Listing 4-1: Routines for reading and writing to the parallel port registers and for reading, setting, clearing, and toggling individual bits in a byte. (Sheet 1 of 2)

```
Function BitRead% (Variable%, BitNumber%)
'Returns the value (0 or 1) of the requested bit in a Variable.
Dim BitValue%
'the value of the requested bit
BitValue = 2 ^ BitNumber
BitRead = (Variable And BitValue) \ BitValue
End Function
-----------------------------------
Sub BitReset (Variable%, BitNumber%)
'Resets (clears) the requested bit in a Variable.
Dim BitValue, CurrentValue%
'the value of the requested bit
BitValue = 2 ^ BitNumber
Variable = Variable And (&HFFFF - BitValue)
End Sub
----------------------------------
Sub BitSet (Variable%, BitNumber%)
'Sets the requested bit in a Variable.
Dim BitValue, CurrentValue%
'the value of the requested bit
BitValue = 2 ^ BitNumber
Variable = Variable Or BitValue
End Sub
-----------------------------------
Sub BitToggle (Variable%, BitNumber%)
'Toggles the requested bit in a Variable.
Dim BitValue, CurrentValue%
'the value of the requested bit
BitValue = 2 ^ BitNumber
'Is the current value 0 or 1?
CurrentValue = Variable And BitValue
Select Case CurrentValue
Case 0
'If current value = 0, set it
Variable = Variable Or BitValue
Case Else
'If current value = 1, reset it
Variable = Variable And (&HFFFF - BitValue)
End Select
End Sub
```

Listing 4-1: Routines for reading and writing to the parallel port registers and for reading, setting, clearing, and toggling individual bits in a byte. (Sheet 2 of 2)

```
Function ControlPortRead% (BaseAddress%)
'Reads a parallel port's Control port.
'Calculates the Control-port address from the port's
'base address, and inverts bits 0, 1, & 3 of the byte read.
'The Control-port hardware reinverts these bits,
'so the value read matches the value at the connector.
ControlPortRead = (Inp(BaseAddress + 2) Xor &HB)
End Function
Sub ControlPortWrite (BaseAddress%, ByteToWrite%)
'Writes a byte to a parallel port's Control port.
'Calculates the Control-port address from the port's
```

```
'base address, and inverts bits 0, 1, & 3.
'The Control-port hardware reinverts these bits,
'so Byte is written to the port connector.
Out BaseAddress + 2, ByteToWrite Xor &HB
End Sub
Function DataPortRead% (BaseAddress%)
'Reads a parallel port's Data port.
DataPortRead = Inp(BaseAddress)
End Function
Sub DataPortWrite (BaseAddress%, ByteToWrite%)
'Writes a byte to a parallel port's Data port.
Out BaseAddress, ByteToWrite
End Sub
Function StatusPortRead% (BaseAddress%)
'Reads a parallel port's Status port.
'Calculates the Status-port address from the port's
'base address, and inverts bit 7 of the byte read.
'The Status-port hardware reinverts these bits,
'so the value read matches the value at the connector.
StatusPortRead = (Inp(BaseAddress + 1) Xor &H80)
End Function
```

Listing 4-2: Code for Figure 4-1's form that enables users to find, test, and select ports. (Sheet 2 of 4)

```
Private Sub cmdFindPorts_Click()
'Test the port at each of the standard addresses,
'and at the non-standard address, if the user has entered one.
Dim Index%
Dim PortExists%
Dim Count%
Index = 0
'First, test address 3BCh
Port(Index).Address = &H3BC
PortExists = TestPort(Index)
'If the port exists, increment the index.
If Not (Port(Index).Address) = 0 Then
Index = Index + 1
End If
'Test address 378h
Port(Index).Address = &H378
PortExists = TestPort(Index)
'If the port exists, increment the index.
If Not (Port(Index).Address) = 0 Then
Index = Index + 1
End If
'Test address 278h
Port(Index).Address = &H278
PortExists = TestPort(Index)
'Disable option buttons of unused LPT ports
For Count = Index + 1 To 2
optPortName(Count).Enabled = False
Port(Count).Enabled = False
Next Count
If Not (Port(3).Address = 0) Then
PortExists = TestPort(Index)
Else
optPortName(3).Enabled = False
```

```
End If
End Sub
Private Sub cmdOK_Click()
frmSelectPort.Hide
End Sub
```

