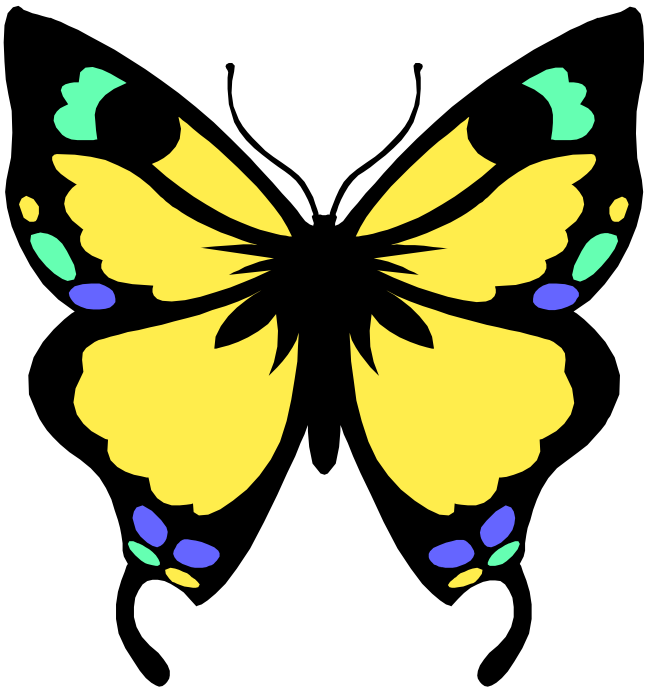
Eine Einführung in die Programmierung mit

VB2005 und FishFace2005.DLL

Ulrich Müller



Inhaltsverzeichnis

Windows.Forms Programme 21

Ulrich Müller, eMail : um@ftcomputing.de Homepage : www.ftcomputing.de

Dokumentname : FishKursVB2005.doc. Download : fishkursvb2005.zip
Druckdatum : 17.07.2006 Bild : Einfügen | Grafik | AusDatei | OFFICE | SCHMTTL3.WMF

Übersichten

Allgemeines

Zweck des Tutorials

Einführung in die Programmierung mit FishFace unter Nutzung von VB2005. Der Schwerpunkt des Tutorials liegt auf der Einarbeitung in VB2005. Dazu wird durchgehend ein Ampelbeispiel von einem einfachen Konsolprogramm bishin zu einem Schaltschrank an einer Kreuzungsecke diskutiert. Es wird sowohl der Einsatz der FishFace-Methoden wie auch der erforderlichen VB2005 vorgestellt.

Installation und Literatur

Microsoft stellt die Express Version von VB2005 kostenlos zur Verfügung. Am einfachsten kommt man zu einem VB2005 Exemplart in Verbindung mit einem Einführungsbuch, z.B. :

Peter Bloch : Einstieg in Visual Basic 2005, Galileo Computing,
ISBN : 3-89842-641-6, Preis 24,90 €

Das Buch beschreibt die Installation und die Entwicklungsumgebung. Über den Index des Buches kann man vollständigere Angaben zu den hier unter der Überschrift "VB2005 Themen" angegeben Hinweisen erhalten.

Ulrich Müller : FishFace40 für VB 2005, www.ftcomputing.de/zip/vb2005fish40setup.exe
Einführung, Referenz und Klassenbibliothek FishFace2005.DLL

Bei weitergehendem Interesse ist :

Michael Kofler : Visual Basic 2005, Addison-Wesley,
ISBN : 3-8273-2338-X, Preis 59,95 €

interessant. Achtung : enthält keine VB2005 CD (nur Beispiel CD).

Reihenfolge

Man sollte das Tutorial in der Reihenfolge Konsol-Programme – Windows.Forms-Programme durcharbeiten. Die Programme sind von der Funktionalität weitgehend gleich, bei den Windows.Forms-Programmen wird dann zusätzlich besonders auf die erweiterten Möglichkeiten der Window-Anwender-Oberfläche eingegangen. Funktionen, die schon bei den Konsol-Programmen beschrieben wurden, werden dann als bekannt vorausgesetzt.

Anschluß der Interfaces

Bei allen Beispielprogrammen wird das erste ROBO Gerät an USB genutzt :

```
ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
```

Soll ein bestimmtes Interface oder ein an COM angeschlossenes Interface genutzt werden, so ist das `ft.OpenInterface` entsprechend zu ändern, eine Interface-Auswahl beim Start des Programmes wird nicht angeboten.

Nutzung des Intelligent Interface

Ist möglich. Allerdings werden dort die O-Ausgänge nicht unterstützt. Es steht dementsprechen nur die Methode SetMotor (nicht SetLamp) zur Verfügung. Bei Nutzung der Methode SetMotors ist daran zu denken, daß die Lampen 2polig angeschlossen sind und dementsprechend auch zwei bits belegen.

Beispiele

Eine Projektmappe mit zwei Projekten und jeweils mehreren Beipielen. Um ein bestimmtes Beispiel zu aktivieren ist das Projekt als Aktives Projekt festzulegen (Projektmappen-Explorer) und dann über Projekt | Eigenschaften das gewünschte Beispiel als Startformular auszuwählen (Projektmappen-Explorer | Eigenschaften | Anwendung)

Konsole : Ampeln

- FishKurs1 : Einfache Ampel, M-Ausgänge
- FishKurs1e : Exceptions (Try, Catch), Texte (WriteLine)
- FishKurs1n : Nachtschaltung, Sub, Const, Select Case, If
- FishKurs2 : Fußgängerampel, O-Ausgänge
- FishKurs2b : Bedarfsschaltung
- FishKurs3 : Kreuzung, SetMotors
- FishKurs3t : Nach Liste, Taktung, Bedarf , Thread
- FishKurs4 : LinksAbbieger, I/O Extension. Nur Hinweis, keine Source

Windows.Forms : Ampeln

- FishKurs2W : Fußgängerampel, Nachtschaltung, Blinken
- FishKurs2bW : FußgängerAmpel, Bedarfsschaltung, Nachtabschaltung
- FishKurs3W : Kreuzung
- FishKurs3bW : Kreuzung mit Bedarfsschaltung, Lösung über Timer

Konsolprogramme

Allgemeines

ROBO Interface an USB / COM, bedingt einsetzbar ist auch das Intelligent Interface (Einpolige Schaltung der Lampen kann problematisch sein).

ROBO Pro : Versionskontrolle, Firmware laden, ftUSB.SYS

FishPanel : Interface Test

Buch und VB2005 CD : Galileo

Was ist ein Konsolprojekt

FishKurs1 : Einfache Ampel

Das Modell

Lampe grün an Ausgang M1
Lampe gelb an Ausgang M2
Lampe rot an Ausgang M3

Befestigt an ein paar Bausteinen 30, ggf. mit Distanzstücken dazwischen. Die Bausteine 30 können einfach am Interface befestigt werden.

