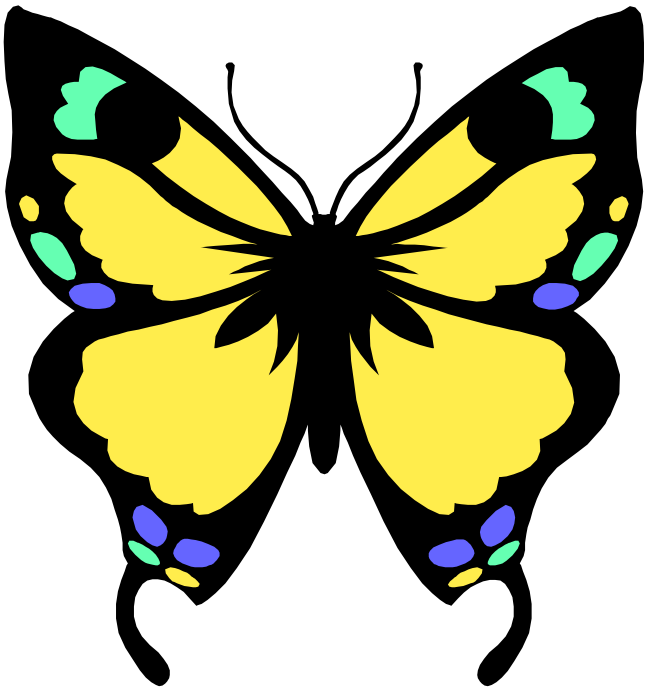
Robo Pro Light -> VB 2005 / 2010

ROBO LT Beginner

Ulrich Müller



Inhaltsverzeichnis

Übersichten

Allgemeines

Die Programme können unter Vista 32bit und Windows 7 32/64bit betrieben werden. Bei 64bit-Systemen ist der Einsatz von Visual Basic 2010 Express (oder VB 2005 .. Prof.) erforderlich, da sonst der Umgang mit den 32bit-DLLs nicht klappt.

Buch

[Born] Born : Visual Basic 2008 easy (wenn verfügbar auch 2010)
ISBN 978-3-8272-4368-3 Verlag Markt + Technik

Im laufenden Text wird wiederholt auf Buchstellen hingewiesen, die verwendete Visual Basic Statements erklären.

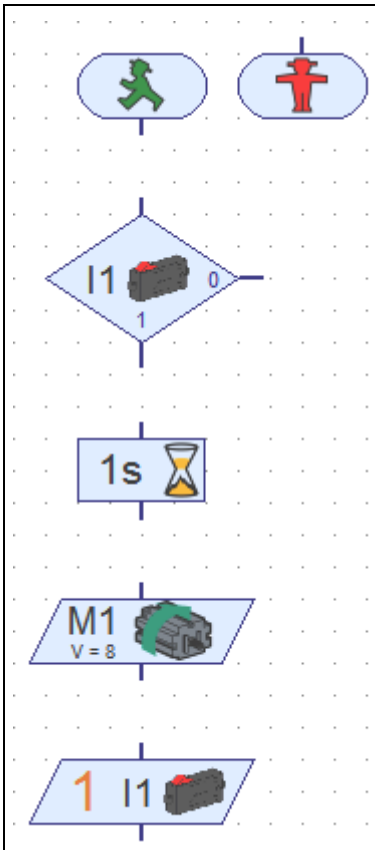
Installation

Das mit dem Kasten ROBO LT Beginner Lab gelieferte Robo Pro Light einschl. Treiber installieren. Achtung : Soll parallel dazu Robo Pro (standard) betrieben werden, so ist dessen Version 3.0 erforderlich.

Visual Basic ab 2005 installieren

www.ftcomputing.de/roboLightVB.htm : Dort roboLightVB.ZIP. Nach dessen Readme.txt installieren.

Gegenüberstellung ROBO Pro Light - VB Befehle

	`Sub Main() --- End Sub` bzw. `Private Sub`
	
	`If fl.GetInput(Inp.I1) Then`
	' --- möglich : Inp.I1, I2, I3 ---
	----- 	
	`End If`
	
	`fl.Pause(1000)` ' --- Zeitangabe in Millisekunden
	
	`fl.SetMotor(Out.M1, Dir.Rechts, Speed.Full)`
	' --- möglich : Out.M1, M2, Dir.Ein (Lampen)
	' Dir.Links, Recht, Aus. Speed.L1 … Full, Half
	' Bei Speed.Full kann es auch entfallen
	`fl.WaitForInput(Inp.I1, True)`
	' --- True kann auch entfallen, alternativ False

Programmrahmen

Hinweis : Der LT Controller muß vor dem Start eines Programmes am Rechner angeschlossen sein.

Konsolprogramme

Die Mehrzahl der Beispielprogramme sind "Konsolenanwendungen" und werden über die Startseite und "Erstellen Projekt" durch Klick auf das entsprechende Symbol erstellt. Man sollte dabei schon den gewünschten Projektnamen angeben, da eine nachträgliche Änderung schwierig sein kann.