Listing 4-2: Code for Figure 4-1‘s form that enables users to find, test, and select ports. (Sheet 3 of 4)

```
Private Sub cmdTestPort_Click()
Dim PortExists%
Dim Index%
‘Get the address of the selected port
Index = -1
Do
Index = Index + 1
Loop Until optPortName(Index).Value = True
PortExists = TestPort(Index)
Select Case PortExists
Case True
MsgBox “Passed: Port “ + Hex$(BaseAddress) + _
“h is “ + Port(Index).PortType + “.”, 0
Case False
MsgBox “Failed port test. “, 0
End Select
End Sub
```

Listing 4-2: Code for Figure 4-1‘s form that enables users to find, test, and select ports. (Sheet 4 of 4)

```
Private Sub Form_Load()
Dim Index%
Left = (Screen.Width - Width) / 2
Top = (Screen.Height - Height) / 2
‘Load the combo boxes with the ECP modes.
For Index = 0 To 3
cboEcpMode(Index).AddItem “SPP (original)”
Next Index
For Index = 0 To 3
cboEcpMode(Index).AddItem “bidirectional”
Next Index
For Index = 0 To 3
cboEcpMode(Index).AddItem “Fast Centronics”
Next Index
For Index = 0 To 3
cboEcpMode(Index).AddItem “ECP”
Next Index
For Index = 0 To 3
cboEcpMode(Index).AddItem “EPP”
Next Index
‘Enable the option buttons for existing ports.
For Index = 0 To 3
optPortName(Index).Enabled = Port(Index).Enabled
Next Index
UpdateLabels
End Sub
Private Sub optPortName_Click(Index As Integer)
‘Store the address and index of the selected port.
```

```
Dim Count%
BaseAddress = Port(Index).Address
IndexOfSelectedPort = Index
EcpDataPortAddress = BaseAddress + &H400
EcrAddress = BaseAddress + &H402
For Count = 0 To 3
cboEcpMode(Count).Enabled = False
Next Count
cboEcpMode(Index).Enabled = True
End Sub
```

Listing 4-3: The startup form for the sample project is blank except for a menu. You can add whatever controls you need for a specific application.

```
Private Sub Form_Load()
StartUp
End Sub
Private Sub Form_Unload(Cancel%)
ShutDown
End
End Sub
Private Sub mnuPort_Click(Index%)
frmSelectPort.Show
End Sub
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 1 of 14)

```
#If Win32 Then
Declare Function GetPrivateProfileStringByKeyName& Lib _
“Kernel32” Alias “GetPrivateProfileStringA” _
(ByVal lpApplicationName$, ByVal lpszKey$, ByVal lpszDefault$, _
ByVal lpszReturnBuffer$, ByVal cchReturnBuffer&, ByVal lpszFile$)
Declare Function WritePrivateProfileString& Lib _
“Kernel32” Alias “WritePrivateProfileStringA” _
(ByVal lpApplicationName$, ByVal lpKeyName$, ByVal lpString$, _
ByVal lpFileName$)
Declare Function GetWindowsDirectory& Lib “Kernel32” _
Alias “GetWindowsDirectoryA” (ByVal lpBuffer$, ByVal nSize%)
#Else
Declare Function GetPrivateProfileStringByKeyName% Lib “Kernel” _
Alias “GetPrivateProfileString” _
(ByVal lpApplicationName$, ByVal lpKeyName$, ByVal lpDefault$, _
ByVal lpReturnedString$, ByVal nSize%, ByVal lpFileName$)
Declare Function WritePrivateProfileString% Lib “Kernel” _
(ByVal lpApplicationName$, ByVal lpKeyName$, _
ByVal lpString$, ByVal lpFileName$)
Declare Function GetWindowsDirectory% Lib “Kernel” _
(ByVal lpBuffer$, ByVal nSize%)
#End If
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 2 of 14)

```
Type PortData
Name As String
Address As Integer
PortType As String
EcpModeDescription As String
EcpModeValue As Integer
Enabled As Integer
End Type
```

```
Global Port(0 To 3) As PortData
Global BaseAddress%
Global PortType$
Global IniFile$
Global EcrAddress%
Global EcrData%
Global EcpDataPortAddress%
Global EppDataPort0Address%
Global IndexOfSelectedPort%
Global PortDescription$
Global EcpExists%
Global SppExists%
Global PS2Exists%
Global EppExists%
Function GetEcpModeDescription$(EcpModeValue%)
Select Case EcpModeValue
Case 0
GetEcpModeDescription = "SPP"
Case 1
GetEcpModeDescription = "PS/2"
Case 2
GetEcpModeDescription = "Fast Centronics"
Case 3
GetEcpModeDescription = "ECP"
Case 4
GetEcpModeDescription = "EPP"
Case 6
GetEcpModeDescription = "Test"
Case 7
GetEcpModeDescription = "Configuration"
End Select
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 3 of 14)

