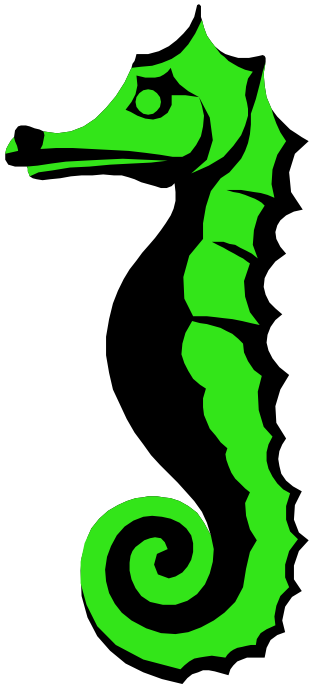
Übersichten und Hinweise zu

ftTeachVB2005

für die Industry Robots
und die ROBO & Intelligent Interfaces
VB2005 Version

Ulrich Müller



Inhaltsverzeichnis

 Dokumentname : ftTeachVB2005.doc. Druckdatum : 18.06.2006
Bild : Einfügen | Graphik | Aus Datei... SEEPFRD7.WMF

Programm Funktionen

Allgemeines

Das Programm ftTeachCS ermöglicht den TeachIn-Betrieb von Robots der fischertechnik Kästen Industry Robots I und II über das ROBO und das Intelligent Interface. Die durch das TeachIn gewonnenen Daten können in einem Scriptfile aufgezeichnet werden und können dann im Run-Modus des Programms als Ausgang für den selbständigen Betrieb des Robots genommen werden.

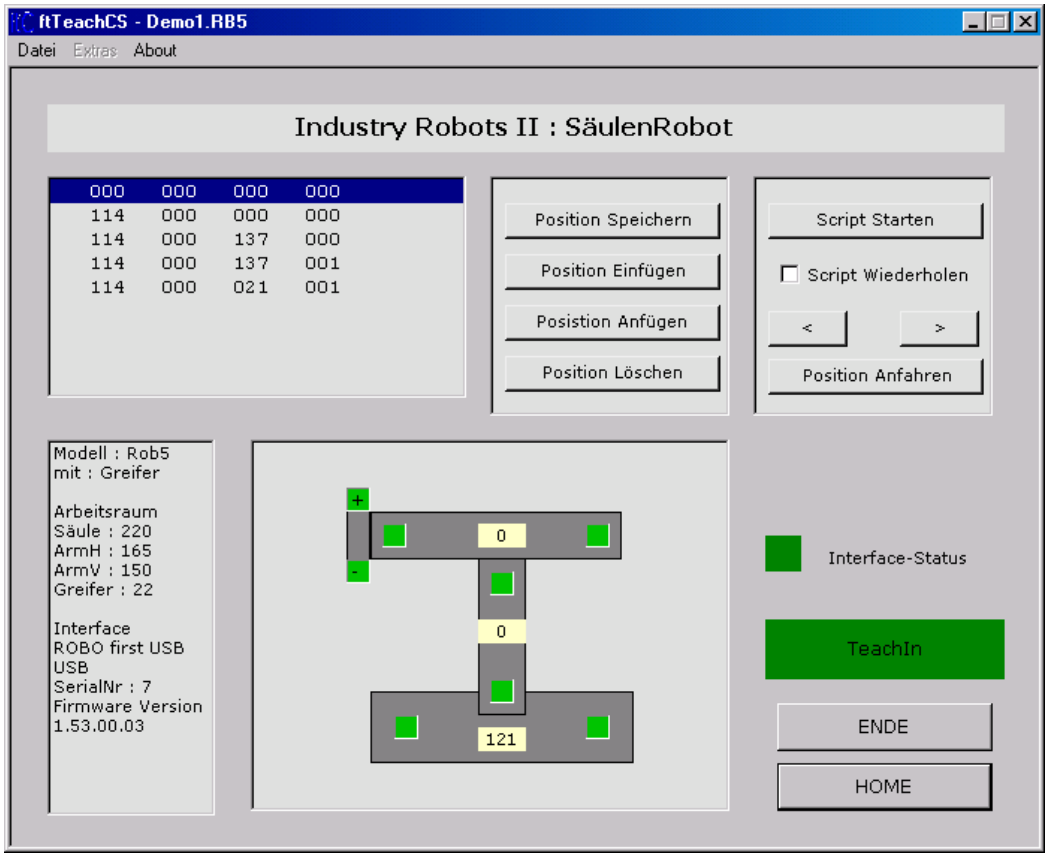
Hauptzweck des in VB2005 (:Net 2.0 Framework) erstellten Programmes ist eine Einführung in ein komplexeres in VB2005 geschriebenes Programm. Wenn man Wert auf ein fertiges TeachIn-Programm legt, sollte man ftTeach40.EXE nutzen (www.ftcomputing.de/ftteach.htm).

