ROBO Pro

ftComputing : Programme für die fischertechnik-Interfaces und -konstruktionskästen

ftComputing.de

Home
Back
Schweiss Rob
TeachIn Rob
Flieger-Karussell
Schrittmotoren
Turm von Hanoi

Sitemap
Index
Links
Impressum

Mail

Einige Anmerkungen zu ROBO Pro

ROBO Pro ist eine graphische Software zur Programmierung von fischertechnik Interfaces. ROBO Pro ist der Nachfolger von LLWin. Mit ROBO Pro kann das neue ROBO Interface samt Zusätzen im Online und im Download Modus mit allen Eigenschaften betrieben werden. Außerdem kann das Intelligent Interface - ggf. mit angeschlossenem Extension Module - im Online Modus betrieben werden. Zusätzlich gibt es auch noch einen Simulationsmodus, wo's dann auch ganz ohne geht, besonders für das Setzen von Eingängen nimmt man dann das Test-Panel.

ROBO Pro verfolgt ein Stufenkonzept in dem schrittweise komplexere Funktionen angeboten werden.
In Level 1 wird eine Auswahl der von LLWin bekannten Funktionen angeboten (keine Variablen, aber Unterprogramme).
Level 2 bietet zusätzlich Unterprogramm-Parameter und viel DrumRum : Zeichenfunktionen und eine Bibliothek (die auch Funktionen von Level 3 enthalten kann).
Bei Level 3 wirds dann ernst : Variablen und einen Trennung von Steuerfluß (schwarze Linie) und Datenfluß (gelbe Linien) sowie eine ganze Zahl von zusätzlichen Funktionen

für die Steuerung des Programflusses und zur Verknüpfung von Daten.

Programme

Der Schweißroboter ist im Kasten Computing Start Kit eines der komplexesten Modelle. Hier dient er als Vehikel zur Vorstellung verschiedener Programmiermöglichkeiten, angefangen beim Level 1 bis zum Level 3.

Der Schweißroboter wird dann auch noch als Vehikel für den Einsatz eines Bedienpanel und einer TeachIn-Steuerung gezeigt.

Das Flieger-Karussell ist eine lustige Angelegenheit, bei der man kräftig einen auf die Finger bekommen kann, wenn man ins laufende Modell faßt, drum langes Kabel für die Verbindung zum Interface. Ansonsten werden hier besonders Schleifentechniken gezeigt.

Für Fan der Schrittmotoren hier eine kleine Bibliothek zu deren Betrieb.

Anhand des schon etwas komplexeren Programmes Hanoi aus der Serie Hanoi Robots werden die Möglichkeiten von Level 3 gezeigt und erklärt.

Downloads

Die Sources für die Programme sind in RoboPro.ZIP zusammengefaßt.

Stand : 30.12.2004

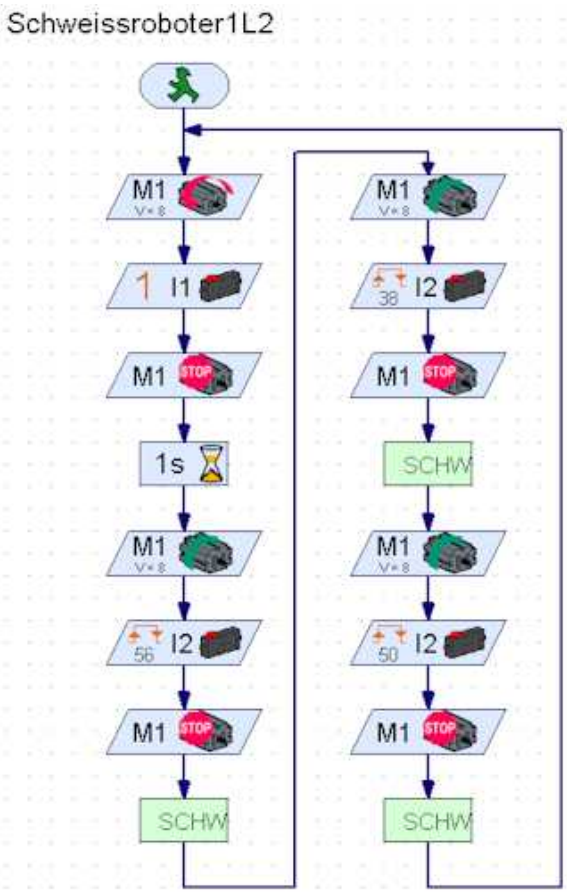
Schweiss Rob

Lösungen für das Modell Schweißroboter des Computing Starter

Aufbau : M1 mit Säulenmotor und I1 Endtaster, I2 Impulstaster, M4 mit Schweißlampe.
Bei mir war das eine Abwandlung des Hanoi-Roboters bei dem zusätzlich auf dem Arm eine Birne(M4) montiert wurde.

Vorgestellt werden verschiedene Lösungsmöglichkeiten, beginnend auf Level 2 mit einer einfachen sequentiellen Lösung bis hin zu einer komplexeren, listengesteuerten Lösung auf Level 3.
Auf der nächsten Seite wirds dann richtig schön : es kommt noch eine TeachIn-Lösung hinzu.

Schweissroboter1L2



Sieht beinahe wie ein LLWin-Programm aus. Es geht hier alles schön der Reihe nach :

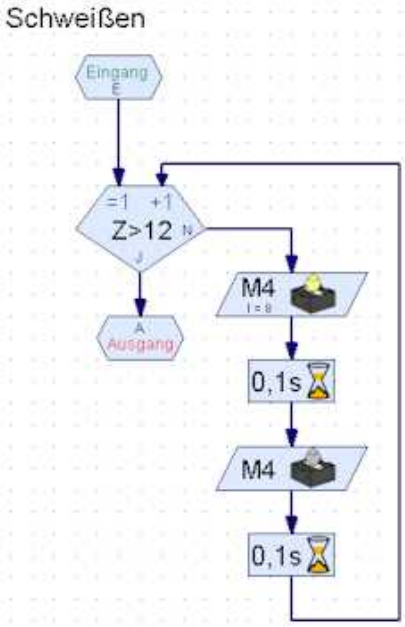
- Anfahren der Home-Position (bis Pause)
- Anfahren Position 56 und Schweißen
- Anfahren Position 94 (56 + 38)
- Anfahren Position 144 (94 + 50)

und das in einer Endlosschleife.

Neu (gegenüber LLWin) ist hier :

- die gelbe 1 : Warten auf Taster an I1 = true
  man spart also den bei LLWin üblichen Abfragekringel
- die gelben Pfeile : Zählen von Impulsen am angegebenen
  Eingang, hier werden alle gezählt (nur fallende, steigende
  sind auch möglich). Hier wird gegenüber LLWin konsequent
  der Gebrauch von externen Variablen vermieden.

Das Schweißen



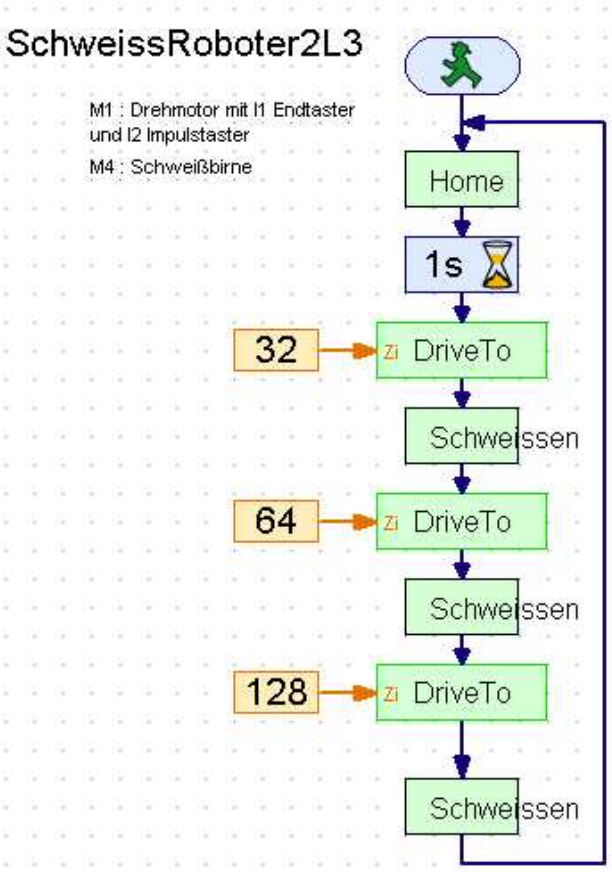
Interessant ist hier das Zählschleifenkonstrukt :

Hier wird eine Schleife 12 mal durchlaufen. Beim Start (Eingang =1) wird der interne Zähler auf 1 gesetzt, bei den weiteren Schleifendurchläufen jeweils um (+1) 1 erhöht.