```
Sub GetIniData()
'Use the Windows API call GetPrivateProfileString to read
'user information from an ini file.
Dim NumberOfCharacters
Dim ReturnBuffer As String * 128
Dim Index%
Dim WindowsDirectory$
'Get the Windows directory, where the ini file is stored.
NumberOfCharacters = GetWindowsDirectory(ReturnBuffer, 127)
WindowsDirectory = Left$(ReturnBuffer, NumberOfCharacters)
IniFile = WindowsDirectory + "\lptprogs.ini"
'If the ini file doesn't exist, don't try to read it.
If Not Dir$(IniFile) = "" Then
'The port addresses:
Port(0).Address = _
CInt(VbGetPrivateProfileString("lptdata","Port0Address",
IniFile))
Port(1).Address = _
CInt(VbGetPrivateProfileString("lptdata","Port1Address",
IniFile))
Port(2).Address = _
CInt(VbGetPrivateProfileString("lptdata","Port2Address",
IniFile))
Port(3).Address = _
CInt(VbGetPrivateProfileString("lptdata","Port3Address",
IniFile))
'The port types:
Port(0).PortType = _
VbGetPrivateProfileString("lptdata", "Port0Type", IniFile)
Port(1).PortType = _
VbGetPrivateProfileString("lptdata", "Port1Type", IniFile)
Port(2).PortType = _
VbGetPrivateProfileString("lptdata", "Port2Type", IniFile)
Port(3).PortType = _
VbGetPrivateProfileString("lptdata", "Port3Type", IniFile)
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 4 of 14)

```
'Port enabled?
Port(0).Enabled = _
CInt(VbGetPrivateProfileString("lptdata",
"Port0Enabled",IniFile))
Port(1).Enabled = _
CInt(VbGetPrivateProfileString("lptdata",
"Port1Enabled",IniFile))
Port(2).Enabled = _
CInt(VbGetPrivateProfileString("lptdata",
"Port2Enabled",IniFile))
Port(3).Enabled = _
CInt(VbGetPrivateProfileString("lptdata",
"Port3Enabled",IniFile))
'The selected port
IndexOfSelectedPort = _
Int(VbGetPrivateProfileString("lptdata", _
"IndexOfSelectedPort", IniFile))
End If
End Sub
Function ReadEcpMode%(TestAddress%)
'The Ecr mode is in bits 5, 6, and 7 of the ECR.
EcrAddress = TestAddress + &H402
EcrData = Inp(EcrAddress)
ReadEcpMode = (EcrData And &HE0) \ &H20
End Function
Function ReadEppTimeoutBit%(BaseAddress%)
'Reads and clears the EPP timeout bit (Status port bit 0).
'Should be done after each EPP operation.
'The method for clearing the bit varies, so try 3 ways:
'1. Write 1 to Status port bit 0.
'2. Write 0 to Status port, bit 0.
'3. Read the Status port again.
Dim StatusPortAddress%
StatusPortAddress = BaseAddress + 1
ReadEppTimeoutBit = BitRead(StatusPortRead(BaseAddress), 0)
Out StatusPortAddress, 1
Out StatusPortAddress, 0
ReadEppTimeoutBit = BitRead(StatusPortRead(BaseAddress), 0)
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 5 of 14)