Das Programm

```
Imports FishFace40

Module FishKurs1
  Private ft As New FishFace()

  Sub Main()
    ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    Do                    ' --- Betriebsschleife Ampelanzeige ---
      ft.SetMotor(Out.M1, Dir.Ein)    '     Grün, 3000
      ft.Pause(3000)
      ft.SetMotor(Out.M1, Dir.Aus)
      ft.SetMotor(Out.M2, Dir.Ein)    '     Gelb, 1000
      ft.Pause(1000)
      ft.SetMotor(Out.M2, Dir.Aus)
      ft.SetMotor(Out.M3, Dir.Ein)    '     Rot, 3500
      ft.Pause(3500)
      ft.SetMotor(Out.M2, Dir.Ein)    '     Rot/Gelb, 1000
      ft.Pause(1000)
      ft.SetMotor(Out.M3, Dir.Aus)
      ft.SetMotor(Out.M2, Dir.Aus)
    Loop Until ft.Finish()            ' --- Ende durch ESC-Taste
    ft.CloseInterface()
    Console.ReadLine()
  End Sub

End Module
```

In einer Endlosschleife (Do ... Loop) werden die Lampen der Ampel über die FishFace.Methode SetMotor geschaltet.

VB2005 Themen

Datei | Neues Projekt

Anlegen eines Konsol-Projektes

1. Menü | Datei | Neues Projekt | KonsolAnwendung | Projektname : FishKurs
2. Dateiname Module1 in FishKurs1 ändern
3. Menü Projekt | Verweis hinzufügen | Durchsuchen | C:\Programme\ftComputing\FishFace2005.DLL
4. Erste Zeile in FishKurs1.vb einfügen : Imports FishFace40

5. Erste Zeile im Module FishKurs : Private ft As New FishFace()
6. Speichern

Module

Programmrahmen bei Konsolprojekten.

Die Datei FishKurs1.vb (Module1.vb) besteht nach dem Anlegen des Projekts aus dem Module FishKurs1 (ex Module1) mit der Sub Main(), mit der das Konsolprogramm gestartet wird.

Options

Die Options Explicit On und Strict Off sind default und werden hier vorausgesetzt.

Imports

Verweis auf einen Namensraum einer Assembly

Hier Verweis auf die FishFace2005.DLL (Verweis hinzufügen) und deren Nutzung über den Imports des zugehörenden Namensraums FishFace40. Gilt für die jeweilige Source.

ft

Instanzieren einer Klasse

Hier wird zur Nutzung eines ROBO Interfaces durch Instanzieren (New) ein Objekt (ft) der Klasse FishFace angelegt. Sie kann aus dem gesamten Programm zugegriffen werden.

ft.SetMotor

Nutzung von Methoden eines Objektes

Die Objekte einer Klasse bieten in der Regel eine Reihe von Eigenschaften und Methoden, die durch voranstellen des Objektnamens genutzt werden können.
z.B. ft.SetMotor(Out.M1, Dir.Links) : Schalten des Interface-Ausganges auf Links-Lauf.

Out.M1

Aufzählung (Enum)

Eine besondere Form von Konstanten, die logisch zusammengehören, hier die Aufzählung der Ausgänge eines ROBO Interfaces z.B. Out.M1 für den Ausgang M1

Do ... Loop ..

Schleifenbefehl

Alle Befehle zwischen Do und Loop werden wiederholt, bis die Schleife beendet (Until ft.Finish()) wird. Hier geschieht das durch ft.Finish() = true. Das bedeutet im aktuellen Fall ein Drücken der ESC-Taste.

Console.ReadLine

Tastatureingabe

Einlesen einer Zeile (Text und RETURN-Taste) von der Tastatur. Hier wird Console.ReadLine eingesetzt um das Konsole-Fenter "offen zu halten". Bei Programmende wird es standardmäßig geschlossen, man kann die Ausgaben dann nicht mehr kontrollieren. Da keine wirkliche Eingabe mehr erwartet wird, reicht die RETURN-Taste um dann das Fenster endgültig zu schließen. Eine evtl. zusätzliche Eingabe wird verworfen.

FishFace Themen

OpenInterface

Auswahl eines Interfaces, Herstellen einer Verbindung zum Interface.

```
ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
```

Mit der Methode OpenInterface des Objekts ft der Klasse FishFace wird ft angewiesen mit dem ersten ROBO Interface, das an USB gefunden wird (0 : SerialNr beliebig), zu arbeiten und die Verbindung dazu herzustellen.

Auch bei den folgenden Beispielen wird mit diesem Wert gearbeitet. Wenn man das ROBO Interface lieber an der seriellen Schnittstelle betreiben will, ist das OpenInterface zu modifizieren :

```
ft.OpenInterface(IFTypen.ftROBO_IF_COM, Port.COM1, 0)
```

Hier wird es an COM1 betrieben (0 : Eingeschränkte Auswertung auch der Analogeingänge).

CloseInterface

Beenden der Verbindung zum Interface

SetMotor

Schalten eines M-Ausganges

```
ft.SetMotor(Out.M1, Dir.Links)
```

heißt, schalten des Ausganges M1 am Interface auf Links-Betrieb. Die Enum's sind nicht zwingend, verdeutlichen aber die Sache. Möglich wäre auch ft.SetMotor(1, 1). Zusätzlich ist auch noch eine Geschwindigkeitsangabe (bei Lampen eine Angabe der Helligkeit) möglich :

```
ft.SetMotor(Out.M1, Dir.Links, Speed.Half)
```

Pause

Anhalten des Programmablaufs

Ein SetMotor ist ein asynchroner Befehl, d.h. der Motor läuft (die Lampe leuchtet) bis sie wieder ausgeschaltet wird. Wie lange sie leuchten soll wird hier durch ft.Pause(1000) = 1 Sekunde festgelegt.

Finish

Anmelden eines Abbruchwunsches

Programme mit "endlosen" Schleifen lassen sich gar nicht so einfach beenden, eine einfache Methode ist ein Until ft.Finish() am Schleifenende. Hier wird abgefragt, ob inzwischen die ESC-Taste gedrückt wurde, dann wird die Schleife beendet. Da die Pause-Methoden ebenfalls abbrechen, wenn ESC gedrückt wurde, geht das dann ganz schnell.

FishKurs1e : Ausbau

Modell wie gehabt

Das Programm

```
Imports FishFace40
Module FishKurs1e
  Private ft As New FishFace()
  Const mRot As Out = Out.M3
  Const mGelb As Out = Out.M2
  Const mGruen As Out = Out.M1

  Sub Main()
    Console.WriteLine("--- FishKurs1t : gestartet ---")
    Try
      ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
      Console.WriteLine("--- Verbindung zum Interface ---")
      Console.WriteLine("--- Ampel läuft, Abbruch : ESC ---")
      Do                ' --- Betriebsschleife Ampelanzeige ---
        ft.SetMotor(mGruen, Dir.Ein)    '     Grün, 3000
        ft.Pause(3000)
        ......
      Loop Until ft.Finish()           ' --- Ende durch ESC-Taste
    Catch eft As FishFaceException
      Console.WriteLine(eft.Message)
    End Try
    ft.CloseInterface()
    Console.WriteLine("--- Beenden durch RETURN ---")
    Console.ReadLine()
  End Sub

End Module
```

VB2005 Themen

Symbolische Konstanten

Verwenden von Konstanten zur Bezeichnung von Inhalten

```
Const mRot as Out = Out.M3
```

Während die Enumeration Out.M3 einen Hinweis auf den Anschluß am Interface gibt, wird durch die Konstante mRot die Nutzung des Interface-Ausganges beschrieben (m soll für M-Ausgang stehen). Man kann statt mRot natürlich auch noch ausführlicher RoteLampe oder LampeRot sagen.