Konsolanwendungen mit FishFace2005.DLL haben hier immer den gleichen Aufbau :

```
Imports FishFace40
Module Karussel1
   Dim fl As New FishFace()

   Sub Main()
      Console.WriteLine("--- Karussel starten : I1-Taster ---")
      fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)

--- Nutzfunktionen ---

      fl.CloseInterface()
      Console.WriteLine("--- Schließen : Return-Taste ---")
      Console.ReadLine()
   End Sub
End Module
```

`Imports`: Der Namespace FishFace40 soll verwendet werden. Zusätzlich erforderlich (über My Projects) der Verweis auf die Assembly FishFace2005.DLL

`Module`: Rahmen für das Konsolprogramm (Ausgaben erfolgen in einem DOS-Fenster (Eingabeaufforderung der Startleiste)

`Dim`: Instanz der Klasse FishFace für den Zugriff auf den LT Controller (über die Objekt-Variable `fl`)

`Sub` Main() - `End Sub`: Das Hauptprogramm mit dem gestartet wird.

`Console.WriteLine` : Kommentierende Texte, die im DOS-Fenster angezeigt werden.
`Console.ReadLine()` hält das DOS-Fenster bei Programmende offen (und lesbar).

`OpenInterface`(..) und `CloseInterface` : Herstellen und Beenden einer Verbindung zum LT Controller an USB.

Windowsprogramme

Die hier gezeigten Windowsprogramme haben einen einfachen Aufbau. Sie bestehen aus einer Form mit dem START-Button (Name cmdAction) und der zugehörenden Click-Routine (cmdAction_Click) sowie einem Label (lblStatus) zur Statusanzeige und einem Label (lblHinweis) zur zusätzlichen Anzeige von Hinweisen. Ein Windowsprogramm wird - wie ein Konsolprogramm - über die Startseite und "Erstellen Projekt" durch Klick auf die Vorlage "Windows-Anwendung" erstellt. Auch hier : Gleich den gewünschten Projektnamen angeben.

```
Imports FishFace40
Public Class Karussel3Win

  Private Sub cmdAction_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles cmdAction.Click
    Dim fl As New FishFace()
    fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    lblStatus.Text = "Karussel gestartet"
    cmdAction.Enabled = False

    --- Nutzfunktionen ---

    fl.CloseInterface()
    lblStatus.Text = "Das war's"
    cmdAction.Enabled = True
  End Sub
End Class
```

`Imports` : Wie Console

`Class` : Hier wird anstelle eines Moduls eine Klasse erstellt. Genaugenommen eine partielle Klasse, in Teil 2 sind die Statements für die Windows Form enthalten.

`Sub` : Die Routine, die bei Click auf den zugehörende Button angesprungen wird.

`Dim, OpenInterface, CloseInterface` : wie gehabt.

`lblStatus.Text` : Anzeigetexte für das Label lblStatus anstelle von Console.WriteLine

`cmdAction.Enabled` : Während der Ausführung der Click-Routine wird der Button gesperrt.

Modell-Programme

Karussel

Betrieb eines Karussels über einen Motor an M1 und einen Starttaster an I1.

1 : Karusselfahren für 10 Sekunden, Start über I1

2 : Wiederholmöglichkeit durch Endlosschleife

3 : Zusätzlich Drehen in der Gegenrichtung

Version 1 :

```
fl.WaitForInput(Inp.I1)
fl.SetMotor(Out.M1, Dir.Rechts)
fl.Pause(10000)
fl.SetMotor(Out.M1, Dir.Aus)
```

Warten auf Starttaster, Einschalten des Motors, 10 Sekunden warten, Ausschalten des Motors. Diese Befehle werden anstelle des Textes -Nutzfunktionen- in den Programmrahmen eingefügt.

Version 2 :

```
Do
  Console.WriteLine("Neue Runde : I1-Taster")
  fl.WaitForInput(Inp.I1)
  fl.SetMotor(Out.M1, Dir.Rechts)
  fl.Pause(10000)
  fl.SetMotor(Out.M1, Dir.Aus)
Loop Until fl.Finish()
```

Wie 1, jetzt aber in einer "endlosen" Do ----- Loop Schleife, die über die ESC-Taste der Tastatur abgebrochen werden kann (Führe die Befehle zwischen Do und Loop aus bis Finish() feststellt, dass eine ESC-Taste gedrückt wurde).

Do --- Loop [Born] Seite 91

Version 3 :

```
    Do
      -------
      fl.SetMotor(Out.M1, Dir.Links)
      fl.Pause(10000)
      fl.SetMotor(Out.M1, Dir.Aus)
    Loop Until fl.Finish()
```

Wie 2, hinzugekommen sind 10 Sekunden in Gegenrichtung.

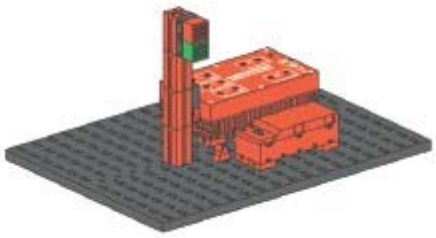
Windows-Version

3Win wie Console Version 3. Der Ablauf erfolgt in der Click-Routine des START-Buttons (cmdAction). Die Ausgabe erfolgen jetzt in das Label lblStatus.

Auf die Schleife über I1 wurde verzichtet : START kann bei Bedarf wiederholt betätigt werden.