```
Sub SetEcpMode(EcpModeValue%)
'Store the Ecp mode's value and description in the Port array.
Port(IndexOfSelectedPort).EcpModeValue = EcpModeValue
Port(IndexOfSelectedPort).EcpModeDescription = _
GetEcpModeDescription(EcpModeValue)
EcrAddress = BaseAddress + &H402
'Read the ECR & clear bits 5, 6, 7.
EcrData = Inp(EcrAddress) And &H1F
'Write the selected value to bits 5, 6, 7.
EcrData = EcrData + EcpModeValue * &H20
Out EcrAddress, EcrData
End Sub
Sub ShutDown()
WriteIniData
End
End Sub
Sub StartUp()
Dim PortExists%
Dim Index%
'Get information from the ini file.
GetIniData
'Load the forms.
frmMain.Left = (Screen.Width - frmMain.Width) / 2
frmMain.Top = (Screen.Height - frmMain.Height) / 2
Load frmSelectPort
frmSelectPort.optPortName(IndexOfSelectedPort).Value = True
frmMain.Show
End Sub
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 6 of 14)

```
Function TestForEcp%(TestAddress%)
'Test for the presence of an ECP.
'If the ECP is idle and the FIFO empty,
'in the ECP's Ecr (at Base Address+402h),
'bit 1(Fifo full)=0, and bit 0(Fifo empty)=1.
'The first test is to see if these bits differ from the
'corresponding bits in the Control port (at Base Address+2).
'If so, a further test is to write 34h to the Ecr,
'then read it back. Bit 1 is read/write, and bit 0 is read-only.
'If the value read is 35h, the port is an ECP.
Dim EcrBit0%, EcrBit1%
Dim ControlBit0%, ControlBit1%
Dim ControlPortData%
Dim TestEcrAddress%
Dim OriginalEcrData%
TestForEcp = False
EcrAddress = TestAddress + &H402
'Read ECR bits 0 & 1 and Control Port bit 1.
EcrData = Inp(EcrAddress)
EcrBit0 = BitRead(EcrData, 0)
EcrBit1 = BitRead(EcrData, 1)
ControlPortData = ControlPortRead(TestAddress)
ControlBit1 = BitRead(ControlPortData, 1)
If EcrBit0 = 1 And EcrBit1 = 0 Then
```

```
'Compare Control bit 1 to ECR bit 1.
'Toggle the Control bit if necessary,
'to be sure the two registers are different.
If ControlBit1 = 0 Then
ControlPortWrite TestAddress, &HF
ControlPortData = ControlPortRead(TestAddress)
ControlBit1 = BitRead(ControlPortData, 1)
End If
If EcrBit1 <> ControlBit1 Then
OriginalEcrData = EcrData
Out EcrAddress, &H34
EcrData = Inp(EcrAddress)
If EcrData = &H35 Then
TestForEcp = True
End If
'Restore the ECR to its original value.
Out EcrAddress, OriginalEcrData
End If
End If
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 7 of 14)