Konsolausgaben

Die Klasse Console bietet eine Reihe von allgemeinen (statischen) Methoden für die Ein- und Ausgabe. Hier wird WriteLine() zur Ausgabe von Textzeilen im Konsolfenster. Im ersten Programm wurde bereits Console.ReadLine() (Lesen einer Zeile von der Tastatur) zum Warten auf die RETURN-Taste genutzt.

Exceptions

Abfangen und Behandeln von Ausnahmen (Fehlern)

Wenn in dem durch Try ... Catch geklammerten Bereich ein Fehler auftritt, kann er durch Befehle im Bereich Catch ... End Try abgefangen werden. Die hier behandelten Fehler werden durch eine Exception Klausel (hier eft FishFaceException) beschrieben. Hier werden also Fehler der Klasse FishFace behandelt. Das ist oft ein fehlerhaftes OpenInterface

FishKurs1n : Nachtschaltung

Modell

Wie gehabt, zusätzlich ein Taster an I1

Das Programm

```
Private Sub NachtZyklus()
  ft.SetMotor(mGelb, Dir.Ein)
  ft.Pause(700)
  ft.SetMotor(mGelb, Dir.Aus)
  ft.Pause(250)
End Sub

Private Sub TagZyklus()
  .... aus der Sub Main() übernommen
End Sub

Sub Main()
  ... OpenInterface ...
  Do                    ' --- Betriebsschleife Ampelanzeige ---
    Select Case TimeString
    Case #6:00:00 AM# To #10:00:00 PM#
      If ft.GetInput(iNacht) Then NachtZyklus() Else TagZyklus()
    Case Else
      NachtZyklus()
    End Select
  Loop Until ft.Finish()   ' --- Ende durch ESC-Taste
  ... CloseInterface ...
End Sub
```

In der Zeit von 10:00 bis 22:00 läuft der normale Tagzyklus sonst wird gelb geblinkt. Für Testzwecke kann das Gelbblinken auch durch Taster iNacht zu Tageszeiten erfolgen.

VB2005 Themen

Sub

Zusammenfassen von Programmteilen zu einer Einheit.

Wenn das Main() zu unübersichtlich werden droht, lagert man am besten Funktionen in Unterprogramme (Sub's) aus. Hier geschehen mit dem TagZyklus, der bisher im Main angesiedelt war. Jetzt bilden die gleichen Befehle die Funktion einer Sub TagZyklus().

Hinzugekommen ist eine vergleichbare SubNachtZyklus().

In einer Sub kann man auf eigene Daten (hier nicht der Fall) zugreifen, auf die Aufrufparameter wie bei den FishFace.Methoden und dann noch auf "globale" Daten, hier das Objekt ft, eine Instanz der Klasse FishFace. Das ist möglich, weil ft auf Modul-Ebene – außerhalb einer Sub deklariert wurde.

Time

Arbeiten mit Datums- und Zeitwerten.

Ist ein manchmal recht vertracktes Geschäft, da Datum und Zeit länderspezifisch dargestellt werden. Im Programm gilt jedoch einheitlich die "amerikanische" Schreibweise von Datum und Zeit, angezeigt wird aber länderspezifisch. Die Zeitkonstante #10:00:00 PM# wird mit Console.WriteLine(#10:00:00 PM#) also als 22:00:00 angezeigt (hier auch noch mit dem Datum 1.1.1). Will man das nicht :

```
Console.WriteLine(#10:00:00 PM#.ToString("HH:mm:ss"))
```

Hier wird zusätzlich die Methode TimeString genutzt um die aktuelle Uhrzeit zu bestimmen

If

Verzweigung im sonst sequentiellen Programmablauf in Abhängigkeit von einer Bedingung

Hier wird in Abhängigkeit von I-Eingang I1 der NachtZyklus (True, Taster gedrückt) und sonst (Else) der TagZyklus ausgeführt.

Select Case

Mehrfach-Verzweigung in Abhängigkeit von einen Ausdruck.

Visual Basic läßt im Select Case Konstrukt im Gegensatz zu anderen Programmiersprachen eine Vielzahl von Ausdrücken zu. Hier wird die aktuelle Zeit ausgewertet (Select Case TimeString). Case #6:00:00 AM# To #10:00:00 PM# ist für den Tagbetrieb zuständig. Case Else für den Rest. Mit einem If wäre es auch gegangen, das Case Konstrukt bietet in diesem Fall aber die Möglichkeit weitere Uhrzeiten zu berücksichtigen.

FishFace Themen

GetInput

Abfragen eines I-Einganges

ft.GetInput(Inp.I1) oder hier ft.GetInput(iNacht) liefert das Ergebnis True, wenn der I-Eingang geschlossen ist und False, wenn er offen ist.

FishKurs2 : Fußgängerampel

Modell

Großer Umbau : Die Lampen werden nicht mehr (zweipolig) an einen M-Ausgang angeschlossen, sondern mit einem Pol an einen O-Ausgang und dem zweiten an Masse. So schafft man Platz für weitere Lampen am Interface (geht so nur mit ROBO Interfaces).

Lampe oGruen an Ausgang O1
Lampe oGelb an Ausgang O2
Lampe oRot an Ausgang O3
Lampe oFussGruen an Ausgang O4
Lampe oFussrot an Ausgang O5
Taster iNacht an Eingang I1 (wie gehabt)

Lampenbezeichnung mit "o" am Anfang, um den Anschluß an den O-Ausgang anzuzeigen.

Das Programm

Hat sich eigentlich kaum verändert. Deswegen hier nur die Sub TagZyklus

```
Private Sub TagZyklus()
  ft.SetLamp(oGruen, Dir.Ein)          '     Grün, 3000
  ft.SetLamp(oFussRot, Dir.Ein)
  ft.Pause(3000)
  ft.SetLamp(oGruen, Dir.Aus)
  ft.SetLamp(oGelb, Dir.Ein)           '     Gelb, 1000
  ft.Pause(1000)
  ft.SetLamp(oGelb, Dir.Aus)
  ft.SetLamp(oRot, Dir.Ein)            '     Rot, 3500
  ft.SetLamp(oFussRot, Dir.Aus)
  ft.SetLamp(oFussGruen, Dir.Ein)
  ft.Pause(3500)
  ft.SetLamp(oFussGruen, Dir.Aus)
  ft.SetLamp(oFussRot, Dir.Ein)
  ft.SetLamp(oGelb, Dir.Ein)           '     Rot/Gelb, 1000
  ft.Pause(1000)
  ft.SetLamp(oRot, Dir.Aus)
  ft.SetLamp(oGelb, Dir.Aus)
End Sub
```

Im Tagzyklus werden jetzt mit SetLamp (statt SetMotor) die Lampen für die Autos und die Fußgänger in einem "Rutsch" geschaltet.