Auch hier werden externe Variable vermieden. Die Konstruktion ist einfacher als die bei LLWin übliche Variablenabfrage und Incrementierung.

Schweißroboter2L3

Sequentieller Ablauf wie gehabt, aber :



- Auslagerung der Positionierung in ein Unterprogramm
  Das Unterprogramm bekommt die anzufahrende
  Position über einen Parameter mitgeteilt.

- Ebenso wurde das Anfahren der Home-Position in ein
  Unterprogramm verlagert.

- neues Element Konstanten, die über gelbe Linien mit
  dem Zielelement verbunden werden.

- Level 3 bringt eine Trennung von Kontrollfluß
  (schwarze Linien) und Datenfluß (gelbe Linien, sie
  werden in den weiteren Programmteilen vermehrt
  auftreten).

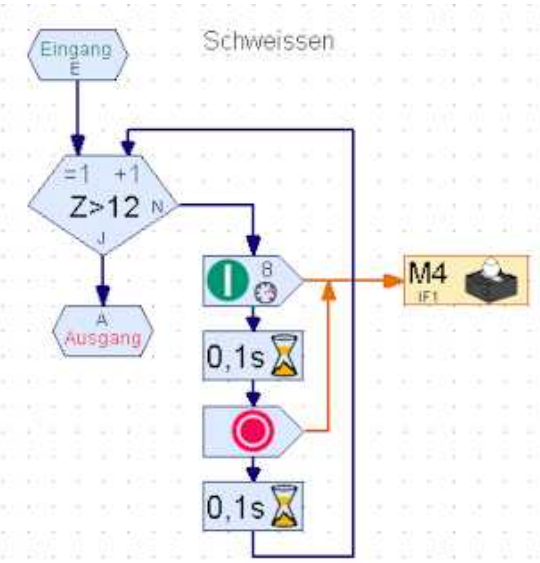
- Damit verbunden ist die Einführung von Konstanten,
  globalen und lokalen Variablen.

- Ebenso wird jetzt zwischen Befehlen (im Kontrollfluß)
  und Ein- und Ausgängen (z.B. Interface M1 - M4 und
  I1 - I8) sowie Operatoren (im Datenfluß) unter-
  schieden.

- Level 1/2 Symbole können mit Level 3 Symbolen gemischt

verwendet. Ob mans dann tut, ist Geschmackssache (und eine der Bequemlichkeit, siehe nachfolgende Routine Schweissen).

Das renovierte Schweißen

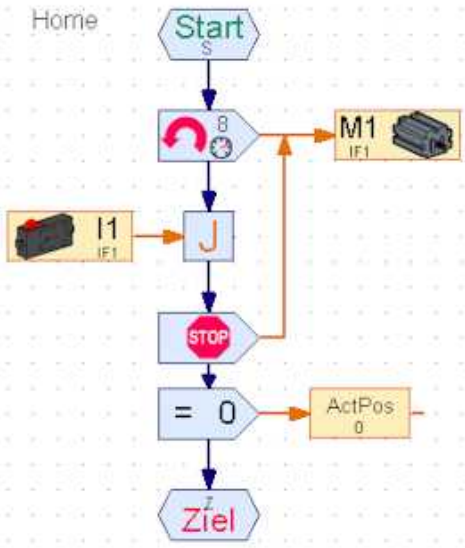


Der Ablauf ist wie in Level 1.

Die Schweißbirne wird jetzt aber über Level 3-Elemente angesteuert. Mit dem grünen Kreis geht es mit voller Helligkeit (8) los, warten und aus. Diese Befehle befinden sich im Kontrollfluß und senden über den Datenfluß (gelb) entsprechende Nachrichten an die Birne.

Das Zählelement wurde (faulheitshalber) beibehalten. Es wäre sonst eine Zählvariable, ein Additionbefehl und eine Abfrage fällig gewesen (mit gelben und schwarzen Linien). Das kann man dann weiter unten bewundern.

Anfahren der Home-Position



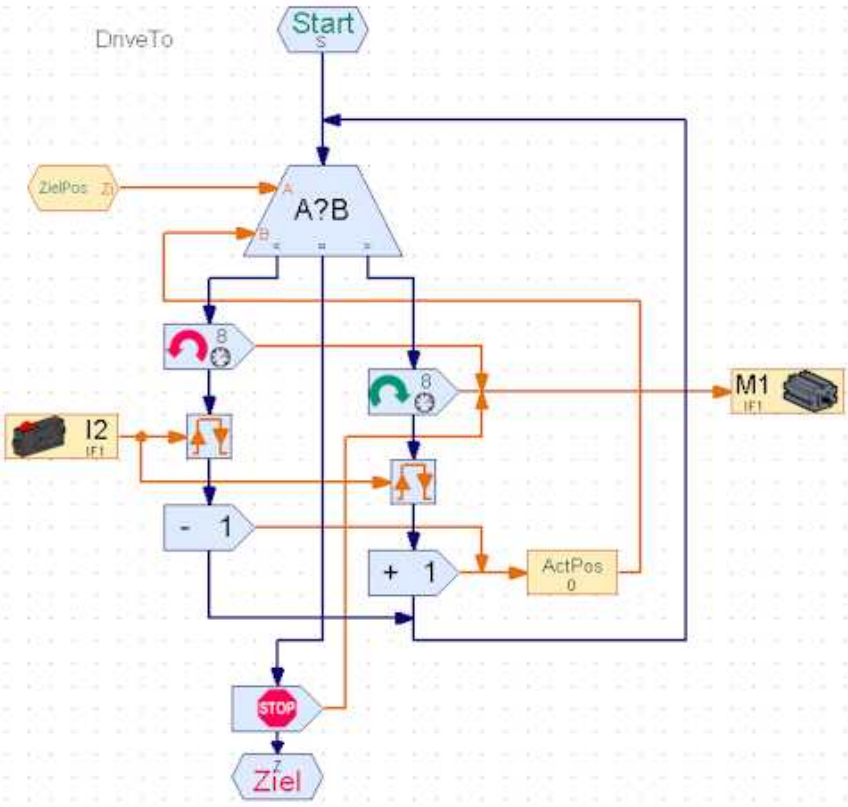
Hier mit dem gelben "J" die gelbe "1" von Level 1 in neuem Gewand : Warten auf I1 = true. Die Varianten Warten auf false oder steigende (0->1), fallende(1->0) Flanke sind ebenfalls möglich

Der Säulenmotor an M1 wird jetzt durch separate Befehle angesteuert (roter Linksdreh, Fullspeed und STOP).

Neu sind hier auch der "=" Befehl, der die Nachricht 0 an die (globale) Variable ActPos sendet. D.h. die Säule steht auf Position 0 gerechnet ab Endtaster an I1.

Mit dem Wert 0 (bei ActPos) wird die Variable bei Programmstart auf 0 initialisiert. Das =0 ist hier erforderlich, das Home mehrfach aufgerufen wird (und natürlich der Ordnung halber).

Anfahren einer vorgegebenen Position



Ist schon etwas vertrackter.

Die anzufahrende Position wird über den Parameter ZielPos vorgegeben (immer gerechnet ab Endtaster). Die aktuelle Position (auch ab Endtaster) wird in der globalen Variablen ActPos gehalten (im Hauptprogramm tritt sie gar nicht in Erscheinung, sie hier und in Home genutzt).