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles cmdAction.Click
  Dim fl As New FishFace()
  fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
  lblStatus.Text = "Karussel gestartet"
  cmdAction.Enabled = False

  fl.SetMotor(Out.M1, Dir.Rechts)
  fl.Pause(10000)
  fl.SetMotor(Out.M1, Dir.Aus)
  fl.Pause(1000)

  fl.SetMotor(Out.M1, Dir.Links)
  fl.Pause(10000)
  fl.SetMotor(Out.M1, Dir.Aus)

  fl.CloseInterface()
  lblStatus.Text = "Das war's"
  cmdAction.Enabled = True
End Sub
```

Fußgängerampel



Betrieb einer Fußgängerampel mit Lampen Rot an M1 und Grün an M2 sowie einem Anforderungstaster an I1 in einer Endlosschleife

1 : Rot einschalten, Fußphase anfordern,
5 Sekunden warten. 10 Sekunden grün

2 : Zusätzlich Grünblinken am Ende der Fußphase

Version 1 :

```
    fl.SetMotor(Out.M1, Dir.Ein)
    Do
      If fl.GetInput(Inp.I1) Then
        Console.WriteLine("Signal kommt")
        fl.Pause(5000)
        fl.SetMotor(Out.M1, Dir.Aus)
        fl.SetMotor(Out.M2, Dir.Ein)
        fl.Pause(10000)
        fl.SetMotor(Out.M1, Dir.Ein)
        fl.SetMotor(Out.M2, Dir.Aus)
      End If
      Loop Until fl.Finish()
```

Einschalten Rot, Endlosschleife mit warten auf Anforderung Fußphase (If fl.GetInput..), bei Erkennen : Nachricht, Warten 5 Sekunden, Rot Aus, Grün An für 10 Sekunden, dann wieder Rot an, Grün aus.

If … Then [Born] Seite 96 ff.

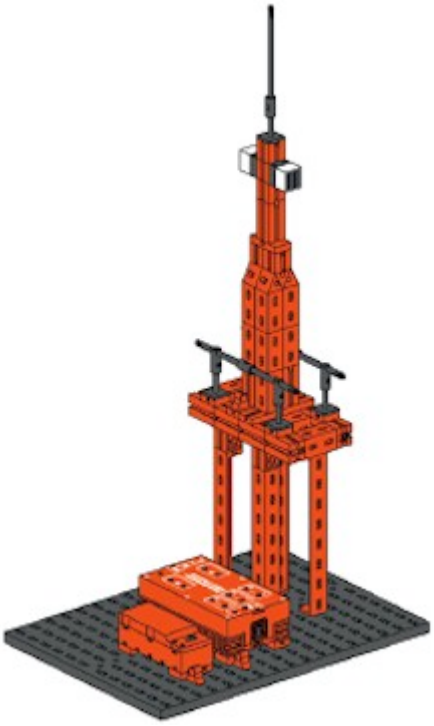
Version 2 :

```
  Dim   fl          As New FishFace()
  Const Rot         As Out = Out.M1
  Const Gruen       As Out = Out.O2
  Const Anforderung As Inp = Inp.I1

    fl.SetMotor(Rot, Dir.Ein)
    Do
      If fl.GetInput(Anforderung) Then
        Console.WriteLine("Signal kommt")
        fl.Pause(5000)
        fl.SetMotor(Rot, Dir.Aus)
        fl.SetMotor(Gruen, Dir.Ein)
        fl.Pause(10000)

        For i As Integer = 1 To 3
            fl.SetMotor(Gruen, Dir.Aus)
            fl.Pause(1000)
            fl.SetMotor(Gruen, Dir.Ein)
            fl.Pause(1000)
        Next

        fl.SetMotor(Rot, Dir.Ein)
        fl.SetMotor(Gruen, Dir.Aus)
      End If
    Loop Until fl.Finish()
```

`Const`: Um die Lesbarkeit des Programmes zu erhöhen werden hier anstelle der bisherigen `Enums` (von FishFace) Konstanten mit "sprechenden" Namen verwendet. Hinzugekommen ist eine `For` -Schleife in der Gruen 3mal blinkt.

[Born] Const Seite 54, Enum Seite 94, For .. Next [MS] Seite 105

Leuchtturm mit Blinklicht



Betrieb eines Leuchtturms mit zwei weißen Lampen (M1, M2) an der Turmspitze.

1 : Gleichtaktfeuer - Hell und Dunkel sind gleich lang (2 Sekunden)

2 : Blitzprinzip - Die Lichtphasen sind kürzer als die Dunkelphasen (0,5 / 1,5 Sekunden)

3. Blinkprinzip - Lichtphasen sind kürzer als die Dunkelphasen. Die Lampen leuchten unabhängig voneinander.

Version 1 :

```
Do
  fl.SetMotor(Out.M1, Dir.Ein)
  fl.SetMotor(Out.M2, Dir.Ein)
  fl.Pause(2000)
  fl.SetMotor(Out.M1, Dir.Aus)
  fl.SetMotor(Out.M2, Dir.Aus)
  fl.Pause(2000)
Loop Until fl.Finish()
```

Version 2 :

```
Do
  fl.SetMotor(Out.M1, Dir.Ein)
  fl.SetMotor(Out.M2, Dir.Ein)
  fl.Pause(300)
  fl.SetMotor(Out.M1, Dir.Aus)
  fl.SetMotor(Out.M2, Dir.Aus)
  fl.Pause(1500)
Loop Until fl.Finish()
```

Wie gehabt, aber mit anderen Zeiten.

Version 3 - ist haarig :