FishFace Themen

SetLamp

Schalten eines O-Auganges

ft.SetLamp(Out.O1, Dir.Ein)

heißt, Einschalten des O-Ausganges (ein Pin an O1, der zweite an Masse). Möglich ist auch
ft.SetLamp(Out.O1, Dir.Ein, Speed.Half)
um den Eingang z.B. mit halber Stärke einzuschalten.

FishKurs2b : Bedarfsschaltung, Nachtabschaltung

Modell

Wie gehabt. Der Taster an I1 wird jetzt aber als Taster für die Anforderung eines Fußgängerzyklus genutzt.

Das Programm

```
Private Sub FussZyklus()
  ... neuer Name, sonst wie TagZyklus()
End Sub

Private Sub NachtAbschaltung()
    ft.ClearMotors()
    ft.Pause(10000)
End Sub

Sub Main()
  ... OpenInterface ...
  ft.SetLamp(oGruen, Dir.Ein)
  ft.SetLamp(oFussRot, Dir.Ein)
  Do    ' --- Betriebsschleife Ampelanzeige ---
    If TimeString > #6:00:00 AM# And TimeString < #10:00:00 PM# Then
      If ft.GetInput(iFussWunsch) Then FussZyklus()
    Else
      NachtAbschaltung()
    End If
  Loop Until ft.Finish()              ' --- Ende durch ESC-Taste
  ... CloseInterface ...
End Sub
```

Die AutoAmpel steht standardmäßig auf Grün, bei Betätigen von iFussWunsch wird ein Fußgängerzyklus gestartet. Funktioniert nur in der Zeit von 6:00 bis 22:00 Uhr, Nachts wird die Ampel ganz abgeschaltet.

FishFace Themen

ClearMotors

Abschalten aller M-Ausgänge

ft.ClearMotors()

FishKurs3 : Kreuzung

Modell

AutoAmpel mit FußgängerAmpel und AutoAmpel für Querstraße.
Damit sind an einem Interface alle O-Ausgänge belegt.

Das Programm

```
Const cGruen = 1, cGelb = 2, cRot = 4
Const cFussGruen = 8, cFussRot = 16
Const cQuerGruen = 32, cQuerGelb = 64, cQuerRot = 128

Private Sub TagZyklus()
    ft.SetMotors(cGruen + cFussGruen + cQuerRot)
    ft.Pause(3500)
    ft.SetMotors(cGelb + cFussRot + cQuerGelb + cQuerRot)
    ft.Pause(1000)
    ft.SetMotors(cRot + cFussRot + cQuerGruen)
    ft.Pause(3500)
    ft.SetMotors(cGelb + cRot + cFussRot + cQuerGelb)
    ft.Pause(1000)
End Sub

Private Sub NachtZyklus()
    ft.SetMotors(cGelb + cQuerGelb)
    ft.Pause(666)
    ft.ClearMotors()
    ft.Pause(333)
End Sub

Sub Main()
  ... OpenInterface ...
  Do    ' --- Betriebsschleife Ampelanzeige ---
    If TimeString > #6:00:00 AM# And TimeString < #10:00:00 PM# Then
      TagZyklus()
    Else
      NachtZyklus()
    End If
  Loop Until ft.Finish()           ' --- Ende durch ESC-Taste
  ... CloseInterface ...
End Sub
```

Die Ampel schaltet eine Kreuzung in einem festen Ablauf, also keine Beeinflussung durch Fußgänger oder Autos. Nachts wird wieder geblinkt. Da jetzt 8 Lampen geschaltet werden müssen (eigentlich + 2 für eine zweite FußgängerAmpel), wird es mit dem einzelnen Schalten der Lampen durch SetLamp schon etwas mühsam. Es wird deswegen auf SetMotor**s** (gemeinsames Schalten aller Lampen) umgestellt.

Dazu werden neue Konstanten benötigt, die die Stellung der jeweiligen Lampe im Parameter von SetMotors berücksichtigen. SetMotors erwartet eine Integer Zahl, die als binär ausgewert wird :

Lampe cGrün ist an O1 geschaltet, sie erhält den IntegerWert 1, binär		0000 0001
Lampe cGelb ist an O2 geschaltet, sie erhältt den IntegerWert 2, binär		0000 0010
Lampe cRot	4	0000 0100
Lampe cFussGruen	8	0000 1000
Lampe cFussRot	16	0001 0000
Lampe cQuerGruen	32	0010 0000

Lampe cQuerGelb	64	0100 0000
Lampe cQuerRot	128	1000 0000
Die Werte können addiert werden :		
ft.SetMotors(cRot + cFussRot + cQuerGruen) AutoAmpel und zug. FußAmpel = rot, Quer = grün	52	0011 0100

FishFace Themen

SetMotors

Gemeinsames Setzen aller M- bzw. O-Ausgänge

ft.SetMotors(4)

setzt z.B. die rote AutoAmpelLampe, wenn ein Motor an M2 gesteckt ist, wird er auf links geschaltet. In diesem Fall werden 2 bit ausgewertet (bit 3 und 4, von rechts gezählt).

FishKurs3b : Kreuzung mit Bedarfssteuerung

Modell

Wie gehabt, der Taster iFussWunsch tritt wieder in Erscheinung.

Das Programm

```
Private TagListe() As Integer = {cGruen + cFussRot + cQuerRot, _
                                 cGruen + cFussRot + cQuerRot, _
                                 cGruen + cFussRot + cQuerRot, _
                                 cGelb  + cFussRot + cQuerGelb+
                                 cQuerRot, _
                                 cRot   + cFussRot + cQuerGruen, _
                                 cGelb + cRot + cFussRot+QuerGelb, _
                                 cGelb + cRot + cFussRot+QuerGelb, _
                                 cGelb + cRot + cFussRot + QuerGelb}

Private FussListe() As Integer = {cGruen + cFussGruen + cQuerRot, _
                .....}

Private Sub Zyklus(ByVal Liste() As Integer)
  For i As Integer = 0 To Liste.Length - 1
    ft.SetMotors(Liste(i))
    If ft.GetInput(iFussWunsch) And Not FussWunsch Then
      Console.WriteLine("Signal     kommt")
      FussWunsch = True
    End If
    ft.Pause(1000)
  Next
End Sub

Private Sub NachtZyklus()
  ft.SetMotors(cGelb + cQuerGelb)
  ft.Pause(666)
  ft.ClearMotors()
  ft.Pause(333)
End Sub

Sub Main()
  ... OpenInterface ...
  Do               ' --- Betriebsschleife Ampelanzeige ---
    If TimeString > #6:00:00 AM# And TimeString < #11:00:00 PM# Then
      If FussWunsch Then
        Console.WriteLine("FussZyklus laeuft")
        Zyklus(FussListe)
        FussWunsch = False
      Else
        Console.WriteLine("TagZyklus  laeuft")
        Zyklus(TagListe)
      End If
    Else
      NachtZyklus()
    End If
  Loop Until ft.Finish()       ' --- Ende durch ESC-Taste
  ... CloseInterface ...
End Sub
```