HINWEIS : Diese VB2005 Version von ftTeach wurde sehr eng nach der C# Version erstellt, es konnten deswegen (Faulheitshalber) sogar die Bilder dieser Ausgabe genutzt werden (Für den Fall, die Titelzeile der Bilder erregt Anstoß).

Gegenüber der C# Version wurde bei der Serialisierung von Binary auf XML umgestellt, nach kann sie dann mit dem Explorer lesen.

Programmfunktionen

Haupt-Form



Das Programm kann über die Robot-Graphik im TeachIn-Modus den Robot durch Klick auf die grünen Felder auf bestimmte Positionen verfahren werden und die erreichte Position über die Positions-Buttons abspeichern.

Über die Script-Buttons kann dann im Run-Modus das im TeachIn-Modus erstellte Script abgefahren werden.

Über das Datei-Menü kann ein Script abgespeichert werden und später auch wieder geladen werden.

Die Interface- und Modell-Einstellungen können über eigene Formen eingestellt und angepaßt werden.

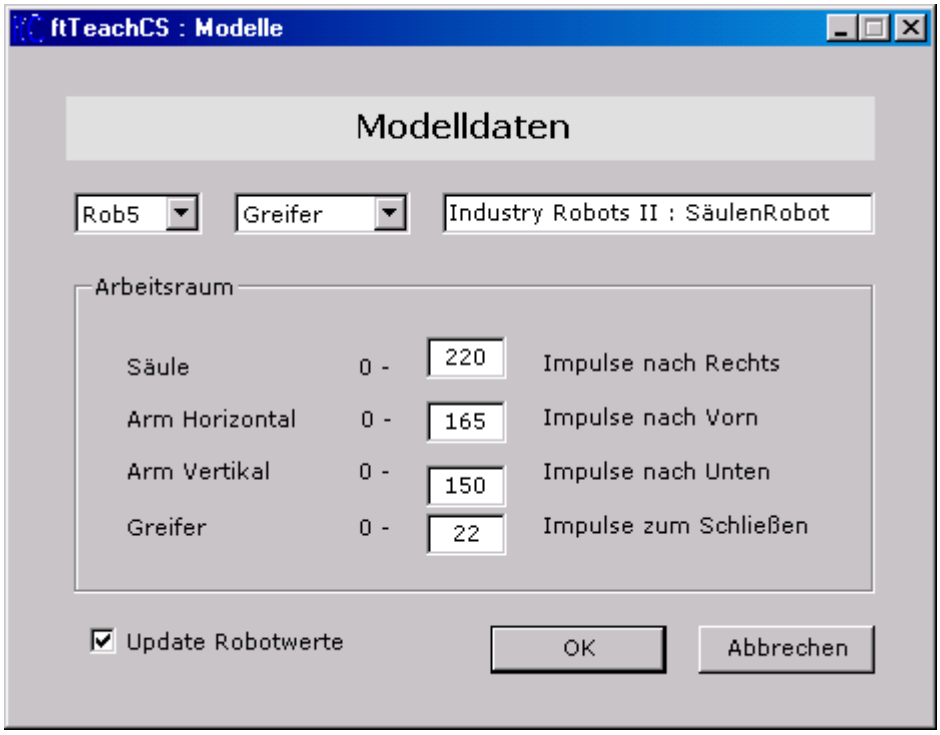
Der Abbruch des Run-Modus kann über einen mit HALT beschrifteten Button (aktuell im Bild mit HOME beschriftet) oder die ESC-Taste erfolgen.