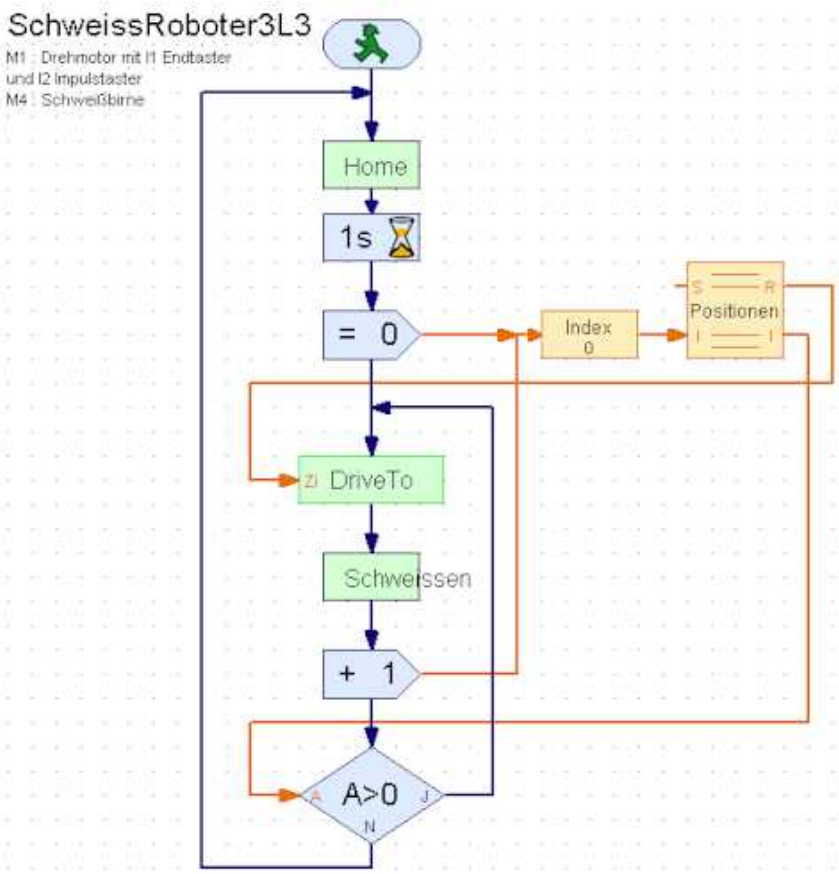
Zentraler Befehl ist die Abfrage A?B mit der ZielPos und ActPos verglichen werden. Bei Gleichheit ist zu Ende, Motor Aus.

Ist ZielPos > ActPos gehts nach rechts mit M1, die bei I2 auftretenden Impulse werden gezählt ("Pfeil"-Befehl) und ActPos wird jeweils um 1 erhöht bis es dann gleich ist.

Bei ZielPos < ActPos gehts nach links in Richtung Endtaster, ansonsten sinngemäß.

Man könnte hier natürlich (vorsichtshalber) auch noch den Endtaster an I1 abfragen und die maximal mögliche Fahrstrecke. Das kriegen wir später.

Schweißroboter3L3



Ist schon beinahe eine Luxusausführung. Hier werden die anzufahrenden Positionen in der Tabelle (oder Liste) Positionen gespeichert. In diesem Beispiel schon bei der Programmerstellung.

Die Tabelle erhält über den I-Eingang die Nummer (Index, ab 0) des Tabellenelementes das über den R-Ausgang auszugeben ist. Hier landet es dann als Parameter bei DriveTo.

Der I-Ausgang gibt bei einem Indexwert kleiner der Anzahl Tabellenelemente (hier 3) die aktuelle Anzahl der Tabellenelemente (hier also 3) aus, bei Index 3 oder größer wird 0 ausgegeben. Das wird zur Schleifensteuerung bei A>0 genutzt.

Die verwendeten Unterprogramme entsprechen denen vom vorhergehenden Beispiel.

Der aktuelle Index wird in einer Variablen gehalten (augenscheinlich erforderlich, da der +1 Befehl sonst nicht ankommt, der =0 tuts).

Stand : 20.12.2004

TeachIn Rob

Der Schweißroboter mit Bedienpanel und TeachIn

Aufbau : wie gehabt.

Vorgestellt werden der Einsatz eines Bedienpanels mit Positionsanzeige und eine einfache TeachIn-Lösung für den Schweißroboter zur Bestimmung der Schweißpositionen. Das Programm verwendet einige Komponenten der vorherigen Schweißroboter-Programme : Home, DriveTo, Schweissen sind geblieben, Das Main wurde in das neue UP Play verlagert. Hinzugekommen sind : Das Panel, ein neues Main, Record, Store und Go.

Das Bedienpanel

Oben die Anzeige der aktuellen **Position**. Das Update der Position erfolgt über Bedienfeldausgänge, die in Home, Play und Go platziert sind. Die Elemente des Bedienfeldes selber sind alle dem zentral dem Main zugeordnet.

Play startet den Schweißablauf, geschweißt wird solange, wie Play gedrückt ist (Play hat dazu eine Rastfunktion bekommen). Die Schweiß-Positionen werden der Tabelle Positionen entnommen.

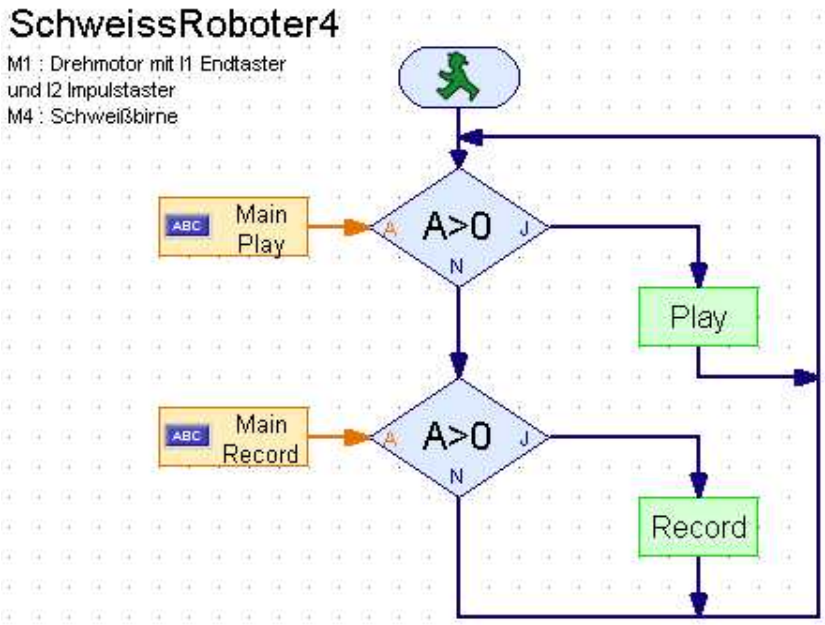
Record startet eine Aufzeichnung durch Anfahren der Home-Position und Löschen der Positionstabelle.

Go läßt den Robot nach rechts (weg vom Endtaster) fahren, das gilt solange er (der Button) gedrückt ist (keine Rastfunktion).

Store speichert die aktuelle Position aus ActPos durch Anhängen an Tabelle Positionen.

End beendet den Aufzeichnungsmodus.

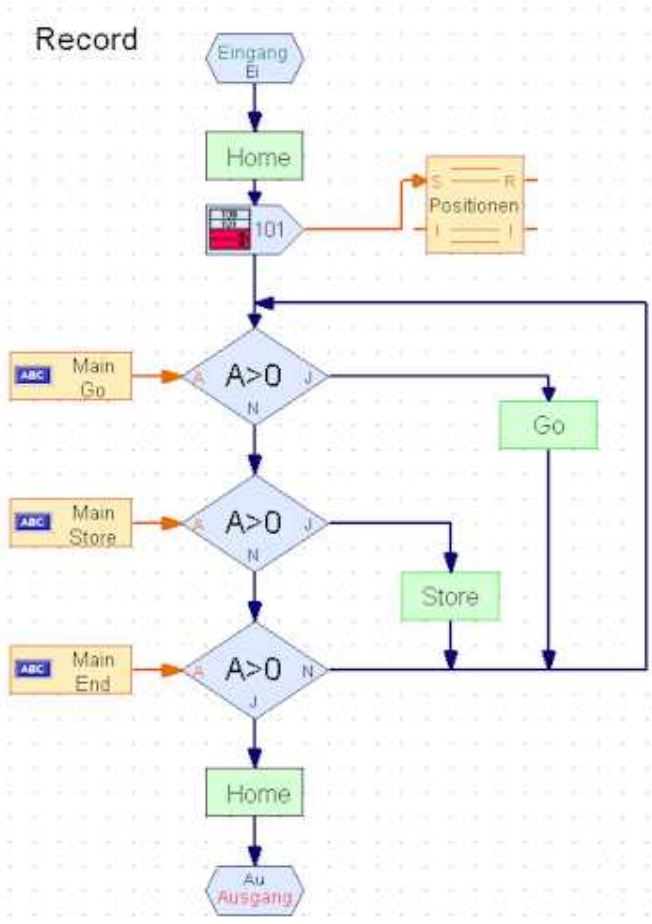
Der Main-Loop



Im Main werden die Bedienfeldeingänge **Play** und **Record** in einer Endlosschleife abgefragt. Im zutreffenden Fall werden dann die UP's Play und Record aufgerufen.