Hier wird die Kreuzung standardmäßig in einem "Automodus" betrieben, d.h. die Fußgängerampel steht auf Dauerrot. Dafür gibt es wieder einen Taster zur Anforderung einer Fußgängerphase. Das Problem dabei ist den iFussWunsch zu erkennen, die Ampel ist ja ständig mit der Autophase beschäftigt. Da hilft nur, sich dazwischenschummeln. Damit das mit vertretbaren Aufwand geht, werden die Schaltbefehle in einer Tabelle untergebracht. Dies Tabelle wird dann in einem allgemeinen Unterprogramm in einem festen Takt "abgearbeitet", wenns mit einer Schaltung länger dauert, wird dann die gleiche Schaltung entsprechend mehrmals in die Tabelle eingetragen.

Wenn man nach jedem Takt eine Abfrage von iFussWunsch einbaut, geht's mit der Anforderung nach einer Fußgängerphase sehr schön. Damit wildes Drücken von iFussWunsch nicht zu ständiger Fußgängerei führt, wird die Variable FussWunsch als Sperre eingebaut. Sie nimmt die Werte False (kein iFussWunsch) und True (iFussWunsch möglich) an.

Der Nachtzyklus bleibt wie gehabt.

VB2005 Themen

Variable

Benanntes Feld, das durch das Programm veränderbare Werte aufnehmen kann.

```
Private FussWunsch As Boolean = False
```

Die Variable FussWunsch ist vom Typ Boolean und kann die Wahrheitswerte True und False annehmen. Beim Start des Programm hat sie den Wert False. Die Platzierung vor allen Sub's macht sie zu einer für den Modul von allen Sub's erreichbaren Variablen (globalen). Da sie außerdem mit Private gekennzeichnet wurde, ist sie von Sub's außerhalb von Module FishKurs3b nicht erreichbar (Dafür müßte sie mit Public gekennzeichnet werden).

Die Konstanten z.B. Const cGruen = 1 unterscheiden sich von Variablen dadurch, das sie im Programmablauf nicht verändert werden können. Da sie hier mit keinem Gültigkeitsbereich gekennzeichnet wurde gilt Private.

Array (Tabelle)

Zusammenfassung mehrerer gleichartiger Variablen in einer Liste.

```
Private TagListe() As Integer = {....}
```

Die Schaltbefehle für den Automodus werden in einer Liste mit Variablen vom Typ Integer zusammen gefaßt. Sie werden hier gleichzeitig mit Werten belegt, dabei ergibt sich aus der Anzahl der Werte auch die Länge der Liste.

Private TagListe(8) As Integer

TagListe enthält 9 Integerwerte, die von 0 .. 8 indiziert werden (angesprochen werden können).

Parameter

Werte, die einem Unterprogramm (Sub) übergeben werden können

```
Zyklus(TagListe)
```

Der Sub Zyklus zur Steuerung eines AmpelZyklus wird die Liste für den AutoModus übergeben.

For .. Next Schleife

Wiederholung der Befehl in der Klammer For .. Next

```
For i As Integer = 0 To Liste.Length-1

....
```

```
Next
```

Die Befehle zum Abarbeiten der Steuerungsliste. Dabei wird mit der Laufvariablen i auf die einzelnen Elemente der Liste zugegriffen. i wird hier für den ausschließlichen Gebrauch in der Schleife deklariert.

FishKurs3t : Thread

Modell

Genau wie gehabt

Das Programm

```
Imports System.Threading
Imports FishFace40

Module FishKurs3t
  Private ft As New FishFace()
  Private fussWunschThread As Thread
  Private FussWunsch As Boolean = False
.....

Private Sub FussWunschTaste()
  Do
    If ft.GetInput(iFussWunsch) And Not FussWunsch Then
      Console.WriteLine("Signal     kommt")
      FussWunsch = True
      ft.Pause(1000)
    End If
  Loop Until ft.Finish()
End Sub

Sub Main()
  ... OpenInterface ...
  fussWunschThread = New Thread(AddressOf FussWunschTaste)
  fussWunschThread.Start()
  Do                    ' --- Betriebsschleife Ampelanzeige ---
    If TimeString > #6:00:00 AM# And TimeString < #11:00:00 PM# Then
      If FussWunsch Then
        Console.WriteLine("FussZyklus laeuft")
        Zyklus(FussListe)
        FussWunsch = False
      Else
        Console.WriteLine("TagZyklus  laeuft")
        Zyklus(TagListe)
      End If
    Else
      NachtZyklus()
    End If
  Loop Until ft.Finish()     ' --- Ende durch ESC-Taste
  fussWunschThread.Join()
  ... CloseInterface ...
End Sub
```

Alternative zur Lösung 3b. Hier wird zur Tastenabfrage iFussWunsch ein eigener Thread eingesetzt. Vorteil : Die Tastenabfrage läuft unabhängig vom restlichen Programm und muß nicht irgendwo dazwischen "gewurstelt" werden. Nachteil : Es wird deutlich komplizierter.

Gegenüber 3b wird die iFussWunsch-Abfrage aus Zyklus() herausgenommen und in einer eigenen Sub FussWunschTaste() untergebracht. Der Programm-Aufbau bleibt sonst gleich. Aber es wird ein Thread (fussWunschThread - "Programm-Faden") eingeführt in dem die Tastenabfrage läuft. Die entsprechenden Programmstellen wurde fett markiert :

```
fussWunschThread = new Thread(AddressOf FussWunschTaste)
```

Instanzierung des weiter oben deklarierten Threads, dabei wird der Instanz die Adresse (der Name) der im Thread auszuführenden Sub mitgeteilt, der Thread selber wird mit

`fussWunschThread.Start()` gestartet.

Am Ende von Sub Main muß noch mit `fussWunschThread.Join()` auf das Thread-Ende gewartet werden. Das Ende wird durch die ESC-Taste ausgelöst, die in ft.Finish() ausgewertet wird (in Sub Main() und Sub FussWunschTaste).

VB2005 Themen

Threading

Ausführen eines Programmteils unabhängig vom Hauptteil des Programms

```
Imports System.Threading
```

Erforderlich um die Threading.Methoden einfach ansprechen zu können. Zu den Thread.Methoden siehe oben.