Interface-Einstellungen

Aufruf über Menü | Extras | Interface erlaubt es die Auswahl des aktuell gewünschten Interfaces aus einer Liste der möglichen Interfaces. Bei einem Anschluß über USB kann zusätzlich noch die Serialnummer eingegeben werden. Bei Anschluß über COM kann der COM-Port ausgewählt werden.

Zusätzlich zur der Liste der möglichen Interfaces enthält Position 0 der Liste immer das aktuelle Interface. Es kann gewählt werden, ob SerialNr bzw. COM-Port auch in dieser Liste abgespeichtert werden sollen.

Modell-Einstellungen



Aufruf über Menü | Datei | Modell erlaubt die Auswahl des aktuell gewünschten Modells aus einer Liste der möglichen Modelle.

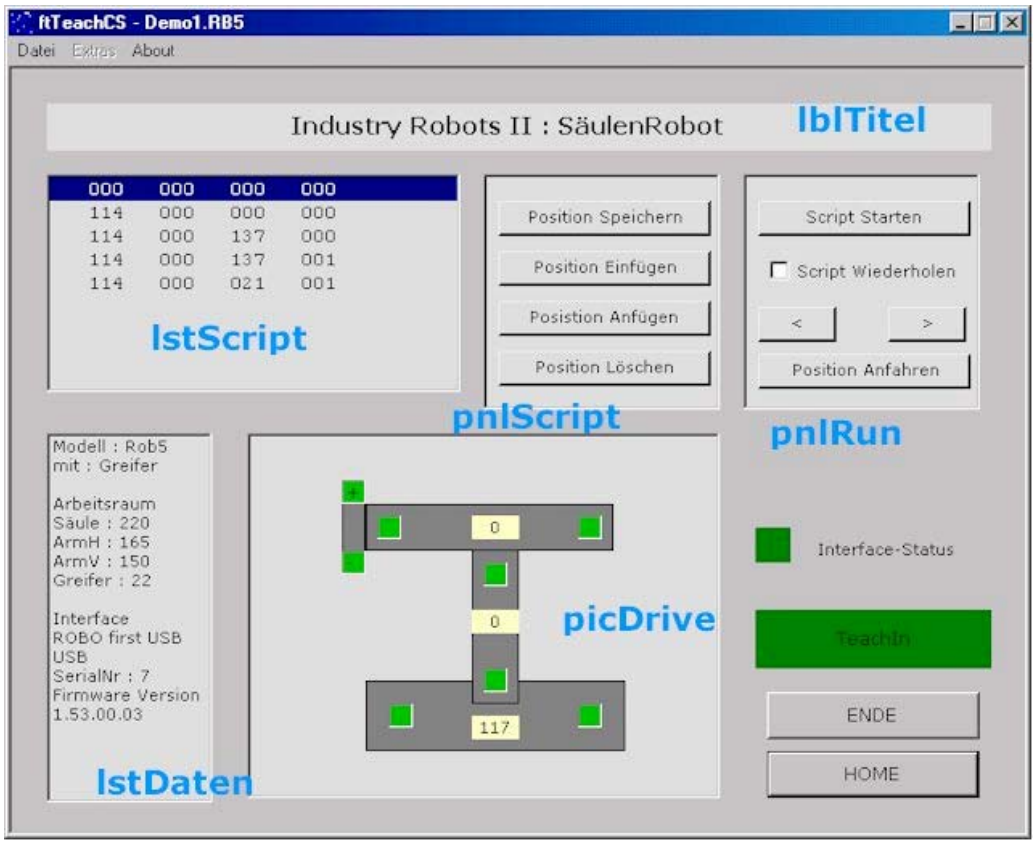
Zusatzgerät, beschreibender Text und Arbeitsraum können modifiziert werden und wahlweise auch in der Liste der möglichen Modelle gespeichert werden, Position 0 enthält immer das aktuelle Modell.

