mastprog.hlp"
endConst
```

DIVECUST

Methods

open

mouseEnter

mouseExit

mouseRightUp

Var and Uses windows

Var window

Uses window

Methods and Procedures from MASTLIB.LSL

MASTObjMsg

MASTLIB.open

Objects contained by DIVECUST

#Page2

diveCust Uses Window

```
Uses objectPAL

  ; Custom methods contained in MASTLIB.LIB (library)

  mastOBJMsg(formName String, pageName String, objectID String)
  createObjMenu(var formName Form,  pageName String, uio UIObject)

endUses
```

See also
MASTObjMsg

diveCust Var Window

```
Var

  mastLIB Library  ; Pointer to MASTLIB.LIB, the library
                   ; of ObjectPAL routines that are shared
                   ; between this form, DIVEPLAN, and other
                   ; forms in this application directory


endVar
```

open

diveCust.open

```
; Paradox for Windows sample application, 03/29/93

method open(var eventInfo Event)

; This method verifies that the user opened this form from
; the directory containing the data files.  If so, then it
; opens MASTLIB; otherwise, it displays an error message and
; exits.

  if eventInfo.isPreFilter() then  ; Execute for every object on form.

  else                             ; Execute only for the form itself.

     if not isFile("DIVECUST.FSL") then
           disableDefault
           msgStop("Directory Error", "The Paradox for Windows " +
                   "working directory must be set to the directory " +
                   "containing this form, for example: " +
                "C:\\PDOXWIN\\DIVEPLAN.")
           formReturn(0)
     else
           mastLib.open("MASTLIB", globalToDesktop)
     endIf

  endIf

endMethod
```

See also

MASTLIB.open

mouseEnter

diveCust.mouseEnter

```
method mouseEnter(var eventInfo MouseEvent)

; This method displays a help message in the status line for each
; object as the user moves the pointer over it.

var

  uiTarget uiObject  ; Holds the name of the object that the pointer is
                     ; currently pointing to.

endVar

  if eventInfo.isPreFilter() then  ; Execute for each object on form.

     eventInfo.getTarget(uiTarget)
     mastLib.mastObjMsg("diveCust", "thePage", uiTarget.Name)

  else                             ; Execute only for the form itself.

  endIf

endMethod
```

See also

MASTObjMsg

mouseExit

diveCust.mouseExit

```
method mouseExit(var eventInfo MouseEvent)

; This method clears a pointer help message from the status bar

  if eventInfo.isPreFilter() then  ; Execute for each object on form

     message("")

  else                             ; Execute only for the form itself.
                                   ; (This branch here because it was
  endIf                            ; automatically entered by Paradox
                                   ; when this method was created.  It
                                   ; remains for clarity.)

endMethod
```

mouseRightUp

diveCust.mouseRightUp

```
method mouseRightUp(var eventInfo MouseEvent)

; This method displays an "object inspector" menu for
; several objects on the form.  These menus let the user
; browse the ObjectPAL source code, via a compiled WINHELP
; Help file.  To accomplish this, the method calls a custom
; method (createObjMenu()) in MASTLIB.

var

  uiTarget uiObject ; The object that was right-clicked
  thisForm Form     ; Form variable for MASTLIB.createObjMenu()
  Obj_Name String   ; Used to make sure we search for the
                    ; right object when we call createObjMenu.

endVar


  if eventInfo.isPreFilter() then  ; Execute for each object on the form.

     ; First, get the object that was clicked.

     eventInfo.getTarget(uiTarget)

     ; Because fields and buttons are made up of other objects, the object
     ; that received the mouse click may not be the one the user thinks it
     ; is.  So, we'll see if the name of the target object is one we gave
     ; it.  If not, the name will start with a pound sign (#) and we need
     ; to work with its container.

     obj_Name = uiTarget.Name
     While obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
           uiTarget.attach(uiTarget.containerName)
           obj_Name = uiTarget.Name
     endWhile

     ; uiTarget is now attached to a named object (or the form), so
     ; attach the form variable to the form that's currently running
     ; (namely, this one) and then call createObjMenu (to display the
     ; object inspector menu).