Die Bedienfeldeingänge lösen selber keine Ereignisse aus, deswegen hier die etwas umständliche Schleifenlösung, die auch bei Record nocheinmal wieder auftaucht.

Das Aufzeichnen : Record, Go und Store



Anfahren der **Home** Position.

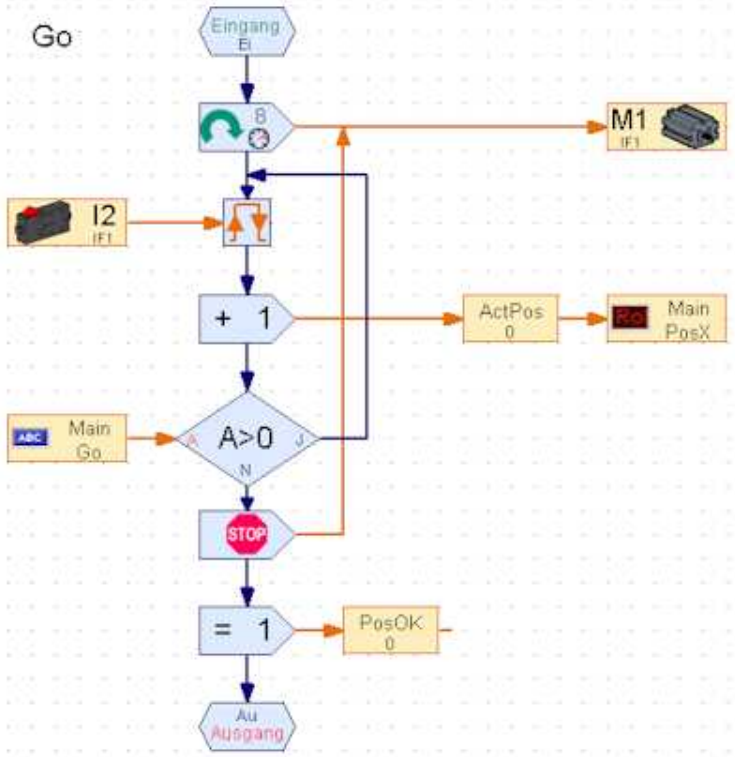
Löschen der Tabelle Positionen durch Entfernen der letzten 101 Elemente. Da die Tabelle nur 100 Elemente aufnehmen kann ist sie dann leer.

Loop in dem der Go, der Store und der End-Button abgefragt werden. Im zutreffenden Fall werden dann Go und Store auf gerufen. Bei End wird nach Anfahren der Home-Positionen das UP verlassen.

Hinweis : Die Buttons sind auf dem Bedienfeld von Main platziert.

Hinweis 2 : Ein Drücken eines Buttons ist so schnell, daß Maßnahmen zur Vermeidung eines Mehrfachaufrufs eines UP's getroffen werden müssen. Dazu wird in Go die globale Variable PosOK beim Verlassen auf 1 (true) gesetzt. Store seinerseits nimmt die Arbeit nur auf, wenn PosOK = 1 ist und setzt selber PosOK auf 0 (false).

Das Ansteuern einer Position



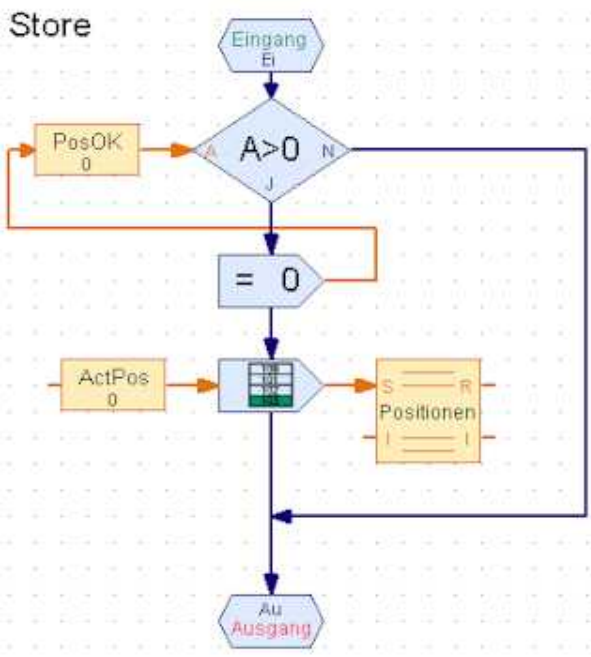
Starten des Säulenmotors

Warten auf Impulstaster, erhöhen um 1 und Speichern in ActPos, wenn Go immernoch gedrückt in Schleife zum Impulszählen.

Verlassen des UP's, wenn Go nicht mehr gedrückt ist, vorher noch PosOK = true signalisieren.

Die aktuelle **Position** wird dem Bedienfeldausgang PosX (in Main) übergeben.

Speichern der aktuellen Position



Bevor es mit dem Speichern losgeht, erstmal fragen, ob es auch erlaubt ist : **PosOk** = true. In diesem Fall wird es dann gleich auf 0 (false) gesetzt um beim Speichern nicht gestört zu werden.

Das **Speichern** der ActPos geschieht über einen entsprechenden Tabellenbefehl, der den Wert von ActPos hinten an die Tabelle anhängt. Angesteuert wird der S(Speicheren)-Eingang von Positionen.

Das Spreichern ist die einfachere Übung, Schwierigkeiten macht das fehlende **Click**-Ereignes des Store-Buttons.

Das wars denn auch schon. Man kann sich das Leben natürlich noch komplizierter einrichten, wenn man sich einem Industry Robot zuwendet. Doch das kriegen wir später.

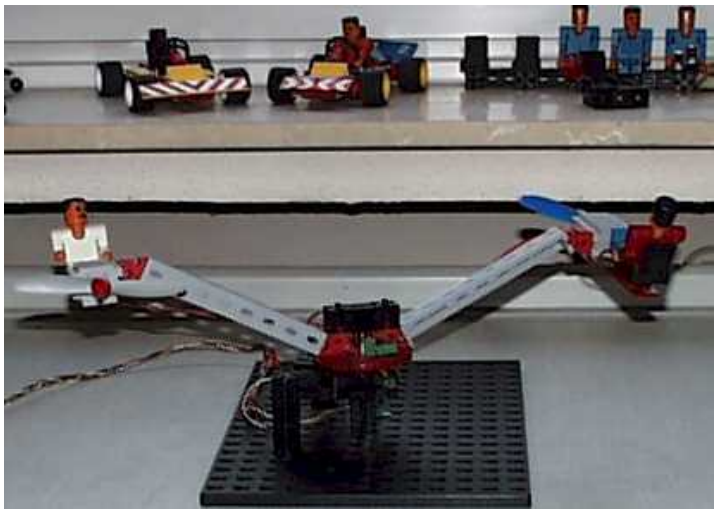
Stand : 20.12.2004

Flieger-Karussell

Zwei Luftschrauben treiben ein Karussell

Eigentlich reicht es, das Ding ein- und irgendwann wieder auszuschalten. Man kann es aber auch komplizierter machen und das über ein ROBO Pro-Programm tun, das dann durch Vor- und Rückwärtsgang und munteres Hupen Leben in die Geschichte bringt.

Das Modell



Das Modell ist sehr einfach : Grundplatte mit Achse in der Mitte darauf einer der alten dicken Schleifringe mit zwei Leiterbahnen. Stromzufuhr über zwei in gefederten Gelenksteinen gelagerten Schleifekontakten.

Obendrauf ein großes Scheibenrad mit zwei Armen an deren Enden Minimotoren mit Propeller drauf stecken. Dekoriert durch jeweils ein Männeken.

Und irgendwo dann noch ein Summer.

Das Interface sollte an einer Strippe hängen, damit man das Programm auch im laufenden Flug abstellen kann, ohne sich die Finger zu verbiegen. Das Karussell erreicht enorme Geschwindigkeiten.

Das Hauptprogramm

Ist sehr schlicht :

Der Reihe nach nur Unterprogrammaufrufe. Man könnte noch eine Wiederholungsschleife einbauen :

StartHupe : ein wildes Quäcken des Summers zum Auftakt.

Anlauf : Langsames Hochfahren der Geschichte. s.u.

Lauf8 : 10 Sekunden Fahren mit Fullspeed.