Source Programm

Source

ftTeachMain.VB :	Hauptform mit der wesentlichen Programmlogik
ftTeachDaten.VB :	Klasse mit persistenten Daten (Modell, Interface)
ftTeachModell.VB :	Form zum Festlegen der Modelldaten
ftTeachSchnitt.VB :	Form zum Festlegen der Interfacedaten
ftTeachAbout.VB :	Form zur Anzeige von Programmdaten
FishFace2005.DLL :	Assembly mit Klassen zur Steuerung der Interfaces
FishFace2005.XML :	Die erläuternden Texte dazu

TeachMain



Aufbau : Controls, zugeordnete Aufgaben

lblTitel	Titelzeile mit dem Namen des aktuellen Robots noch offen : dyn. Titel in this.Text
lstScript	ListBox mit dem Daten des aktuellen Scriptfiles.
pnlScript	Panel mit den Buttons für Script Operationen cmdPosSpeichern : Speichern der aktuellen Position in markierte Zeile cmdPosEinfugen : Einfügen der aktuellen Position vor markierte Zeile cmdPosAnfugen : Hinzufügen der aktuellen Position ans Scriptende cmdPosLoschen : Löschen der markierten Position
pnlRun	Panel für Button für den RobotBetrieb mit ScriptDaten. cmdRunStart : Fahren gesamtes Scipt chkRunRepeat : Endlose Wiederholung des Scripts cmdRunBack : Fahren zur Position vor der Markierten cmdRunForward : Fahren zur Position hinder der Markierten cmdRunTo : Fahren zur markierten Position.
lstDaten	Auflistung der aktuellen Robot und InterfaceDaten
picDrive	TeachIn- und AnzeigePanel für den Robot Anzeige | Fahren 1 | Fahren 2 | Position lblSaule, lblSauleLinks, lblSauleRecht, lblSaulePos lblArmH, lblArmHVor, lblArmHRuck, lblArmHPos lblArmV, lblArmVHoch, lblArmVAb, lblArmVPos lblGreifer, lblGreiferAuf,

cmdAction	Click-Ereignis : EIN : Starten Interface STARTEN : Starten Robot-Betrieb mit HomeRun HALT : Anhalten Robot-Betrieb
cmdEnde	Click-Ereignis Beenden des Programms im Zusammenspiel mit dem Form_Closed Ereignis
lblStatus	Anzeige des Betriebsstatus
lblBetrieb	Anzeige des Interfacestatus
mnuModell	Aufruf der Modell-Form
mnuAbout	Aufruf der About-Form
mnuInterface	Aufruf der Interface-Form
mnuNeu	Starten mit neuem Script-File
mnuOffnen	Öffnen eines vorhandenen Script-Files
mnuSpeichern	Speichern des aktuellen Script-Files
mnuSpeichernU	Speichern des aktuellen Script-Files unter neuem Namen
tmrI	Überwachung des Interfaces
Positionsanzeige	Anzeige der aktuellen Robot-Position und der Robot-Aktivitäten Ereignis-Routine aus FishFace / FishRobot

VB2005 Programmiertechniken

Serialisierung / Persistenz

Einige Programmdaten, die Interfacedaten und die Modelldaten werden über den Programmlauf hinaus in einem XML File gespeichert. Dazu wird die .NET Fähigkeit der Serialisierung genutzt (`Imports System.Xml.Serialization`). Dazu müssen die zu speichernden Daten in Klassen zusammengefaßt werden. Hier ist das die Klasse ftTeachDaten im gleichnamigen File. Es enthält eine Liste der verfügbaren Interfaces und RobotModelle. Diese Listen können im Programm editiert werden. Auf Position 0 der Liste befindet sich das aktuell gewählte Interface bzw. RobotModell.