```
Function TestForEpp%(TestAddress%)
'Write to an Epp register, then read it back.
'If the reads match the writes, it's probably an Epp.
'Skip this test if TestAddress = 3BCh.
Dim ByteRead%
Dim StatusPortData%
Dim EppAddressPort%
Dim TimeoutBit%
Dim StatusPortAddress%
StatusPortAddress = TestAddress + 1
TestForEpp = False
'Use EppAddressPort for testing.
'SPPs, ECPs, and PS/2 ports don't have this register.
EppAddressPort = TestAddress + 3
Out EppAddressPort, &H55
'Clear the timeout bit after each EPP operation.
TimeoutBit = ReadEppTimeoutBit%(TestAddress%)
ByteRead = Inp(EppAddressPort)
TimeoutBit = ReadEppTimeoutBit%(TestAddress%)
If ByteRead = &H55 Then
Out EppAddressPort, &HAA
TimeoutBit = ReadEppTimeoutBit%(TestAddress%)
ByteRead = Inp(EppAddressPort)
TimeoutBit = ReadEppTimeoutBit%(TestAddress%)
If ByteRead = &HAA Then
TestForEpp = True
End If
End If
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 8 of 14)

```
Function TestForPS2%(TestAddress%)
'Tests a parallel port's Data port for bidirectional ability.
'First, try to tri-state (disable) the Data outputs by
```

```
'setting bit 5 of the Control port.
'Then write 2 values to the Data port and read each back
'If the values match, the Data outputs are not disabled,
'and the port is not bidirectional.
'If the values don't match,
'the Data outputs are disabled and the port is bidirectional.
Dim DataInput%
Dim ControlPortData%
Dim OriginalControlPortData%
Dim OriginalDataPortData%
'Set Control port bit 5.
ControlPortWrite TestAddress, &H2F
TestForPS2 = False
'Write the first byte and read it back:
DataPortWrite TestAddress, &H55
DataInput = DataPortRead(TestAddress)
'If it doesn't match, the port is bidirectional.
If Not DataInput = &H55 Then TestForPS2 = True
'If it matches, write another and read it back.
If DataInput = &H55 Then
DataPortWrite TestAddress, &HAA
DataInput = DataPortRead(TestAddress)
'If it doesn't match, the port is bidirectional
If Not DataInput = &HAA Then
TestForPS2 = True
End If
End If
'Reset Control port bit 5
ControlPortWrite TestAddress, &HF
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 9 of 14)

```
Function TestForSpp%(TestAddress%)
'Write two bytes and read them back.
'If the reads match the writes, the port exists.
Dim ByteRead%
'Be sure that Control port bit 5 = 0 (Data outputs enabled).
ControlPortWrite TestAddress, &HF
TestForSpp = False
DataPortWrite TestAddress, &H55
ByteRead = DataPortRead(TestAddress)
If ByteRead = &H55 Then
DataPortWrite TestAddress, &HAA
ByteRead = DataPortRead(TestAddress)
If ByteRead = &HAA Then
TestForSpp = True
End If
End If
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 10 of 14)

```
Function TestPort%(PortIndex%)
'Test for a port's presence, and if it exists, the type of port.
'In order, check for presence of ECP, EPP, SPP, and PS/2 port.
'Update the information in the Port array and the display.
Dim EcpModeDescription$
Dim EcpModeValue%
Dim TestAddress%
TestPort = False
EcpExists = False
EppExists = False
SppExists = False
PS2Exists = False
PortType = ""
TestAddress = Port(PortIndex).Address
'Begin by hiding all port details.
frmSelectPort.lblAddress(PortIndex).Visible = False
frmSelectPort.lblType(PortIndex).Visible = False
frmSelectPort.cboEcpMode(PortIndex).Visible = False
EcpExists = TestForEcp(TestAddress)
If EcpExists Then
PortType = "ECP"
'Read the current Ecp mode.
EcpModeValue = ReadEcpMode(TestAddress)
Else
'If it's not an ECP, look for an EPP.
'If TestAddress = 3BCh, skip the EPP test.
'EPPs aren't allowed at 3BCh due to possible conflict
'with video memory.
frmSelectPort.cboEcpMode(PortIndex).Visible = False
If TestAddress = &H3BC Then
EppExists = False
Else
EppExists = TestForEpp(TestAddress)
End If
frmSelectPort.cboEcpMode(PortIndex).Visible = False
EppExists = TestForEpp(TestAddress)
If EppExists Then
PortType = "EPP"
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 11 of 14)