```
Imports System.Threading
Imports FishFace40
Module LeuchtTurm3
  Dim Lampe1 As New Thread(AddressOf Lampe1Blinken)
  Dim Lampe2 As New Thread(AddressOf Lampe2Blinken)
  Dim fl As New FishFace()

  Sub Main()
    Console.WriteLine("--- Gestartet : Abbruch mit ESC-Taste ---")
    fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    Lampe1.Start()
    Lampe2.Start()

    Lampe1.Join()
    Lampe2.Join()
    fl.CloseInterface()
    Console.WriteLine("--- Schließen : Return-Taste ---")
    Console.ReadLine()
  End Sub

  Private Sub Lampe1Blinken()
    Do
      fl.SetMotor(Out.M1, Dir.Ein)
      fl.Pause(2000)
      fl.SetMotor(Out.M1, Dir.Aus)
      fl.Pause(3000)
    Loop Until fl.Finish()
  End Sub
  Private Sub Lampe2Blinken()
    Do
      fl.SetMotor(Out.M2, Dir.Ein)
      fl.Pause(3000)
      fl.SetMotor(Out.M2, Dir.Aus)
      fl.Pause(5000)
    Loop Until fl.Finish()
  End Sub
End Module
```

Mit Robo Pro ist das deutlich einfacher : Man benötigt dazu nur ein zweites grüne Männchen

Mit VB2010 geschieht das über Threading (mehrere unabhängige Programmzweige) :

```
Imports System.Threading
```

Einbeziehen des Namensraumes für Threading.

```
  Dim Lampe1 As New Thread(AddressOf Lampe1Blinken)
  Dim Lampe2 As New Thread(AddressOf Lampe2Blinken)
```

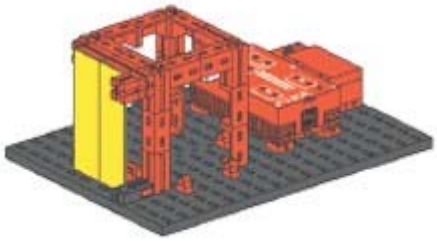
Definieren von zwei Threads ("Programmfäden") die eigentliche Verarbeitung geschieht in den zugehörenden Threadroutinen (`AddressOf` ..)

```
    Lampe1.Start()
    Lampe2.Start()

    Lampe1.Join()
    Lampe2.Join()
```

Die Threads werden aus Main heraus gestartet, anschließend wird auf deren Ende gewartet (geschieht - wie gewohnt - durch ESC-Taste)

Geblinkt wird in den Threadroutinen (`Sub`) Lampe1Blinken, Lampe2Blinken. Das Blinken selber wie gewohnt.

Kühlschrank



Simulation eines Kühlschranks.

1 : Tür offen (Taster I1 aus) - weiße Lampe an

2 : Zusätzlich - nach 3 Sekunden Rotblinken.

Version 1 (harmlos) :

```
    Do
      If Not fl.GetInput(Inp.I1) Then
        fl.SetMotor(Out.M1, Dir.Ein)
      Else
        fl.SetMotor(Out.M1, Dir.Aus)
      End If
    Loop Until fl.Finish()
```

`fl.GetInput` liefert als Return-Wert `True`, wenn der Sensor geschlossen ist und `False`, wenn nicht. Im `If` muß ein logischer Ausdruck stehen, der `True` oder `False` liefert, das tut GetInput.

Wenn man nun aber lieber den Then- und den Else-Zweig vertauschen will, kann man das durch ein vorangestelltes `Not` tun. Man kann aber auch einfach `fl.GetInput(Inp.I1) = False` schreiben.

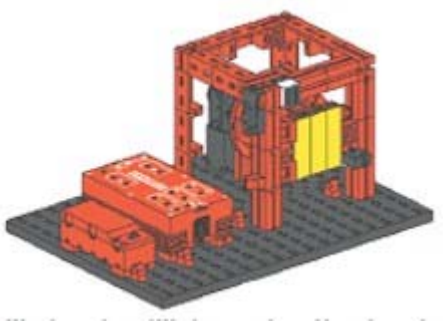
Version 2 (wieder eine Thread-Lösung) :

Funktioniert genauso wie die Leuchtturm-Lösung, deswegen hier nur die Thread-Routine für das Rot-Blinken.

```
  Private Sub LampeRotBlinken()
    Do
      fl.WaitForInput(Inp.I1, False)
      fl.Pause(3000)
      Do Until fl.GetInput(Inp.I1) Or fl.Finish()
        fl.SetMotor(Out.M2, Dir.Ein)
        fl.Pause(500)
        fl.SetMotor(Out.M2, Dir.Aus)
        fl.Pause(500)
      Loop
    Loop Until fl.Finish()
  End Sub
```

Es wird endlos auf das Öffnen der Kühlschranktür gewartet. Wird sie geöffnet, wird nach 3 Sekunden Rot geblinkt. Der Do -- Loop fragt ständig den Türtaster ab und enthält ein zusätzlich ein `Or fl.Finish()` um die Abbrechbarkeit des Programms sicherzustellen.

Waschmaschine



Die Trommel der Waschmaschine wird nach allen Regeln der Kunst gescheucht. Motor an M1, Anzeige an M2, Starttaster an I1, TuerZu-Taster an I2.

1: Nach Starttaster langsam Drehen für 10 Sekunden, 1 Sekunde Pause

2 : Wie 1 plus Warten auf TürZu

3 : Plus Schleudergang

4 : Plus Trocknen und TextAnzeige

Version 1 :