Das File mit den persistenten Daten im XML-Format liegt im Pfad der ftTeachVB2005.EXE.

```
Private IniName As String = New IO.DirectoryInfo(".\").FullName & _
                            "ftTeachDaten.XML"
```

Im `ftTeachMain_Load` Ereignis wird eine Instanz (pd) der Klasse ftTeachDaten erzeugt :

```
If IO.File.Exists(IniName) Then
   Dim ser As New XmlSerializer(GetType(ftTeachDaten))
   Dim xml As New IO.StreamReader(IniName, IO.FileMode.Open)
   pd = CType(ser.Deserialize(xml), ftTeachDaten)
   xml.Close()
Else
   pd = New ftTeachDaten()
End If
Me.Left = pd.FormLeft
Me.Top = pd.FormTop
Me.Text = Application.ProductName
```

Wenn kein ftTeachDaten.XML vorhanden ist, werden die Default-Daten aus der Klassen-Instanz pd von ftTeachDaten genutzt. Ansonsten wird der XmlSerializer (ser) und ein Hilfsfile (xml) aufgesetzt, der Serializer liest dann mit einem Befehl die gespeicherten Daten und deserialisiert sie dabei.

Im `ftTeachMain_FormClosed` Ereignis werden die aktuellen Daten der Instanz pd von ftTeachDaten dann serialisiert :

```
Dim ser As New XmlSerializer(GetType(ftTeachDaten))
Dim xml As New IO.StreamWriter(IniName)
pd.FormLeft = Me.Left
pd.FormTop = Me.Top
ser.Serialize(xml, pd)
xml.Close()
```

Es werden wieder ser und xml aufgesetzt und dann mit einem Befehl Serialisiert.

Script-Files

Die Script-Files sind einfache Text-Files, die die Liste der anzufahrenden Positionen enthalten. Reihenfolge : Säule, Arm waagerecht, - senkrecht, Zusatz / Greifer (Offen 0, geschlossen 1) :

```
Private Sub mnuOffnen_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles mnuOffnen.Click
' --- Einlesen eines ScriptFiles nach lstScript ---
  Dim Z As String
  DirtySpeichern()
  openScript.InitialDirectory = Application.StartupPath
  If openScript.ShowDialog() = Windows.Forms.DialogResult.OK Then
    ScriptName = openScript.FileName
    Dim sr As New IO.StreamReader(ScriptName)
```

```
    Z = sr.ReadLine()
    lstScript.Items.Clear()
    Do Until IsNothing(Z)
      lstScript.Items.Add(Z)
      Z = sr.ReadLine()
    Loop
    sr.Close()
    lstScript.SelectedIndex = 0
    Me.Text = Application.ProductName & " - " & _
              ScriptName.Substring(ScriptName.LastIndexOf("\") + 1)
    IsDirty = False
  End If
End Sub
```

In mnuOffnen wird zum Öffnen der normale DateiDialog geführt und dann über den System.IO.StreamReader zeilenweise (ReadLine) gelesen und in die ListBox lstScript abgestellt. Der Name des gelesenen Files wird zusätzlich in die Titelzeile der Form abgestellt. Das Speichern geschieht analog über mnuSpeichern, mnuSpeichernU unter Nuzung der Routine Speichern (sr.WriteLine()).

In beiden Fällen wird ein IsDirty-Flag beachtet über das ggf. (IsDirty == true) eine Abfrage gestartet wird, ob das aktuelle Script-File gespeichert werden soll.

Die aktuelle Position (erreichte / anzufahrende) wird in einem Array MoveWerte gehalten (MoveToScript() / ScriptToMove()).

Der Aufruf der Routinen erfolgt über Menü.