```
Else
'If it's not an EPP, look for an SPP.
SppExists = TestForSpp(TestAddress)
If SppExists Then
'Test for a PS/2 port only if the SPP exists
'(because if the port doesn't exist,
'it will pass the PS/2 test!)
PS2Exists = TestForPS2(TestAddress)
If PS2Exists Then
PortType = "PS/2"
Else
PortType = "SPP"
End If
Else
PortType = ""
End If
End If
End If
If PortType = "" Then
frmSelectPort.optPortName(PortIndex).Enabled = False
Port(PortIndex).PortType = ""
Port(PortIndex).Address = 0
Port(PortIndex).Enabled = False
Else
TestPort = True
Port(PortIndex).Enabled = True
Port(PortIndex).PortType = PortType
Port(PortIndex).Enabled = True
If EcpExists Then
Port(PortIndex).EcpModeValue = EcpModeValue
Port(PortIndex).EcpModeDescription = _
GetEcpModeDescription(EcpModeValue)
End If
End If
UpdateLabels
End Function
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 12 of 14)

```
Sub UpdateLabels()
'Use the information in the Port array to update the display.
Dim Index%
Dim EcpModeValue%
For Index = 0 To 3
frmSelectPort.lblAddress(Index).Caption = _
Hex$(Port(Index).Address) + "h"
If Port(Index).Enabled = True Then
frmSelectPort.optPortName(Index).Enabled = True
frmSelectPort.lblAddress(Index).Visible = True
frmSelectPort.lblType(Index).Caption = _
Port(Index).PortType
frmSelectPort.lblType(Index).Visible = True
If Port(Index).PortType = "ECP" Then
EcpModeValue = ReadEcpMode(Port(Index).Address)
frmSelectPort.cboEcpMode(Index).ListIndex = _
EcpModeValue
Port(Index).EcpModeValue = EcpModeValue
Port(Index).EcpModeDescription = _
GetEcpModeDescription(EcpModeValue)
frmSelectPort.cboEcpMode(Index).Visible = True
Else
frmSelectPort.cboEcpMode(Index).Visible = False
End If
Else
frmSelectPort.optPortName(Index).Enabled = False
frmSelectPort.lblAddress(Index).Visible = False
frmSelectPort.lblType(Index).Visible = False
frmSelectPort.cboEcpMode(Index).Visible = False
End If
Next Index
End Sub
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 13 of 14)

```
Sub WriteIniData()
Dim BaseAddressWrite%
Dim PortTypeWrite%
Dim Index%
Dim IniWrite
```

```
'Use Windows API call WritePrivateProfileString to save
'initialization information.
'If the ini file doesn't exist, it will be created and stored in
'the Windows directory.
'The port addresses:
IniWrite = WritePrivateProfileString _
("lptdata", "Port0Address", CStr(Port(0).Address), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port1Address", CStr(Port(1).Address), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port2Address", CStr(Port(2).Address), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port3Address", CStr(Port(3).Address), IniFile)
'The port types:
IniWrite = WritePrivateProfileString _
("lptdata", "Port0Type", Port(0).PortType, IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port1Type", Port(1).PortType, IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port2Type", Port(2).PortType, IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port3Type", Port(3).PortType, IniFile)
'Port enabled?
IniWrite = WritePrivateProfileString _
("lptdata", "Port0Enabled", CStr(Port(0).Enabled), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port1Enabled", CStr(Port(1).Enabled), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port2Enabled", CStr(Port(2).Enabled), IniFile)
IniWrite = WritePrivateProfileString _
("lptdata", "Port3Enabled", CStr(Port(3).Enabled), IniFile)
```

Listing 4-4: Code for finding and testing ports, and getting and saving initialization data from an ini file. (Sheet 14 of 14)

```
'Find the selected port and save it:
Index = 4
Do
Index = Index - 1
Loop Until (frmSelectPort.optPortName(Index).Value = True) _
Or Index = 0
IniWrite = WritePrivateProfileString("lptdata", _
"IndexOfSelectedPort", CStr(Index), IniFile)
End Sub
Function VbGetPrivateProfileString$(section$, key$, file$)
Dim KeyValue$
'Characters returned as integer in 16-bit, long in 32-bit.
Dim Characters
KeyValue = String$(128, 0)
Characters = GetPrivateProfileStringByKeyName _
(section, key, "", KeyValue, 127, file)
KeyValue = Left$(KeyValue, Characters)
VbGetPrivateProfileString = KeyValue
End Function
```