```
    Do
      Console.WriteLine("--- Start Waschgang : I1-Taster ---")
      fl.WaitForInput(Starter)
      fl.SetMotor(Anzeige, Dir.Ein)
      fl.SetMotor(Motor, Dir.Rechts, Speed.L2)
      fl.Pause(10000)
      fl.SetMotor(Motor, Dir.Aus)
      fl.SetMotor(Anzeige, Dir.Aus)
      fl.Pause(1000)
    Loop Until fl.Finish()
```

Version 2 :

```
    Do
      Console.WriteLine("--- Start Waschgang : I1-Taster ---")
      Console.WriteLine("--- Tür muß geschlossen sein ---")
      Do
        fl.WaitForInput(Starter)
      Loop Until fl.GetInput(TuerZu)
      fl.SetMotor(Anzeige, Dir.Ein)
      fl.SetMotor(Motor, Dir.Rechts, Speed.L2)
----
    Loop Until fl.Finish()
```

In Do --- Loop Until wird auf den Stattaster gewartet, der Do-Loop wird aber erst verlassen, wenn TuerZu ist.

Version 3 :

Mit dem versprochenen Schleudergang :

```
      fl.SetMotor(Motor, Dir.Rechts, Speed.Full)
      fl.Pause(15000)
```

Version 4 :

Mit Trockengang und Textausgabe zur Anzeige des aktuellen Waschganges.

```
    Do
      Console.WriteLine("--- Start Waschgang : I1-Taster ---")
      Console.WriteLine("--- Tür muß geschlossen sein ---")
      Do
        fl.WaitForInput(Starter)
      Loop Until fl.GetInput(TuerZu)
      fl.SetMotor(Anzeige, Dir.Ein)

      Console.WriteLine("Waschen")
      fl.SetMotor(Motor, Dir.Rechts, Speed.L2)
      fl.Pause(10000)
      fl.SetMotor(Motor, Dir.Aus)
      fl.Pause(1000)
```

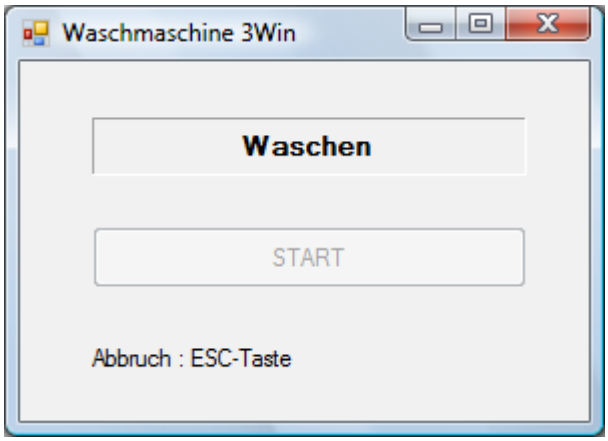
```
      Console.WriteLine("Schleudern")
      fl.SetMotor(Motor, Dir.Rechts, Speed.Full)
      fl.Pause(15000)
      fl.SetMotor(Motor, Dir.Aus)
      fl.Pause(1000)

      Console.WriteLine("Trocknen")
      Trocknen(Dir.Rechts)
      Trocknen(Dir.Links)
      fl.SetMotor(Anzeige, Dir.Aus)
    Loop Until fl.Finish()
```

Und der zugehörenden Sub Trocknen. Hier wird der Parameter Richtung zu Wechsel der Drehrichtung genutzt.

```
  Private Sub Trocknen(ByVal Richtung As Dir)
    fl.SetMotor(Motor, Richtung, Speed.L1)
    fl.Pause(10000)
    fl.SetMotor(Motor, Dir.Aus)
    fl.Pause(1000)
  End Sub
```

[Born] Sub Seite 105

Windows-Version



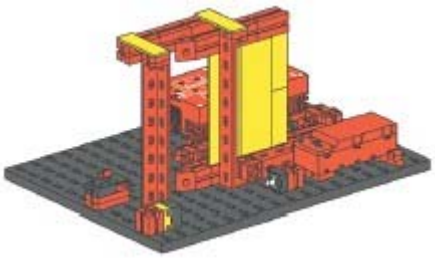
4Win wie Console Version 4.

Das Programm läuft vollständig in der Click-Routine, die zum START-Button (cmdAction) gehört und gleicht dem ConsoleProgramm fast vollständig.
Unterschied :
Die Ausgaben werden hier nicht fortlaufend in das DOS-Fenster geschrieben, sondern in den Label lblStatus.
Der laufende Betrieb wird ebenfalls über die ESC-Taste abgebrochen, die Form wird wie üblich über X (rechts oben) geschlossen. Der START-Button wird während des Betriebes deaktiviert.

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
    fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    Do
      lblStatus.Text = "Starten : I1-Taster"
      cmdAction.Enabled = False
      Do
        fl.WaitForInput(Starter)
        If Not fl.GetInput(TuerZu) Then _
                                lblStatus.Text = "Tür ZU, nochmal"
      Loop Until fl.Finish() OrElse fl.GetInput(TuerZu)
      fl.SetMotor(Anzeige, Dir.Ein)

      lblStatus.Text = "Waschen"
      fl.SetMotor(Motor, Dir.Rechts, Speed.L2)
      fl.Pause(10000)
      fl.SetMotor(Motor, Dir.Aus)
      fl.Pause(1000)

      lblStatus.Text = "Schleudern"
      fl.SetMotor(Motor, Dir.Rechts, Speed.Full)
      fl.Pause(15000)
      fl.SetMotor(Motor, Dir.Aus)
      fl.Pause(1000)

      lblStatus.Text = "Trocknen"
      Trocknen(Dir.Rechts)
      Trocknen(Dir.Links)
      fl.SetMotor(Anzeige, Dir.Aus)
    Loop Until fl.Finish()
    cmdAction.Enabled = True
    lblStatus.Text = "Abbruch"
    fl.CloseInterface()
  End Sub
```