     thisForm.attach()
     mastLib.createObjMenu(thisForm, "thePage", uiTarget)

     ; If the user right-clicked an object that can be selected,
     ; then place an event in the queue that selects it.

     if uiTarget.class = "Field" then
           If uiTarget.tabStop then
                 self.postAction(uiTarget.moveTo())
           endIf
```

```
    endIf

  else                              ; Execute only for the form itself.

  endIf

endMethod
```

See also
MASTObjMsg

DIVECUST.#Page2

Methods
open

Var window
Var window

Objects contained by #Page2
Help button
Oops button
Okay button
Panel|VCR
Customer
Dive Flag Box

DIVECUST.#Page2 Var Window

```
Var

  cust_TC  tCursor

endVar
```

open

diveCust.#Page2.open

```
method open(var eventInfo Event)

  self.action(dataBeginEdit)

endMethod
```

pushButton

diveCust.#Page2.btn_Help.pushButton

```
method pushButton(var eventInfo Event)

const

  helpFile   = ":WORK:mastuser.hlp" ; path to Help file
  headerHelp = LongInt(20006)

endConst

  doDefault
  helpShowContext(helpFile, headerHelp)

endMethod
```

pushButton

diveCust.#Page2.btn_Oops.pushButton

```
method pushButton(var eventInfo Event)

  okToMove = True
  formReturn("Cancel")

endMethod
```

pushButton

diveCust.#Page2.btn_Okay.pushButton

```
method pushButton(var eventInfo Event)

  self.action(dataEndEdit)
  formReturn(Customer_No)

endMethod
```

DIVECUST.#Page2.panel|VCR

The VCR panel has no methods attached to it.

Objects contained by DIVECUST

Delete Customer button

Add Customer button

Next Set button

Last Record button

Next Record button

Previous Record button

Previous Set button

First Record button

pushButton

diveCust.#Page2.del_Cust.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataDeleteRecord)

endMethod
```

pushButton

diveCust.#Page2.add_Cust.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataInsertRecord)

endMethod
```

pushButton

diveCust.#Page2.next_Set.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataNextSet)

endMethod
```

pushButton

diveCust.#Page2.last_Rec.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataEnd)

endMethod
```

pushButton

diveCust.#Page2.next_Rec.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataNextRecord)

endMethod
```

pushButton

diveCust.#Page2.prev_Rec.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataPriorRecord)

endMethod
```

.pushButton

diveCust.#Page2.prev_Set.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataPriorSet)

endMethod
```

pushButton

diveCust.#Page2.firstRec.pushButton

```
method pushButton(var eventInfo Event)

  active.postAction(dataBegin)

endMethod
```

DIVECUST.#Page2.Customer

Methods

arrive

canDepart

action

Var window

Var window

DIVECUST.#Page2.Customer Var Window

```
Var

  checkRec Logical

endVar
```

arrive

diveCust.#Page2.Customer.arrive

```
method arrive(var eventInfo MoveEvent)

; This method makes sure each record has a key value by
; checking to see if we arrive on a record without a
; Customer No.


  doDefault; Complete any normal processing

  ; Cust_TC is defined in the Var method for this record
  ; object.  We use this technique of generating a new
  ; ID number because we are working with ID numbers that
  ; do not correspond to record numbers, and the table
  ; is relatively small.  However, if we were working
  ; with a large table or one on a network, we would
  ; use the "separate table" approach described in
  ; the chapter on data model objects in your ObjectPAL documentation.

  ; If you have the luxury of starting ID numbers from
  ; scratch, consider using a technique based on nRecords()
  ; instead of cMax.  When working with large tables, this
  ; approach is much faster than cMax().

  If Customer_No.isBlank() then

     cust_TC.attach(Customer_No)
     newIDNo = cust_TC.cMax("Customer No") + 1
     Customer_No = newIDNo
     checkRec = Yes

  else
     checkRec = No
  endIf

endMethod
```