Programm-Ablaufsteuerung

Um Fehlbedienungen, die zu einem Crash des Modells führen können, schon im Vorfeld ausschließen zu können, werden die infrage kommenden Controls der Bedieneroberfläche in Abhängigkeit eines "Programm-Modus" matrixartig gesteuert (Routine ModusWechsel). In der Routine werden alle betroffenen Controls auf dem zum Programm-Modus(Ein, Home,TeachIn, Run) passenden Stand gesetzt (z.B. Enabled = true;)

Der Programm-Ablauf selber wird durch den Button cmdAction gesteuert :

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
  Select Case cmdAction.Text
  Case "EIN"
    Try
      Dim MotList(,) As Integer = {{1, Modell.SauleMax}, _
                                   {2, Modell.ArmHMax}, _
                                   {3, Modell.ArmVMax}, _
                                   {4, Modell.GreiferMax}}
      ft = New FishRobot(MotList)
      If Anschluss.PortNr > 0 Then      ' --- Anschluss an COM
        ft.OpenInterface(Anschluss.Typ, Anschluss.PortNr, 999, True)
      Else                              ' --- Anschluss an USB
        ft.OpenInterface(Anschluss.Typ, Anschluss.SerialNr, True)
      End If
      Anschluss.FirmwareVersion = ft.ActDevice.Firmware
      Anschluss.SerialNr = ft.ActDevice.SerialNr
      tmrI.Enabled = True
      ModellDaten()
      ModusWechsel(PrgModus.Home)
    Catch eft As FishFaceException
      lblStatus.Text = eft.Message
      lblStatus.BackColor = Color.Red
      ft.CloseInterface()
    End Try
  Case "HOME"
```

```
    Try
      ModusWechsel(PrgModus.Run)
      HomeRun()
      ModusWechsel(PrgModus.TeachIn)
    Catch eft As FishFaceException
      lblStatus.Text = eft.Message
      lblStatus.BackColor = Color.Red
    End Try
  Case "HALT"
    ft.NotHalt = True
    chkRunRepeat.Checked = False
  End Select
End Sub
```

Bei Start des Programms hat er die Beschriftung EIN, es wird das zuvor ausgewählte Interface gestartet. Den Achsen des Modells werden dabei Motornummer und maximaler Fahrweg zugeordnet. Das OpenInterface geschieht in Abhängigkeit vom Interface-Anschluß. Nach erfolgreichem Open werden die aktuellen Interface-Daten ausgelesen und angezeigt. Die Beschriftung des Buttons wechselt auf HOME (in ModusWechsel).

In den anderen Fällen ist der Button mit HALT beschriftet, ein Setzen von ft.NotHalt = true hält den Robot an.

TeachIn-Steuerung

Auf dem picDrive ist aus einer Reihe von Labeln ein Robot-Symbol zusammen gebaut worden. Der recheckige (grüne) Label zur Steuerung der einzelnen Robot-Komponeten über Maus genutzt werden :