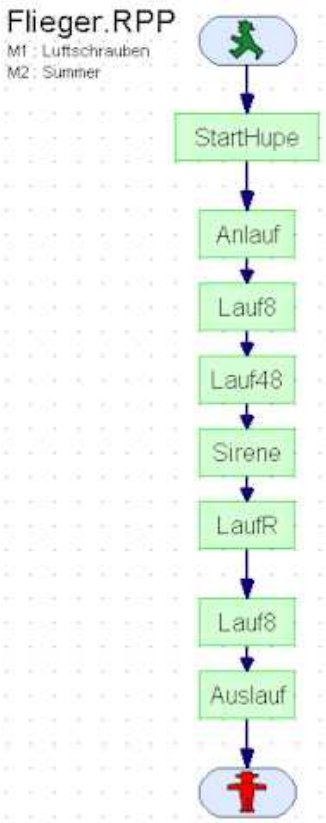
Lauf48 : Fahren mit wechselnden Geschwindigkeiten. s.u.

Sirene : Ein klägliches Jaulen zum Finale. s.u.

LaufR : Fullspeed Rückwärts

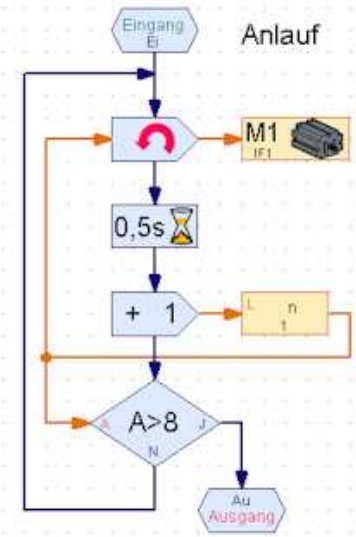
Lauf8 : Wieder vorwärts

Auslauf : Langsamer Auslauf mit Gegenstrombremsen zum



Schluß s.u.

Anlauf : Schleife über Geschwindigkeit, lokale Variable



Der Motor wird mit der niedrigsten Geschwindigkeitsstufe gestartet. die Geschwindigkeit wird dann alle halbe Sekunde um 1 erhöht, bis das Maximum (8) erreicht ist.

Für die fällige Schleife wird hier eine lokale Variable n, der Befehl +1 und eine einfache Abfrage genommen.

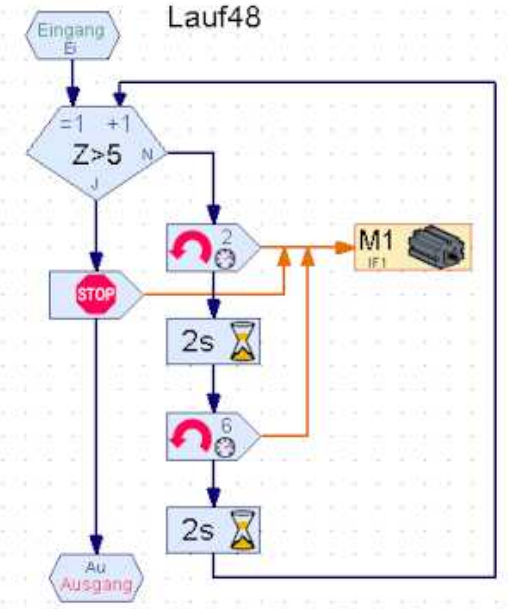
Da n eine lokale Variable ist, wird sie bei jedem Aufruf von Anlauf auf den im Kasten angegebenen Wert (1) initialisiert, die Schleife läuft also von 1 - 8.

Die Variable n dient gleichzeitig als Eingangswert für den Motor-Befehl.

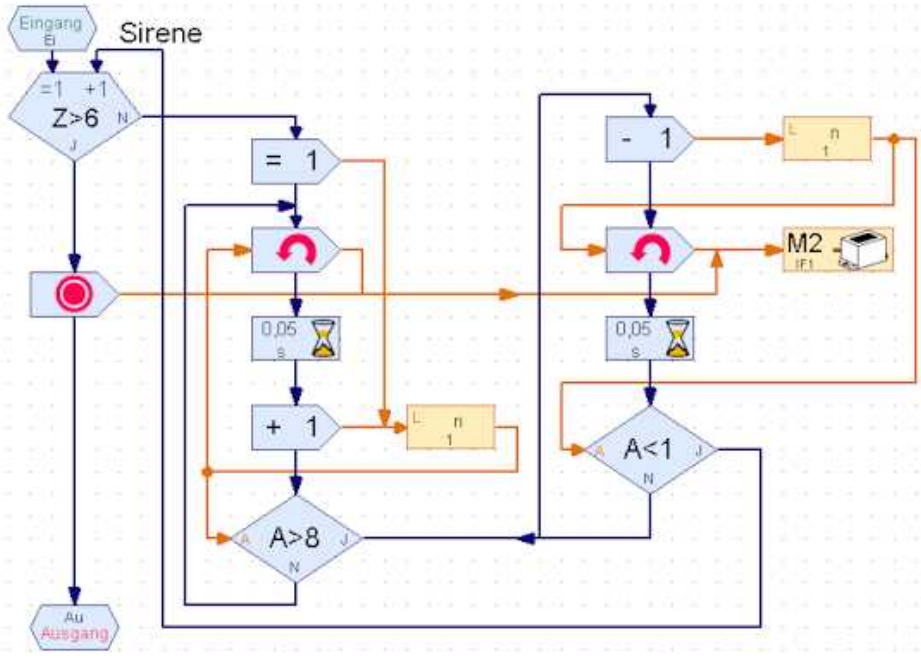
Fahren mit wechselnden Geschwindigkeiten

Wieder eine Schleife, die 5mal durchlaufen wird. Hier aber eine Schleife ohne externe Variable. Ist bequemer, wenn man keine Variable benötigt.

Der Motor läuft hier mit den wechselnden Geschwindigkeit 2 und 6 jeweils 2 Sekunden immer rechtsrum. Es geht natürlich auch anders ... probieren.



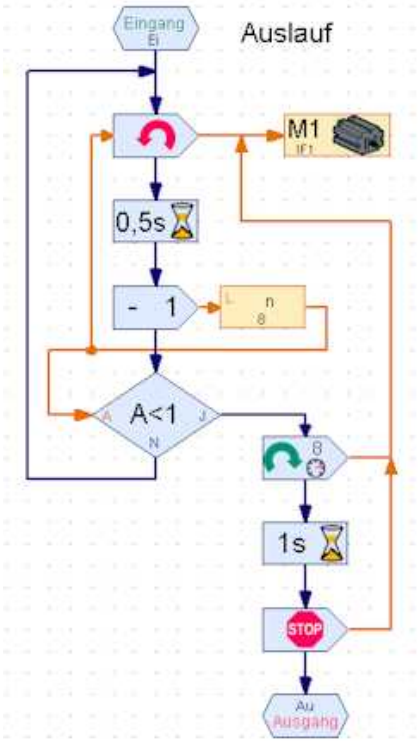
An- und abschwellende "Töne" auf dem Summer



Hier sind zwei Schleifen geschachtelt : die äußere ist eine einfache zur Zählung des An und Ab. Das An- und Abschwellen wird durch eine Schleife mit Variable (siehe Anlauf) realisiert. Da für die beiden innereren Schleifen die gleiche lokale Variable n (das ist ein anderes n als in Anlauf) genommen wird, enthalten sie auch die gleichen Werte d.h. sie sind real nur eine Variable, die hier aus Gründen der Übersichtlichkeit zweimal auftaucht. In diesem Fall muß n auf 1 initialisiert werden, da die Schleife mehrfach durchlaufen wird. Bei der zweiten Schleife ist das nicht erforderlich, da

sie in die Schleife mit dem Wert 9 kommt und der Befehl -1 zu Beginn der Schleife steht.

Der Auslauf



Der Auslauf bringt nicht mehr viel Neues. Eine Schleife mit Variabler, die hier abwärts zählt. Und dann ein Bremsen durch Motorlauf in Gegenrichtung um das Ding zum Stehen zu bekommen.