Windows.Forms Programme

FishKurs 2W : Fußgängerampel

Modell

wie Konsole FishKurs2

Das Programm

Mit :

- lblStatus : Anzeigefeld
– cmdAction : Starten von Actions
- cmdEnde : Beenden von Actions / Programm

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
  Try
    ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    lblStatus.Text = "Ampelschleife läuft"
    cmdEnde.Text = "HALT"
    cmdAction.Enabled = False
    Do              ' --- Betriebsschleife Ampelanzeige ---
      Select Case TimeString
        Case #6:00:00 AM# To #10:00:00 PM#
          If ft.GetInput(iNacht) Then NachtZyklus() Else TagZyklus()
       Case Else
         NachtZyklus()
      End Select
    Loop Until ft.Finish() ' --- Ende ESC-Taste oder ft.NotHalt
    lblStatus.Text = "Ampelschleife beendet"
```

```
    cmdEnde.Text = "ENDE"
    cmdAction.Enabled = True
    ft.CloseInterface()
  Catch eft As FishFaceException
    lblStatus.Text = eft.Message
  End Try
End Sub

Private Sub cmdEnde_Click(ByVal sender As System.Object, _
                  ByVal e As System.EventArgs) Handles cmdEnde.Click
  If cmdEnde.Text = "HALT" Then ft.NotHalt = True Else Me.Close()
End Sub

Private Sub FishKurs2W_FormClosing(ByVal sender As System.Object, _
       ByVal e As System.Windows.Forms.FormClosingEventArgs) _
       Handles MyBase.FormClosing
  If cmdEnde.Text = "HALT" Then e.Cancel = True
End Sub
```

Fußgängerampel mit Nachtzyklus wie FishKurs2 (Console). Die Steuerung erfolgt jetzt über Windows.Buttons, die Anzeige auf einem Windows.Label.

VB2005 Themen

Datei | Neues Projekt

Anlegen einer Windows Anwendung.

1. Datei | Neues Projekt | Windows Anwendung
2. Form1 in FishKurs2W umbenennen
3. Verweis auf FishFace2005.DLL hinzufügen
   Menü Projekt | Verweise hinzufügen | Durchsuchen
4. Imports FishFace40 als erste Source Zeile in Fishkurs2 einfügen
5. Erste Zeile in Public Class FishKurs2W : `Private ft As New FishFace()`
6. Form Eigenschaft Text = FishKurs2, Font = Tahoma; 8,25pt
7. Buttons cmdAction (Text Action) und cmdEnde (Text Abbrechen) einfügen
8. Label lblStatus einfügen (AutoSize = False)
9. Form Eigenschaft AcceptButton = cmdAction und CancelButton = cmdEnde

Ablaufsteuerung

Button-Clicks Kontrollieren.

Bei längerlaufenden Routinen, die durch einen Button-Click ausgelöst werden, ist es problematisch, wenn der Button vor Ende der Routine erneut gedrückt wird. Ebenso kann es zu Problemen führen, wenn anderere Buttons oder das Windows x (rechts oben) gedrückt wird. deswegen muß hier zeitweise verriegelt werden :

1. Nach Start über den Action-Button, cmdEnde.Text = "HALT",
   cmdAction.Enabled = False.
2. Nach Ende der Action-Routine : cmdEnde.Text = "ENDE" und
   cmdAction.Enabled = True
3. Bei einem Click auf cmdEnde wird der .Text ausgewertet. Bei Text = "HALT" wird lediglich ft.NotHalt = True gesetzt (eine Ende-Anforderung), das Programm wird nicht beendet. ft.Finish und ft.Pause ihrerseits werten NotHalt dann aus. Pause bricht bei NotHalt = True ab, Finish liefert dann True zurück, also Do .. Loop Ende.

4. Ein Click auf das Windows x (rechts oben) führt bei "HALT" zu einem Abweisen des Programmendes (Cancel = True, EreignisRoutine).

FishKurs 2bW : FußAmpel mit Bedarfsanforderung

Modell

Wie gehabt, I1 ist jetzt wieder Taster für FussWunsch.
Entspricht dem Programm FishKurs2b.

Das Programm

Hinzugekommen ist der Button cmdFussWunsch zur Anforderung einer Fußgängerphase und die maskierten Textfelder mskVonZeit und mskBisZeit zur Eingabe der Zeiten für die Nachtabschaltung.

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
  Dim VonZeit As DateTime = mskVonZeit.Text
  Dim BisZeit As DateTime = mskBisZeit.Text
  Try
    ft.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    lblStatus.Text = "Ampelschleife läuft"
    cmdEnde.Text = "HALT"
    cmdAction.Enabled = False
    ft.SetLamp(oGruen, Dir.Ein)
    ft.SetLamp(oFussRot, Dir.Ein)
    Do                    ' --- Betriebsschleife Ampelanzeige ---
      Select Case TimeString
      Case VonZeit To BisZeit
        If ft.GetInput(iFussWunsch) Or FussWunsch Then FussZyklus()
      Case Else
        NachtAbschaltung()
      End Select
    Loop Until ft.Finish()     ' --- Ende durch ESC oder ft.NotHalt
    lblStatus.Text = "Ampelschleife beendet"
    cmdEnde.Text = "ENDE"
    cmdAction.Enabled = True
```

```
    ft.CloseInterface()
  Catch eft As FishFaceException
    lblStatus.Text = eft.Message
  End Try
End Sub

Private Sub cmdFussWunsch_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles cmdFussWunsch.Click
  FussWunsch = True
End Sub
```

Die Autoampel steht standardmäßig auf Grün, bei Betätigen des Tasters iFussWunsch oder des Buttons cmdFussWunsch wird eine Fußgängerphase geschaltet. Außerhalb von VonZeit – BisZeit wird die Ampel ganz abgeschaltet.

VB2005 Themen

MaskedTextBox

Geprüfte Eingaben in ein Textfeld.

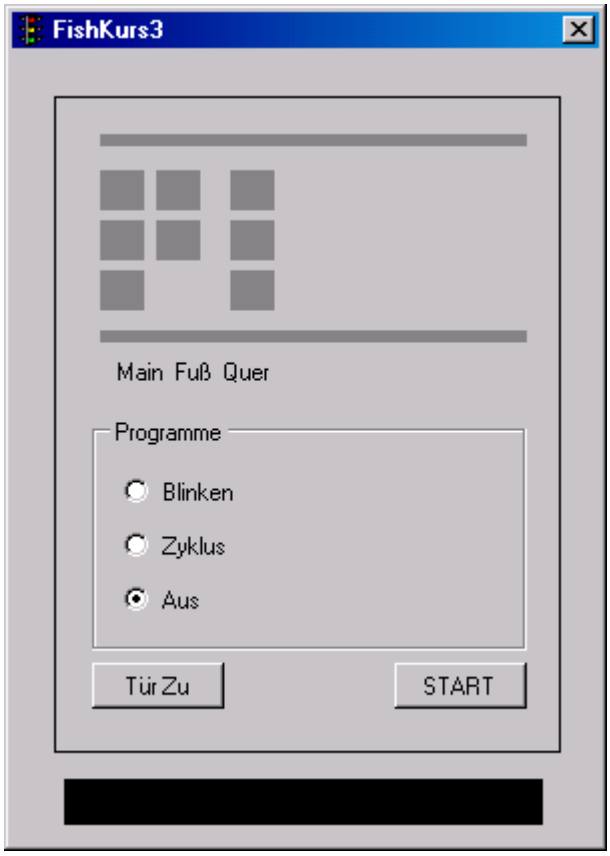
mskVonZeit, mskBisZeit ist nur die Eingabe von Stunden und Minuten zulässig. Das geschieht durch Vorgabe der Maske Zeit in der zugehörenden Eigenschaft Mask.