Hinzu kommt noch die Routine Trocknen analog der DOS Version.

Schiebetür



Schiebetür mit Lichtschranke

1 : Unbedingtes Schließen der Tür

2 : Öffnen der Tür, wenn Lichtschranke unterbrochen. Schließen nach 10 Sekunden

3 : Zusätzlich beim Schließen der Tür : Abbruch, wenn Lichtschranke unterbrochen, wieder öffnen, nach 5 Sekunden erneuter Versuch

Version 1 :

```
Console.WriteLine("--- Tür wird geschlossen ---")
fl.SetMotor(Tuer, Schliessen, Speed.Half)
fl.WaitForInput(TuerZu)
fl.SetMotor(Tuer, Dir.Aus)
fl.Pause(1234)
```

eher einfach

Version 2 :

```
fl.SetMotor(LLampe, Dir.Ein)
fl.Pause(500)
fl.SetMotor(Tuer, Schliessen)
fl.WaitForInput(TuerZu)
fl.SetMotor(Tuer, Dir.Aus)

Do
   fl.WaitForInput(Lichtschranke, Unterbrochen)
   fl.SetMotor(Tuer, Oeffnen)
   fl.WaitForInput(TuerOffen)
   fl.SetMotor(Tuer, Dir.Aus)
   fl.Pause(10000)

   fl.SetMotor(Tuer, Schliessen)
   fl.WaitForInput(TuerZu)
   fl.SetMotor(Tuer, Dir.Aus)
Loop Until fl.Finish()
```

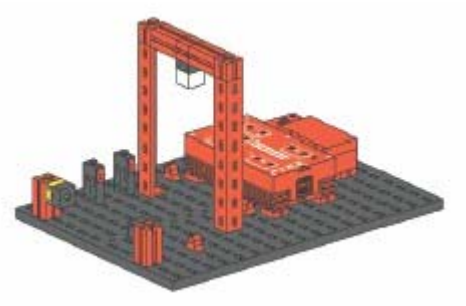
Erstmal Tür schließen und dann im Endlos-Loop auf `Lichtschranke Unterbrochen` warten. Tüer öffnen und nach 10 Sekunden wieder schließen.

Version 3 :

```
-------
Do
   ----
   fl.SetMotor(Tuer, Schliessen)
   Do Until fl.GetInput(TuerZu) Or fl.Finish()
      If fl.GetInput(Lichtschranke) = Unterbrochen Then
           fl.SetMotor(Tuer, Oeffnen)
           fl.WaitForInput(TuerOffen)
           fl.SetMotor(Tuer, Dir.Aus)
           fl.Pause(5000)
           fl.SetMotor(Tuer, TuerZu)
      End If
   Loop
   fl.SetMotor(Tuer, Dir.Aus)
Loop Until fl.Finish()
```

Hier wird die Tür nicht mehr einfach nach 10 Sekunden geschlossen. Bei Schliessen wird ständig nach Lichtschranke = Unterbrochen gefragt und dann ggf. die Tüer wieder geöffnet.

Treppenhausbeleuchtung



Treppenhausbeleuchtung mit Tastern und Lichtschranke.

1: Wenn Taster I1 oder I2 gedrückt wird, Beleuchtung für 10 Sekunden anschalten.

2 : Einschalten zusätzlich durch Unterbrechung der Lichtschranke

3 : Wechselschalter : Einschalten und Ausschalten über Taster.

Version 1 :

```
Do
  If fl.GetInput(Inp.I1) Or fl.GetInput(Inp.I2) Then
    fl.SetMotor(Out.M1, Dir.Ein)
    fl.Pause(10000)
  Else
    fl.SetMotor(Out.M1, Dir.Aus)
  End If
Loop Until fl.Finish()
```

Wenn einer der Taster an I1 oder I2 betätigt wird, für 10 sekunden Einschalten

Version 2 :

```
Do
  If fl.GetInput(Inp.I1) Or fl.GetInput(Inp.I2) _
                         Or Not fl.GetInput(Inp.I3) Then
    fl.SetMotor(Out.M1, Dir.Ein)
    fl.Pause(10000)
  Else
    fl.SetMotor(Out.M1, Dir.Aus)
  End If
Loop Until fl.Finish()
```

Wie 1 : Die Abfrage wurde um `Not fl.GetInput(Inp.I3)` (Lichtschranke) erweitert.