Stand : 21.12.2004

Schrittmotoren

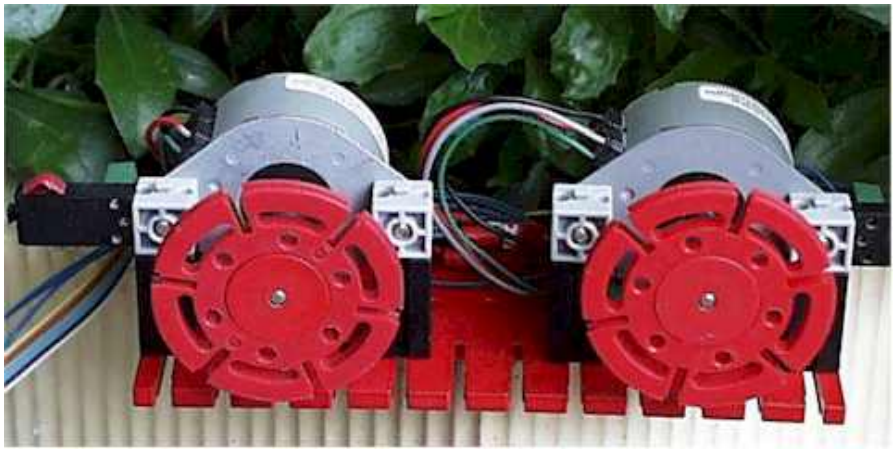
Der Betrieb von Schrittmotoren

Ist immer ein interessantes Thema. z.Zt. wird er vom ROBO Interface nicht direkt unterstützt. Deswegen hier eine kleine Bibliothek (im Stil der mitgelieferten Robot-Bib Position ES) mit Routinen zum Betrieb einzelner Schrittmotoren mit ROBO Pro. Das geht zwar etwas langsam, ist aber recht instruktiv. Ein Betrieb im X/Y-Verbund (Plot-Mode) scheint mir als ROBO Pro-Lösung doch etwas zu aufwendig. Da sollte man in den Kompatibiltätsmodus wechseln und z.B. mit C# und der Assembly FishFa30.DLL (Klasse FishStep) arbeiten.

Anmerkungen zum Schrittmotorbetrieb siehe auch auf den Detail-Seiten Schritt1 und Schritt2. Das dort genannte Zeitverhalten für das Intelligent Interface gilt in etwa auch für das ROBO Interface.

Die vorgestellte ROBO Pro-Lösung wurde mit ROBO Interface - USB - ROBO Pro und Intelligent Interface - COM - ROBO Pro getestet.

Das Modell

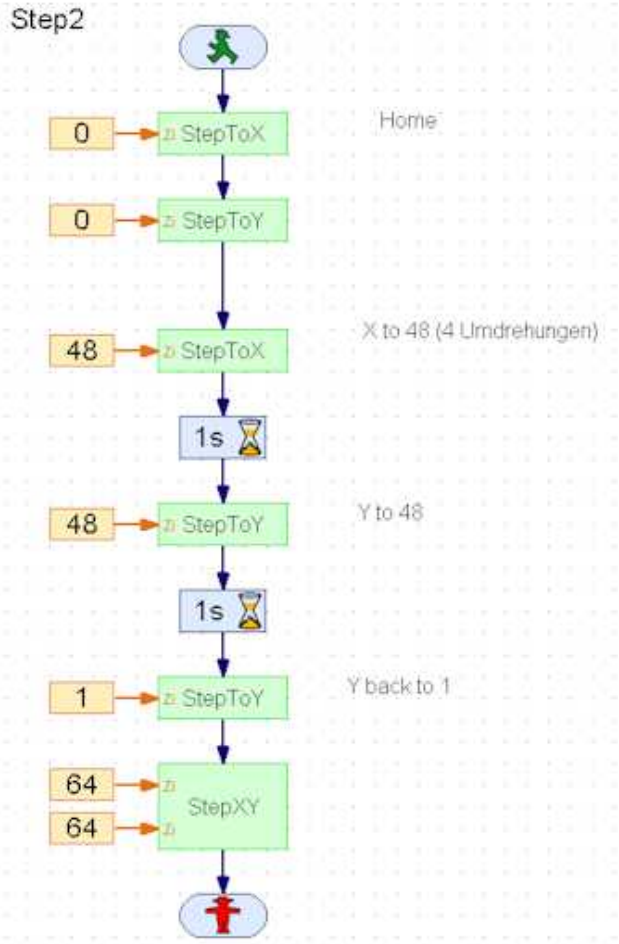


Das Modell ist sehr einfach : Motor links an M1/M2 mit Endtaster I1 und Motor rechts an M3/M4 mit Endtaster I3. Es ist ein reiner Versuchsaufbau. Beim Anfahren der Position 0 (Home) ist manuell der entsprechende Endtaster zu drücken.

Das Hauptprogramm

Ist sehr schlicht :

Der Reihe nach nur Unterprogrammaufrufe und ein paar Pausen, damit man den Zwischenstand geniessen kann :



Anfahren der Home-Position. Die Positionsangabe 0 hat eine Sonderstellung, es werden hier keine Schritte gezählt sondern stracks der zugehörenden Endtaster angefahren.

Anfahren der Position 48 (ab Endtaster) mit dem linken (X) Motor.

Anfahren der Position 48 mit dem rechten (Y) Motor

Zurück nach 1.

Simultanes Anfahren der Position X = 64 und Y = 64. Da StepXY nicht interpoliert (s.o.) wird hier erst eine 45° Schräge und dann eine Gerade gefahren. Das ist bei einem Plot-Betrieb nicht schön - nochmal : s.o.

StepRechts : Motorlauf nach rechts

Beschreibt dern Rechts-Zyklus des Schrittmotors :

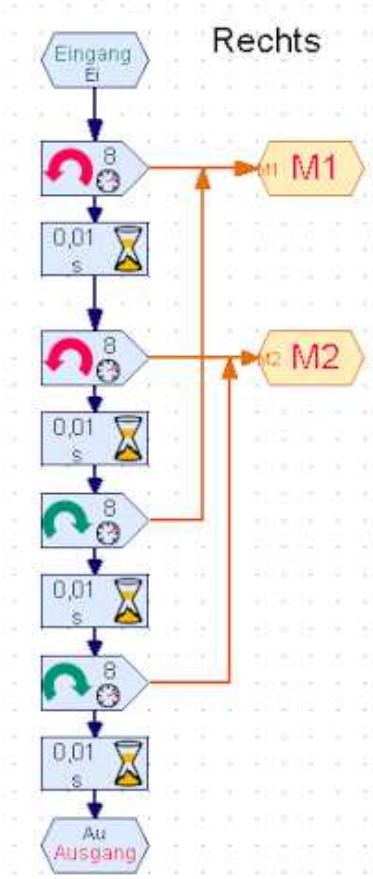
M1 : links, Pause 10 ms

M2 : links, Pause 10 ms

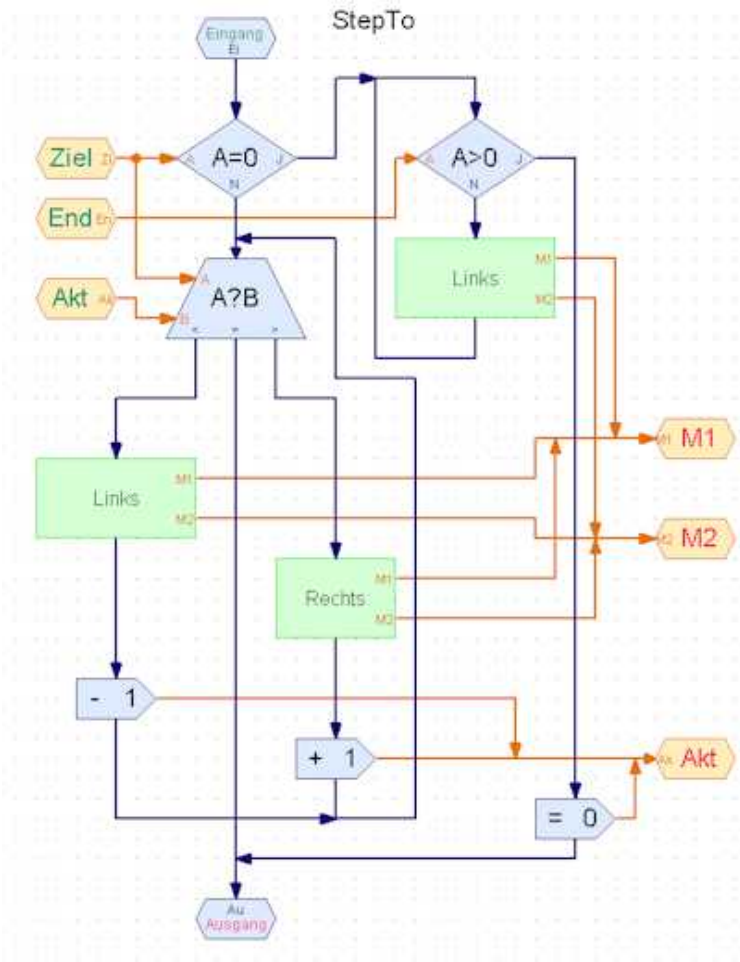
M1 : rechts, --

M2 : rechts

Die von fischertechnik bzw. Knobloch angebotenen Schrittmotoren machen pro Schaltung eine Drehung von 7.5°, ein Zyklus ist mit 30° demensprechend die kleinste ansteuerbare Einheit (12 Zyklen = eine Umdrehung).