canDepart

diveCust.#Page2.Customer.canDepart

```
method canDepart(var eventInfo MoveEvent)

; This method makes sure the current customer record can be saved.
; It demonstrates how you can force fields to contain values without
; resorting to a Required validity check.

var

   msg_Info      String  ; Holds part of the message displayed to the user
                         ; when an attempt is made to post an incomplete
  record.

endVar

  ; First, assign a value to the variable, so we can use it as a test
  ; near the end of the method.

   msg_Info = ""

  ; Next, if the record should be verified and has been changed, then
  ; we'll check it out.  (checkRec is a Logical variable defined in
  ; this object's action method.  By default, it is True, but is set
  ; to False when a user chooses to delete the current record.)

  If checkRec and self.locked then

     switch

           case Name.isBlank() :            ; Is there a customer name?
                 msg_Info = "a customer name"

           case Street.isBlank() :          ; Is there a street address?
                 msg_Info = "a street address"

           case Phone.isBlank() :           ; Is there a phone number?
                 msg_Info = "a phone number"

           case City.isBlank() and State_Prov.isBlank() and
                Country.isBlank() :         ; Information to direct a
  letter?

                 msg_Info = "location information (city, state/prov, or
  country)"

           case Zip_Postal_Code.isBlank() : ; a postal code?
                 msg_Info = "a postal code"

     endSwitch

     ; If msg_Info contains a value, then we know the record doesn't
     ; contain enough information.  When that happens, display an error
```

```
    ; message to the user and prevent Paradox from leaving the record.

    If msg_Info <> "" then
          beep()
          msgStop("Incomplete Customer Information", "You need to enter "
  +
                  msg_Info + " before this record can be saved.")
          eventInfo.setErrorCode(-1)
    endIf
  endIf

endMethod
```

action

diveCust.#Page2.Customer.action

```
method action(var eventInfo ActionEvent)

; This method lets us control a little bit of what happens
; behind the scenes.  It contains the code to verify a
; record should be deleted and the code to follow a record
; if it flies away when posted.

var

  actionID smallInt ; Holds the id of the action
  Response String   ; Holds a user's response to a dialog
  curr_Obj,         ; Keeps track of the current object,
  temp_Obj uiObject ; in case the action fails

endVar

  ; First, let's turn the pointer to the hourglass,
  ; indicating that something's going on, and then
  ; delay any screen updates until just before the
  ; user can work with them.

  setMouseShape(mouseWait)

  ; Now, attach a temporary variable to the active
  ; object, in case this action fails.

  curr_Obj.attach(active.Name)

  ; Save the id of this action to a variable, so we don't
  ; have to call a function every time we need to check its
  ; value.  This improves performance.

  actionID = eventInfo.id()

  switch

     ; The user chose to delete a record, so let's pop up
     ; a dialog making sure they really want to delete it.

     case actionID = dataDeleteRecord :

           Response = msgQuestion("Are you sure?", "If you " +
                      "choose Yes, this customer's information " +
                      "will be permanently deleted.  Choose No " +
                  "to Cancel.")

           ; If the user chose not to delete the record, we
           ; want to prevent anything else from happening.
           ; Otherwise, we'll set a flag which disables the
           ; validity checking in the canDepart method.
```

```
            If Response = "No" then
                  eventInfo.setErrorCode(-1)
            else
                  checkRec = False
            endIf

      ; If the user chose to unlock a record, we want to
      ; follow it if it flies away.  The easiest way to
      ; do this, in this form, is to force a dataPostRecord,
      ; which is similar to the Record | Post/Keep Locked
      ; command.

      case actionID = dataUnlockRecord or
           actionID = dataToggleLockRecord:

            ; Because this could be caused by the user locking
            ; the record (using Record | Lock), we want
            ; to post the record only if it's already locked.

            If self.locked then
                  If not action(dataPostRecord) then
                        eventInfo.setErrorCode(-1)
                  endIf
            endIf

      ; When the record gets locked, we want to make sure
      ; it's valid when we try to leave it, so we'll wait
      ; for that event and set the flag used by canDepart
      ; accordingly.

      case actionID = dataLockRecord :
            checkRec = True

   endSwitch

   ; Now, let's see if the action succeeded.  If not, force
   ; focus to the object that was active when this action
   ; started.  This makes sure everything gets
   ; repainted properly (and prevents things like
   ; disappearing highlights.)

   If eventInfo.errorCode() <> 0 then
      temp_Obj.attach(curr_Obj.containerName)
      temp_Obj.moveTo()
      curr_Obj.moveTo()
   endIf

   ; We're done, so reset the pointer and show any screen
   ; updates.

   setMouseShape(mouseArrow)

endMethod
```