FishKurs3W : Kreuzung

Modell

Entspricht FishKurs3

Das Programm



Wie FishKurs3 : Kreuzung mit Fußgängerampel

Das Programm wird jetzt in einem schicken Schaltschrank untergebracht. Auf einer Anzeigetafel werden die Schaltzustände aller Ampel anzeigt. Es können auch einzelne Programme geschaltet werden.

VB2005 Themen

RadioButton

Zur Auswahl einer Alternative

RadioButtons werden auf einem Container (z.B. einer GroupBox) in Form einer Auswahlliste plaziert. Es kann jeweils nur ein RadioButton auf dem Container markiert sein. Das markieren wird üblicherweise über die EventRoutine CheckChanged vorgenommen. Das Verwenden nur einer CheckedChanged Routine ist sinnvoll, erfordert aber eine etwas mühsame Auswertung der Herkunft. Siehe unten.

Eine EventRoutine für mehrere Events

Oft erfordern die EventRoutinen für mehrere Controls eine gleiche Prozedur, es ist aber trotzdem meist notwendig, das auslösende Control zu kennen.

```
Private Sub rdoProgCheckedChanged(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles rdoAus.CheckedChanged, _
      rboZyklus.CheckedChanged, rboBlinken.CheckedChanged
  Dim R As RadioButton = sender
  ProgName = R.Tag
  ft.NotHalt = True
  cmdAction.Enabled = True
End Sub
```

Hier soll in cmdActionClick eine spezifische Methode aufgerufen werden. Ein Hinweis dazu wird über die Eigenschaft Tag des jeweiligen Controls an die Variable ProgName übergeben. Hinweis : Man hätte in diesem Fall auch die Eigenschaft Text nutzen können, wie schon bei den Button vorher angewendet, Tag ist manchmal etwas flexibler, manches ist auch schlicht Gewohnheit.

Windows.Forms

Die Haupt- und einzige Form des Programm soll einen Schaltschrankk, wie er an Ecken von Kreuzungen zu finden ist, darstellen, auf dem Bild oben fehlt noch die Tür. Die wird hier durch ein Panel mit Button (zum Öffnen/Schließen) dargestellt. Durch ein schlichtes pnlTuer.Visible = False/True wird das bewerkstelligt.

Damit sie beim Umgang mit den Schaltschrankinnereien nicht stört, wird die Form vergrößert und pnlTuer neben pnlSchrank untergebracht. In Ereignis Load der Form wird das dann zurechtgerückt.

Ablaufsteuerung

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
  Dim Prg As String = ProgName
  cmdAction.Enabled = False
  ft.NotHalt = False
  Do
    Select Case Prg
      Case "Blinken"
        Blinken()
      Case "Zyklus"
        If TimeString > #6:00:00 AM# And _
           TimeString < #10:00:00 PM# Then
          TagZyklus()
        Else
          Blinken()
        End If
      Case "Aus"
        Abschalten()
        Exit Do
    End Select
  Loop Until ft.Finish()
End Sub
```

Der Ablauf wird über cmdAction (START) gesteuert, es wird die über die RadioButtons vorgewählte Action ausgeführt. Über das rdoProgCheckedChanged wird das zu betreibende Programm vorgewählt (ProgName) und bei Click auf START in Prg übertragen und schließlich im Do .. Loop nach Ende eines Case dann als neuer Case ausgeführt.

IIf

Wertzuweisung in Abhängigkeit von einer Bedingung

```
lblRot.BackColor = _
        IIF((ft.Outputs And cRot) < 0, Color.Red, Color.Gray
```

Dem Label lblRot wird die Farbe Rot zugewiesen, wenn die And Operation einen Wert > 0 ergibt, d.h. wenn das dritte bit von rechts in ft.Ouputs gesetzt ist. Andernfalls wird die Farbe Grau zugewiesen.

Timer : laufende Anzeige der Ampelfarben

```
Private Sub tmrAnzeige_Tick(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles tmrAnzeige.Tick
  Dim LS As Integer = ft.Outputs
  lblRot.BackColor = IIf((LS And cRot) > 0, Color.Red, Color.Gray)
  lblGelb.BackColor = IIf((LS And cGelb) > 0, _
                                          Color.Yellow, Color.Gray)
  lblGruen.BackColor = IIf((LS And cGruen) > 0, _
                                           Color.Green, Color.Gray)
  lblFussRot.BackColor = IIf((LS And cFussRot) > 0, _
                                             Color.Red, Color.Gray)
  lblFussGruen.BackColor = IIf((LS And cFussGruen) > 0, _
                                           Color.Green, Color.Gray)
  lblQuerRot.BackColor = IIf((LS And cQuerRot) > 0, _
                                             Color.Red, Color.Gray)
  lblQuerGelb.BackColor = IIf((LS And cQuerGelb) > 0, _
                                          Color.Yellow, Color.Gray)
  lblQuerGruen.BackColor = IIf((LS And cQuerGruen) > 0, _
                                           Color.Green, Color.Gray)
End Sub
```

Der Timer tmrAnzeige wird beim Start des Programms im Load-Ereignis gestartet und bei Ende des Programm (Sub Abschalten) wieder angehalten. In der zugehörenden Ereignis-Routine wird zunächst der Status aller O-Ausgänge ermittelt und dann pro Ampel ermittelt, ob sie eingeschaltet ist (Zur Maske siehe FishKurs3).

MsgBox