StepTo : Anfahren einer vorgegebenen Position

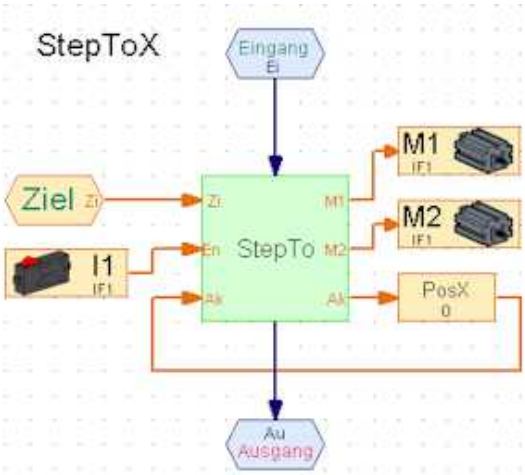


Entspricht weitgehend der Routine Pos aus der Bibliothek Position ES.

Die Impulstasterzählung wurde durch die Routinen Links und Rechts ersetzt.

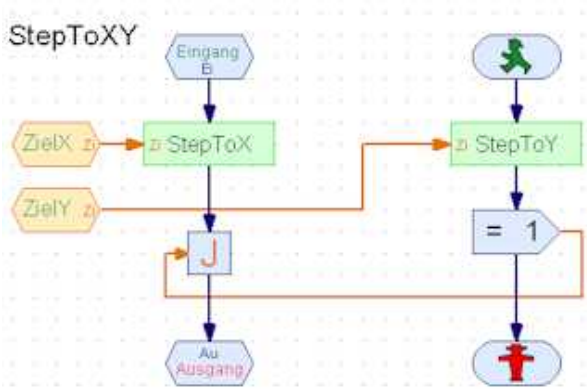
Das Links auf der rechten Seite ist für das Anfahren der Home-Position (Endtaster) verantwortlich.

StepToX : Anfahren einer vorgegeben X-Position



Um den praktischen Betrieb zu vereinfachen wurden hier Motoren, Endtaster und die globale Variable für die aktuelle Position fest zugeordnet.

StepToXY : Anfahren einer vorgegebenen X/Y-Position



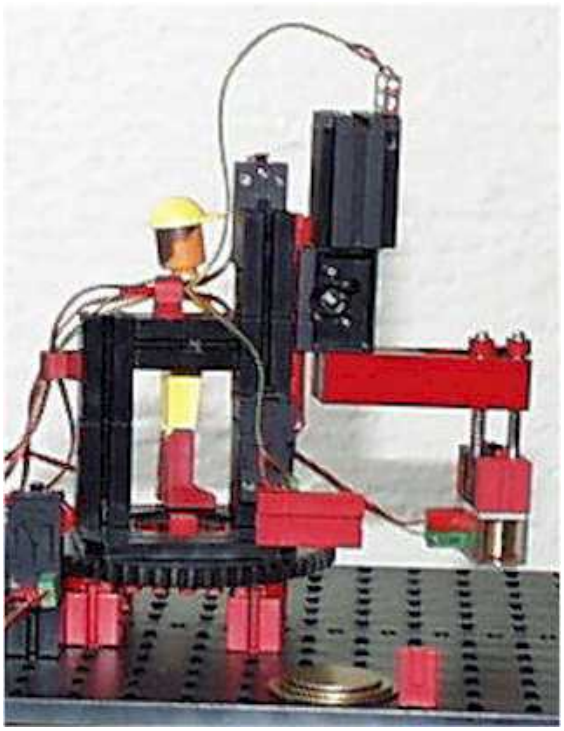
Hier die Routinen StepX und StepY im X/Y-Verbund. Y läuft dabei in einem eigenen Thread, der mit Aufruf von StepXY gestartet wird und sich selber beendet. Der X-Zweig wartet ggf. darauf.

Hinweis : Da die Motoren unabhängig voneinander laufen, kommt auch keine gerade Linie zwischen zwei vorgegebenen Punkten zustande. Dazu siehe die Einleitung oben.

Stand : 30.12.2004

Turm von Hanoi

Lösung für den Magnet Robot und ROBO Pro



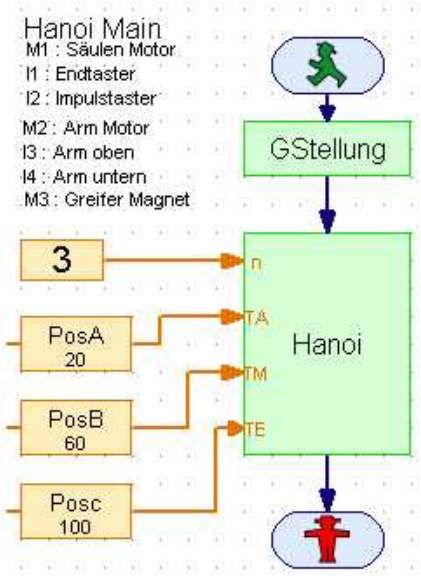
Allgemeines zum Roboter und der Programmlogik siehe Hanoi-Seite

Im Lieferumfang von ROBO Pro ist ein Beispiel für einen "Hanoi-Roboter" enthalten. Es basiert auf dem Säulen-Roboter der Industry Robots und stellt eine recht anspruchsvolle Lösung dar, die alle Komponenten des Robots nutzen.

Auf der Hanoi-Seite werden nun Lösungen für den oben abgebildeten deutlich einfacheren Roboter in den unterschiedlichsten Programmiersprachen gezeigt. Hier nun eine Lösung für ROBO Pro und das ROBO Interface. Ein Betrieb mit dem Intelligent Interface (im Online Modus) ist genauso möglich.

Die vorgestellte - graphische - Lösung entspricht im Aufbau und der Namensgebung den bisherigen textuellen Lösungen. Die interessantesten Routinen werden hier vorgestellt - einschl. einiger Anmerkungen für LLWin-Kenner -, das vollständige Programm ist zusammen mit den anderen Lösungen in HanoiRob.ZIP zu finden.
Zusätzlich wird ein installiertes ROBO Pro benötigt :

Das Hauptprogramm



Sieht vollkommen harmlos aus.

Mit GStellung wird auf die Grundstellung (PosA) gefahren.

Hanoi ist das (rekursive) Steuerprogramm für die Hanoilogik. Neu gegenüber ist hier die Möglichkeit ein Unterprogramm mit direkten Parametern aufzurufen.