Version 3 :

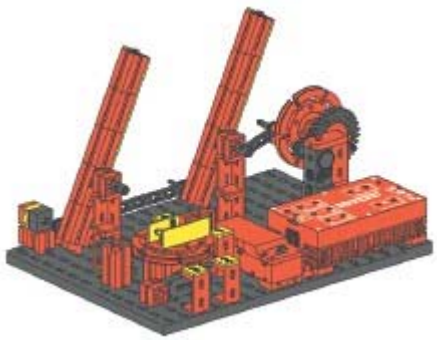
```
    Try
      fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
      Do
        WaitForOff()
        WaitForOn()
        fl.SetMotor(Beleuchtung, Dir.Ein)
        WaitForOff()
        WaitForOn()
        fl.SetMotor(Beleuchtung, Dir.Aus)
      Loop Until fl.Finish()
    Catch e As Exception
      Console.WriteLine(e.Message)
    Finally
      fl.CloseInterface()
    End Try
```

```
  Private Sub WaitForOn()
    Do
      fl.Pause(50)
    Loop Until fl.GetInput(Taster1) Or fl.GetInput(Taster2) _
                                    Or fl.Finish()
  End Sub
  Private Sub WaitForOff()
    fl.WaitForInput(Taster1, False)
    fl.WaitForInput(Taster2, False)
  End Sub
```

Hier wurde der Programm-Rahmen einwenig geändert. Das gesamte Programm läuft jetzt unter der Überwachung des `Try - Catch - Finally` Blocks. Es gibt jetzt nicht mehr die ekligen Meldungen, wenn man mal vergessen hat den LT Controller vor Start des Programms zu starten, es wird bei `Catch` nur noch dezent gemeckert.

Zum Erkennen werden die Subs WaitForOn und WaitForOff eingesetzt.

Scheibenwischer



Scheibenwischer, der über einen Drehschalter in den Betriebsarten Interval, Langsam und Schnell betrieben werden kann.

1 : Drehschalter auf Stellung 1 : Schnellgang

2 : Drehschalter 1 Langsam, 2 Schnell

3 : Drehschalter 1 Interval , 2 Langsam, 3 Schnell

Version 1 : Taster 1 an - Scheibenwischer im Schnellgang

```
Do
  If fl.GetInput(Inp.I1) Then
    fl.SetMotor(Out.M1, Dir.Rechts)
  Else
    fl.SetMotor(Out.M1, Dir.Aus)
  End If
Loop Until fl.Finish()
```

Version 2 Taster 1 an - Langsamgang, Taster 1 und 2 an : Langsamgang

```
Do
  If fl.GetInput(Inp.I1) And fl.GetInput(Inp.I2) Then
    fl.SetMotor(Out.M1, Dir.Rechts)
  ElseIf fl.GetInput(Inp.I1) Then
    fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
  Else
    fl.SetMotor(Out.M1, Dir.Aus)
  End If
Loop Until fl.Finish()
```

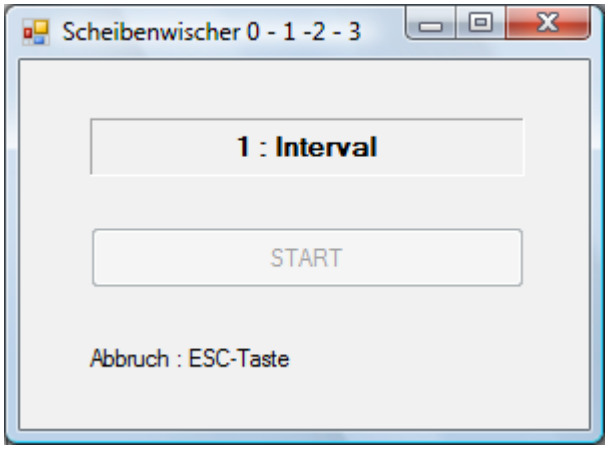
Version 3 :

```
fl.SetMotor(Out.M2, Dir.Ein)
fl.Pause(500)
Do
  If fl.GetInput(Inp.I1) And fl.GetInput(Inp.I2) Then
    fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
  ElseIf fl.GetInput(Inp.I1) And Not fl.GetInput(Inp.I2) Then
    fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
    fl.WaitForChange(Inp.I3, 4) ' --- Lichtschranke
    fl.SetMotor(Out.M1, Dir.Aus)
    fl.Pause(2000)
  ElseIf Not fl.GetInput(Inp.I1) And fl.GetInput(Inp.I2) Then
    fl.SetMotor(Out.M1, Dir.Rechts)
  Else
    fl.SetMotor(Out.M1, Dir.Aus)
  End If
Loop Until fl.Finish()
```

Taster 1 an und Taster 2 aus (Pos 1) : Intervalschaltung
Taster 1 und 2 an (Pos 2) : Langsamgang
Hier wurde anstelle der drei WaitForInput von Robo Light die Methode WaitForChange eingesetzt, die die Intervalsteuerung besser löst. Hier mit vier Wechseln Lichtschranke True/False bzw. False/True.
Taster 1 aus und Taster 2 an (Pos 3) : Schnellgang