mouseClick

diveCust.#Page2.diveFlagBox.mouseClick

```
method mouseClick(var eventInfo MouseEvent)

  checkRec = False

endMethod
```

rptsList

Methods
open

Objects contained by
#Page2

open

rptsList.open

```
method open(var eventInfo Event)

; This method makes sure RPTSLIST is opened by another form; if it
; isn't, then an error message appears and the form closes.

var
  formList Array[] String
endVar

  if not eventInfo.isPreFilter() then ; Execute only for the form.
     enumFormNames(formList)
     If not formList.contains("MAST Vacation Order Form") then
           beep()
           msgStop("Open Error", "You can't open this form directly." +
                   "  It must be opened from the MASTORD form.")
           close()
     endIf
  endif

endmethod
```

RPTSLIST.#Page2

Methods

menuAction

Objects contained by

btnHelp

btnOops

btnOkay

rptType

menuAction

rptsList.#Page2.menuAction

```
method menuAction(var eventInfo MenuEvent)

; This method converts the menuControlClose action (generated
; when a user double-clicks the Control menu) to a Cancel
; action.  It also disables the Maximize and Minimize icons
; when the form is called from DIVEPLAN.FSL.

var

  actionID smallInt  ; Saves eventInfo.id() to a variable for
                     ; best performance

  calledBy Form      ; Holds the handle of the form that opened
                     ; this form

endVar

  actionID = eventInfo.id()
  Switch

     case actionID = MenuControlClose :

           disableDefault
           btnOops.pushButton()

     case actionID = MenuControlMaximize or
          actionID = MenuControlMinimize :

           If formCaller(calledBy) then
                 disableDefault
                 beep()
                 msgInfo("Oops!", "Because this form was opened " +
                         "by " + calledBy.getTitle() + ", the " +
                         "Maximize and Minimize buttons are " +
                         "disabled.  Sorry...")
           endIf
  endSwitch

endMethod
```

RPTSLIST.#Page2.btnHelp

Methods

open

mouseEnter

mouseExit

mouseDown

mouseUp

Var window

Var window

RPTSLIST.#Page2.btnHelp Var window

RPTSLIST.#Page2.btnHelp.Var

```
Var
  btnIsDown Logical
endVar
```

open

rptsList.#Page2.btnHelp.open

```
method open(var eventInfo Event)
  btnUp.visible = True
  btnDown.visible = True
  btnIsDown = False
endMethod
```

mouseEnter

rptsList.#Page2.btnHelp.mouseEnter

```
method mouseEnter(var eventInfo MouseEvent)
If eventinfo.isleftDown() AND btnIsDown  Then
  btnUp.visible = False
endIf

endMethod
```

mouseExit

rptsList.#Page2.btnHelp.mouseExit

```
method mouseExit(var eventInfo MouseEvent)
If eventInfo.isLeftDown() Then
  btnUp.visible = True
endIf
endMethod
```

mouseDown

rptsList.#Page2.btnHelp.mouseDown

```
method mouseDown(var eventInfo MouseEvent)
  btnUp.visible = False
  btnIsDown = True
endMethod
```

mouseUp

rptsList.#Page2.btnHelp.mouseUp

```
method mouseUp(var eventInfo MouseEvent)
  btnUp.visible = True
  If NOT (btnIsDown AND self.hasMouse()) Then
     btnIsDown = False
     Return
  endIf
  btnIsDown = False
  ; ---------------------------------------;
  ;  Insert your 'pushbutton' code here     ;
  ; ---------------------------------------;


endMethod
```