```
If MsgBox("Soll das Programm ganz abgeschaltet werden?", _
              MsgBoxStyle.YesNo, "FishKurs") = MsgBoxResult.Yes Then
```

Ausgabe einer kurzen Meldung.

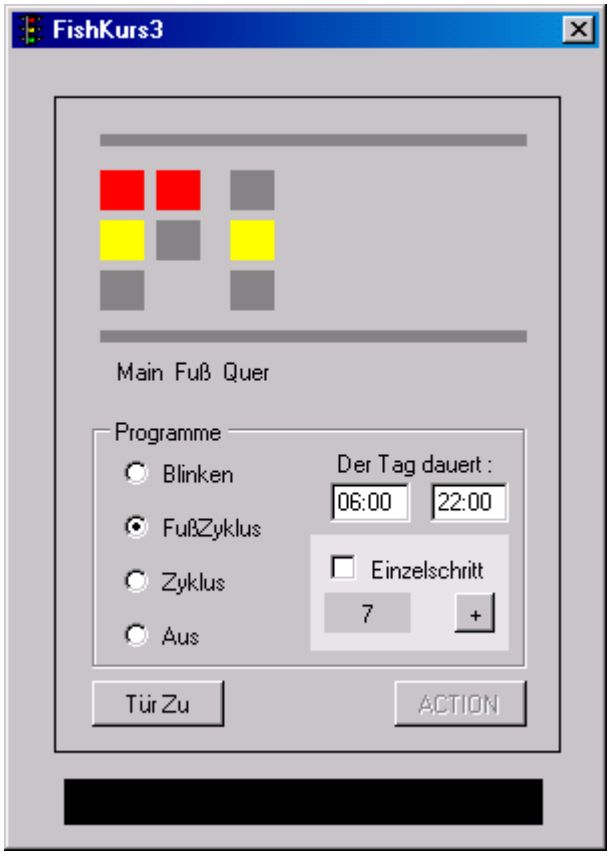
Die Meldung kann lediglich eine Quittung erfordern oder bietet Alternativen zur Beantwortung an.

FishKurs3bW : Kreuzung mit Bedarfssteuerung

Modell

wie gehabt, der Taster iFussWunsch taucht wieder auf.

Das Programm



entspricht weitgehend dem Konsolprogramm FishKurs3b und baut auf FishKurs3W auf. Hinzu gekommen ist die schon bekannte Anzeige der TagZeit, neu ist die Taktanzeige verbunden mit der Möglichkeit, einen Zyklus im Einzelschritt durchzugehen.

VB2005 Themen

Zugang über Paßwort

```
Private Sub cmdOeffnen_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles cmdOeffnen.Click
    If txtPass.Text = "FishKurs" Then pnlTuer.Visible = False
End Sub
```

Ist eigentlich ganz einfach : Die TextBox txtPass bekommt einen PasswordChar verpaßt und die getätigte Eingabe wird vor dem Türöffnen in cmdOeffnen abgefragt.

Exit Do

Vorzeitiges Verlassen eines Do .. Loop

```
  Case "Aus"
    Abschalten()
    Exit Do
  End Select
Loop Until ..
```

Der umgebende Do .. Loop und das Select Construct werden verlassen, weiter nach Loop Until ..

Ablaufsteuerung

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
  Dim Prg As String = ProgName
  Dim VonZeit As DateTime = mskVonZeit.Text
  Dim BisZeit As DateTime = mskBisZeit.Text
  cmdAction.Enabled = False
  ft.NotHalt = False
  Do
    Select Case Prg
      Case "Blinken"
        Blinken()
      Case "FußZyklus"
        Zyklus(FussListe)
      Case "Zyklus"
        If TimeString > VonZeit And TimeString < BisZeit Then
          If FussWunsch Then
            Zyklus(FussListe)
            FussWunsch = False
          Else
            Zyklus(TagListe)
          End If
        Else
          Blinken()
        End If
      Case "Aus"
        Abschalten()
        Exit Do
    End Select
  Loop Until ft.Finish()
End Sub
```

Ist weitgehend vertraut. Hier wird die TagZeit den Feldern mskVonZeit, mskBisZeit entnommen. Innerhalb des normalen Zyklus kann wieder durch den Taster iFussWunsch eine Fußgängerphase gestartet werden. Die fällige Abfrage geschieht hier wieder indirekt um "Dauerdrückern" das Leben schwer zu machen. Die Abfrage des iFussWunsch geschieht in der Ereignis-Routine des Anzeige-Timers. Dort wird die Variable FussWunsch auf True gesetzt. Die FussWunsch-Abfrage geschieht dann hier im Do .. Loop. Dort wird FussWunsch auch nach jeder Fußphase zurückgesetzt.

Einzelschritt

```
Private Sub Zyklus(ByVal Liste() As Integer)
  For i As Integer = 0 To Liste.Length - 1
    lblSchrittNr.Text = i + 1
    ft.SetMotors(Liste(i))
    If chkEinzelSchritt.Checked Then
      NextStep = False
      Do Until NextStep Or ft.NotHalt
        ft.Pause(100)
      Loop
    Else
      ft.Pause(1000)
    End If
  Next
End Sub
```

Bei der sequentiellen Abarbeitung der TaktListe wird zuerst geschaltet und auf chkEinzelSchritt.Checked abgefragt und im zutreffenden Fall wird NextStep zurückgesetzt und auf ein erneutes NextStep = True gewartet. NextStep selber wird über den +Button (cmdNextStep) gesetzt. Die zusätzliche Abfrage auf NotHalt beendet das Warten, wenn die ESC-Taste gedrückt wurde.

Ausblick

Threads

FishFace unterstützt durch internes DoEvent die Unterbrechbarkeit eines Windows-Haupthreads. Für reine Anzeige- oder ButtonClick-Operationen sind Threads deswegen nicht erforderlich. Für regelmäßige Anzeigen, z.B. des Interfacestatus, kann man auf Time-Event-Routinen ausweichen.

Bei größeren Modellen ist es aber oft sinnvoll, einzelne Komponenten in eigenen Threads laufen zu lassen (siehe FishKurs3t). Solange sie im Hintergrund laufen, ist das relative leicht machbar. Man muß da ggf. die gemeinsame Resource Interface im Auge behalten. Der Zugriff auf Windows.Forms ist allerdings ab VB2005 etwas komplexer geworden.

Da der Zugriff auf Windows.Controls aus fremden Threads ab VB2005 nicht mehr so einfach machbar. Wenn nur StatusTexte oder Zählerstände in Labels oder TextBoxen auszugeben sind, ist es vertretbar das über ein :

`Label.CheckForIllegalCrossThreadCalls = False` (entspr. für TextBox ..)

Wieder möglich. Ansonsten muß man auf die Eigenschaft Invoke .. des entsprechenden Controls zugreifen. Das ist dann schon etwas aufwendiger ...

Interfaces mit Extensions

Das ROBO Interface kann mit bis zu drei ROBO I/O Extensions betrieben werden. Dazu werden die Extension-Boxen über mitgelieferte Flachbandkabel kann, ausgehend von der entsprechenden Steckdose des ROBO Interfaces, eine Extension nach der anderen angeschlossen werden (Extension Ext. IN – Ext. OUT). Im Programm selber sind keine weiteren Vorkehrungen zu treffen, man muß wissen, ob 4 – 8 – 12 – 16 zu Verfügung stehen. Sie sind schlicht durchnummeriert. Analog die I-Eingänge. Bei den weiteren Eingänge des Interfaces bzw. der Extensions muß man etwas genauer hinsehen : Entsprechende Enums nutzen.

Mehrere Interfaces

Werden mehrere ROBO Interfaces bzw. Extensions eigenständig an USB bzw. Interfaces auch an COM angeschlossen, muß pro Interface ein eigenes Objekt instanziert werden. ROBO Interface selber können dann wieder bis zu drei Extensions haben. Beim Ansprechen der einzelnen Interface-Objekt fängt die Zählung der Ein- und Ausgänge dann also wieder bei 1 an.