[Born] ElseIf Seite 89

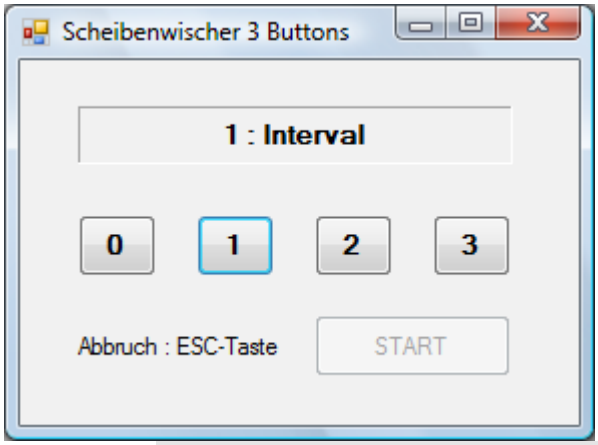
Windows-Versionen



3Win wie Console Version 3.

Das Programm läuft vollständig in der Click-Routine, die zum START-Button (cmdAction) gehört und gleicht dem ConsoleProgramm fast vollständig.
Unterschied :
Die Ausgaben werden hier nicht fortlaufend in das DOS-Fenster geschrieben, sondern in den Label lblStatus.
Der laufende Betrieb wird ebenfalls über die ESC-Taste abgebrochen, die Form wird wie üblich über X (rechts oben) geschlossen. Der START-Button wird während des Betriebes deaktiviert.

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles cmdAction.Click
  fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
  cmdAction.Enabled = False
  fl.SetMotor(Out.M2, Dir.Ein)
  lblStatus.Text = "Drehschalter 0 - 1 - 2 - 3"
  fl.Pause(3000)
  Do
    If fl.GetInput(Inp.I1) And fl.GetInput(Inp.I2) Then
      lblStatus.Text = "2 : Langsam"
      fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
    ElseIf fl.GetInput(Inp.I1) And Not fl.GetInput(Inp.I2) Then
      lblStatus.Text = "1 : Interval"
      fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
      fl.WaitForChange(Inp.I3, 3)
      fl.SetMotor(Out.M1, Dir.Aus)
      fl.Pause(2000)
    ElseIf Not fl.GetInput(Inp.I1) And fl.GetInput(Inp.I2) Then
      lblStatus.Text = "3 : Schnell"
      fl.SetMotor(Out.M1, Dir.Rechts)
    Else
      lblStatus.Text = "0 : Aus"
      fl.SetMotor(Out.M1, Dir.Aus)
    End If
    fl.Pause(112)
  Loop Until fl.Finish()
  lblStatus.Text = "Abbruch"
  fl.CloseInterface()
  cmdAction.Enabled = True
End Sub
```



3bWin : Ohne Nutzung des Drehschalters. Dessen Funktion wird durch die Buttons 0 - 3 wahrgenommen. Auch hier der Betrieb in der Click-Routine von cmdAction. Hinzu kommt die gemeinsame Click-Routine für die Buttons 0 - 3. Dort wird der jeweilige Wert der Eigenschaft .Tag (0 - 3) in die globale Variable Modus gestellt.

Der Ablauf von cmdAction_Click entspricht dem des vorhergehenden Beispiels. Wesentlicher Unterschied : Anstelle der Taster-Zustände wird die globale Variable Modus in einem Select Statement ausgewertet

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles cmdAction.Click
    cmdAction.Enabled = False
    fl.OpenInterface(IFTypen.ftROBO_first_USB, 0)
    fl.SetMotor(Out.M2, Dir.Ein)
    lblStatus.Text = "Button 0 - 1 - 2 - 3"
    Do
      Select Case Modus
        Case 2
             lblStatus.Text = "2 : Langsam"
             fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
        Case 1
             lblStatus.Text = "1 : Interval"
             fl.SetMotor(Out.M1, Dir.Rechts, Speed.L2)
             fl.WaitForChange(Inp.I3, 4)
             fl.SetMotor(Out.M1, Dir.Aus)
             fl.Pause(2000)
        Case 3
             lblStatus.Text = "3 : Schnell"
             fl.SetMotor(Out.M1, Dir.Rechts)
        Case 0
             If ModusAlt > 0 Then
                   lblStatus.Text = "0 : Aus"
                   fl.WaitForInput(Inp.I3, False)
                   ModusAlt = 0
                   fl.SetMotor(Out.M1, Dir.Aus)
             End If
      End Select
      fl.Pause(12)
    Loop Until fl.Finish()
    fl.CloseInterface()
    cmdAction.Enabled = True
  End Sub
```

Hinzugekommen ist in Case 0: ein sauberes Abschalten des Wischers : Es wird erst unmittelbar nach Passieren der Lichtschranke abgeschaltet. Damit wird eine saubere Endstellung erreicht. Da nur bei Wechsel Go -> Off normiert werden soll, wird eine zweite globale Variable ModusAlt eingesetzt.

```
Private Sub ModusClick(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles cmdS0.Click, cmdS3.Click, cmdS2.Click, cmdS1.Click
    ModusAlt = Modus
    Modus = CType(sender, Button).Tag
  End Sub
```

[Born] Dim Seite 57, Integer Seite 59, Select Case Seite 90