RPTSLIST.#Page2.btnOops

Methods

open

mouseEnter

mouseExit

mouseDown

mouseUp

pushButton

Var window

Var window

RPTSLIST.#Page2.btnOops Var window

```
Var
  btnIsDown Logical
endVar
```

open

rptsList.#Page2.btnOops.open

```
method open(var eventInfo Event)
  btnUp.visible = True
  btnDown.visible = True
  btnIsDown = False
endMethod
```

mouseEnter

rptsList.#Page2.btnOops.mouseEnter

```
method mouseEnter(var eventInfo MouseEvent)
If eventinfo.isleftDown() AND btnIsDown  Then
  btnUp.visible = False
endIf

endMethod
```

mouseExit

rptsList.#Page2.btnOops.mouseExit

```
method mouseExit(var eventInfo MouseEvent)
If eventInfo.isLeftDown() Then
  btnUp.visible = True
endIf
endMethod
```

mouseDown

rptsList.#Page2.btnOops.mouseDown

```
method mouseDown(var eventInfo MouseEvent)
  btnUp.visible = False
  btnIsDown = True
endMethod
```

mouseUp

rptsList.#Page2.btnOops.mouseUp

```
method mouseUp(var eventInfo MouseEvent)
  btnUp.visible = True
  If NOT (btnIsDown AND self.hasMouse()) Then
     btnIsDown = False
     Return
  endIf
  btnIsDown = False
  self.pushButton()
endMethod
```

pushButton

rptsList.#Page2.btnOops.pushButton

```
method pushButton(var eventInfo Event)

  okToMove = True
  If customer.Inserting then
     customer.action( dataDeleteRecord )
  endIf
  formReturn("Cancel")

endMethod
```

RPTSLIST.#Page2.btnOkay

Methods

open

mouseEnter

mouseExit

mouseDown

mouseUp

pushButton

Var window

Var window

RPTSLIST.#Page2.btnOkay Var window

```
Var
  btnIsDown Logical
endVar
```

open

rptsList.#Page2.btnOkay.open

```
method open(var eventInfo Event)
  btnUp.visible = True
  btnDown.visible = True
  btnIsDown = False
endMethod
```

mouseEnter

rptsList.#Page2.btnOkay.mouseEnter

```
method mouseEnter(var eventInfo MouseEvent)
If eventinfo.isleftDown() AND btnIsDown  Then
  btnUp.visible = False
endIf

endMethod
```

mouseExit

rptsList.#Page2.btnOkay.mouseExit

```
method mouseExit(var eventInfo MouseEvent)
If eventInfo.isLeftDown() Then
  btnUp.visible = True
endIf
endMethod
```

mouseDown

rptsList.#Page2.btnOkay.mouseDown

```
method mouseDown(var eventInfo MouseEvent)
  btnUp.visible = False
  btnIsDown = True
endMethod
```

mouseUp

rptsList.#Page2.btnOkay.mouseUp

```
method mouseUp(var eventInfo MouseEvent)
  btnUp.visible = True
  If NOT (btnIsDown AND self.hasMouse()) Then
     btnIsDown = False
     Return
  endIf
  btnIsDown = False
  self.pushButton()

endMethod
```

pushButton

rptsList.#Page2.btnOkay.pushButton

```
method pushButton(var eventInfo Event)

  formReturn(rptType)

endMethod
```

open

rptsList.#Page2.rptType.open

```
method open(var eventInfo Event)

; This method initializes the rptTypes field to the first value in the
  list.
; Note that there is a difference between the labels shown to the user and
; the values given to the field.  This was done by filling the list with
; the values the developer wanted to work with and then editing the text
; labels of each radio button to the values the developer wanted the user
; to see.
;
; (To edit a text label while designing a form, select it with the mouse,
; press <F2> to enter Field View, and then use standard Windows editing
; techniques and keystrokes to change the text.  When finished, press <F2>
; to exit Field View.)

  self.value = "quikInv"

endMethod
```