```
Private Sub lblRobMouseDown(ByVal sender As System.Object, _
            ByVal e As System.Windows.Forms.MouseEventArgs) _
  Handles lblSauleRechts.MouseDown, lblSauleLinks.MouseDown, _
          lblArmVAb.MouseDown, lblArmHVor.MouseDown, _
          lblArmHRuck.MouseDown, lblArmHoch.MouseDown
  If IsNothing(ft) Then Return
  Dim L As Label = sender
  ft.NotHalt = False
  Try
    Select Case L.Tag
    Case 1                  ' --- Säule zum Endtaster
      ft.MoveTo(0, ft.MotCntl(1).actPos, ft.MotCntl(2).actPos)
    Case 2                  ' --- Säule weg vom Endtaster
      ft.MoveTo(ft.MotCntl(0).maxPos, ft.MotCntl(1).actPos, _
                ft.MotCntl(2).actPos)
    Case 3                  ' --- Arm, horizontal, zurück
      ft.MoveTo(ft.MotCntl(0).actPos, 0, ft.MotCntl(2).actPos)
    Case 4                   ' --- Arm, horizontal, vor
      ft.MoveTo(ft.MotCntl(0).actPos, ft.MotCntl(1).maxPos, _
                ft.MotCntl(2).actPos)
    Case 5                   ' --- Arm, vertikal, auf
      ft.MoveTo(ft.MotCntl(0).actPos, ft.MotCntl(1).actPos, 0)
    Case 6                   ' --- Arm, vertikal, ab
      ft.MoveTo(ft.MotCntl(0).actPos, ft.MotCntl(1).actPos, _
                ft.MotCntl(2).maxPos)
    End Select
  Catch eft As FishFaceException
    lblBetrieb.BackColor = Color.Red
    lblStatus.Text = eft.Message
  End Try
End Sub
```

Genutzt werden die MouseDown- und MouseUp-Ereignisse der Labels. Bei MouseDown wird der zugehörende Motor gestartet, bei MouseUp wieder angehalten. Zum Motorbetrieb

wird die Methode MoveTo von FishRobot genutzt. Es müssen deswegen alle Positionen des Robots angegeben werden. Für den betroffenen Motor (in der Numerierung der Instanzierung) wird für Fahren nach links (auf den Endtaster zu) als ZielPosition 0 angegeben, bei Fahren nach rechts die max. mögliche. Für die nicht betroffenen Motoren wird deren aktuelle Postion angegeben (keine Angst, intern wird das erkannt, der Motor wird nicht gestartet). Positionen hinter der betroffenen können entfallen.

Um mit einer Ereignis-Routine auszukommen, wurde der Motor und die anzufahrende Richtung in der Tag-Eigenschaft des jeweiligen Label codiert und hier in einem Select Case Konstrukt ausgewertet.

Das Anhalten des Motors (aller Motoren) geschieht durch Setzen von ft.NotHalt = true; Deswegen auch zu Anfang von MouseDown ft.NotHalt = false.

Der Greifer bzw. ein Zusatzgerät wird separat behandelt, da es in Praxis wenig sinnvoll ist, eine Fahrbewegung des Robot mit einer Schließbewegeung zu koppeln. Das Öffnen und Schließen des Greifers geschieht stets vollständig (lblGreiferClick).

Status-Anzeige

Der Interface-Betrieb wird durch eine Timer-Routine (tmrI) überwacht (lblBetrieb grün/rot). Es ist darauf zu achten, dass der Timer vor Programmende abgeschaltet wird.

Der aktuelle Robot-Betrieb (einschließlich laufender Positionsangabe) wird in einer Ereignis-Routine von ft (FishRobot) angezeigt :

```
Private Sub ft_PositionChange(ByVal sender As Object, _
          ByVal ActPositions() As Integer) Handles ft.PositionChange
  Try
    Dim EW As Integer = ft.Outputs
    lblSaulePos.Text = ActPositions(0)
    If (EW And &H3) > 0 Then
      lblSaule.BackColor = Color.Green
    Else
      lblSaule.BackColor = Color.Gray
    End If
    lblArmHPos.Text = ActPositions(1)
    If (EW And &HC) > 0 Then
      lblArmH.BackColor = Color.GreenYellow
    Else
      lblArmH.BackColor = Color.Gray
    End If
      lblArmVPos.Text = ActPositions(2)
    If (EW And &H30) > 0 Then
      lblArmV.BackColor = Color.GreenYellow
    Else
      lblArmV.BackColor = Color.Gray
    End If
  Catch eft As FishFaceException
    lblBetrieb.BackColor = Color.Red
    lblStatus.Text = eft.Message
  End Try
End Sub
```

Dazu wird die ft.Outputs-Eigenschaft entsprechend maskiert. Eine Anzeige in der Timer-Routine wäre genauso möglich gewesen, die Teilung ist eher Geschmacksache. In jedem Fall ist

```
  Private WithEvents ft As FishRobot
  ft.OpenInterface(Anschluss.Typ, Anschluss.PortNr, 999, True)
```

Mit dem Parameter DoEvents = True aufzurufen (kann weggelassen werden, da default = True ist). Da die auslösende MoveTo Methode den Rechner sehr beschäftigt, könnten sonst von Windows die Anzeige-Labels der nicht mehr upgedatet werden. Bei einem COM-Anschluß sollte der Parameter Analogzyklen auf einen hohen Wert (hier 999) gesetzt werden, da sonst schon mal ein Impuls "verschluckt" werden kann. Kleinere Werte sind nur erforderlich, wenn am Robot irgendwo noch ein Analog-Eingang ausgelesen werden soll.