- Konstante 3 : Anzahl der Scheiben

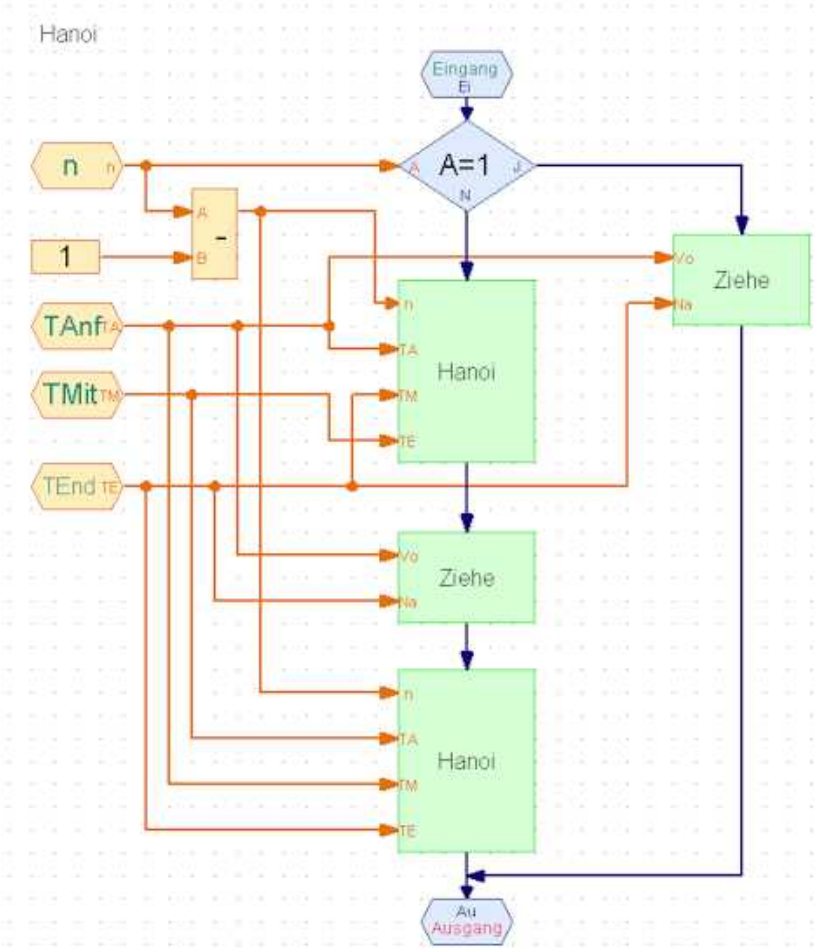
- Globale Variable PosA, PosB, PosC : Positionen der "Stapelplätze" in Anzahl Impulsen ab Endtaster. Die Variablen werden bei Programmstart mit den angegebenen Werten initialisiert. Konstanten wären hier ebenfalls möglich gewesen, so kann aber GStellung auf einfache Weise die gleiche Variable PosA nutzen um die Position des Ausgangsstapels anzufahren.

Das grüne und das rote Männchen dienen der Traditionspflege zu LLWin, das in der Nähe von Magdeburg entstand.

Die Hanoilogik

Kernpunkt der Hanoilogik ist das kunstvolle Vertauschen der jeweiligen Stapelpositionen auf jeder Rekursionsstufe. Der eigentliche Zug - das Umlegen der jeweils oberen Scheibe - fällt dann kaum noch auf. Diese Routine enspricht fast vollständig der Lösung, die ROBO Pro bei den Beispielen beiliegt. Unterschied : hier wird auf Rekursionsende n = 1 abgefragt (Ersparnis einer zusätzlichen Rekusionsstufe, dafür im Endezweig ein weiterer Ziehe-Aufruf), die ROBO Pro Lösung fragt auf n = 0 ab.

Besonders auffällig ist hier das vermehrte Auftreten der gelben Linien, die für den Datenfluß stehen. Die sargartigen Symbole an der linken Seite stehen für die Aufrufparameter (Ausgangsparameter sind ebenfalls möglich). Eingang und Ausgang sprechen für

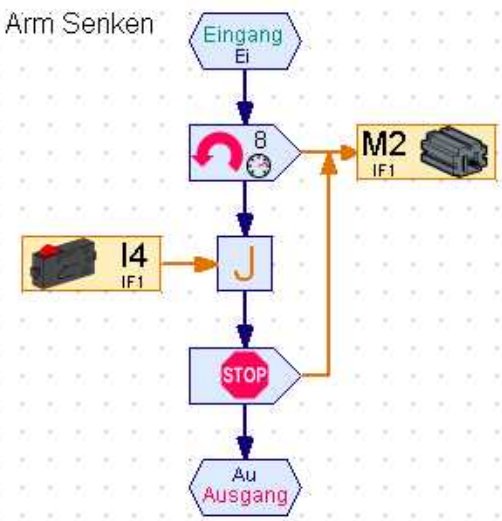


sich (es sind mehrere zulässig).

ROBO Pro bietet keine textuellen Ausdrücke (z.B. n - 1). Das wird hier auch graphisch über entsprechende Operatoren erledigt (gilt auch für logische Operationen).

Allgemein : ROBO Pro kennt ein Level Konzept, das die Komplexität der zur Verfügung stehenden Funktionen regelt. Level 1 und 2 bieten Funktionen, die weitgehend den bekannten Funktionen von LLWin entsprechen (Variablen sind hier aber unbekannt). Level 3 führt globale und lakale Variable ein und huldigt den gelben Linien.
Das Hanoi-Programm ist in Level 3 erstellt.

Zugriff auf das Interface



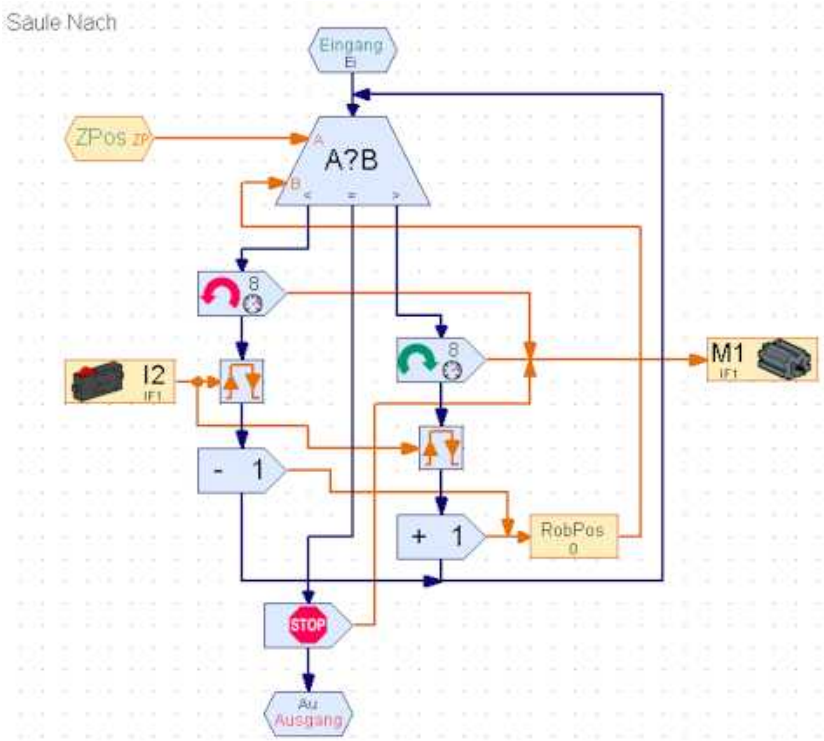
Die Routine ASenken tut nichts weiter als den Armmotor an einer Zahnstange herunterfahren bis der zugehörende Endtaster geschlossen ist.

Das Symbol mit dem roten Links-Kringel liefert eine Nachricht für einen Links-Dreh in Geschwindigkeitsstufe 8, die Nachricht geht an Motor M2 des ersten Interfaces (ROBO oder Intelligent Interface, je nach Voreinstellung).

Das J-Symbol im (schwarzen) Steuerfluß warte auf eine Nachricht (oder auch Signal), daß der Taster an I4 geschlossen wurde. Danach wird ein Stop-Signal an den Motor gesendet.

Die von LLWin bekannten Symbole (Parallelogramme und Pastillen) werden mit Level 1 eingeführt und können alternativ zu den hier angegebenen genutzt werden.

Ansteuerung einer Stapelposition



SNach steuert eine vorgegebene Position (Anzahl Impulse ab Endtaster) an.

Der Parameter ZPos enthält die ZielPosition, die globale Variable RobPos die aktuelle Position.

Im A?B-Symbol wird die Zielposition mit der aktuellen verglichen, bei Gleichheit ist das Ziel erreicht.

Ist die Zielposition größer als die aktuelle werden Impulse von I2 (gelbe Symbole) auf RobPos addiert (+1 Symbol) sonst subtrahiert. Der Motor an M1 wird in der entsprechenden Drehrichtung mit Fullspeed angesteuert (Linksdrehend : zum Endtaster).

Anmerkung : Zuweisungsoperationen (=, +, -) werden mit dem zugehörenden Wert gekoppelt angegeben (-1 entspricht RobPos -= 1; in C++ Schreibweise). Alternative können auch Werte über gelbe Linien von links durchgeleitet werden.

Die weiteren Routinen sind trivial und wiederholen nur die hier schon gezeigten ROBO Pro Elemente.

Stand : 11.12.2004

Turm von Hanoi

Über das Scheibenschichten

Ein Stapel von immer kleiner werdenden Scheiben soll über eine Zwischenstation auf einen neuen Platz gestapelt werden. Immer eine aufeinmal und stets die kleinere Scheibe auf die größere.

Die Lösung dieses Problems erfreut sich im Informatik-Unterricht großer Beliebtheit (bei den Lehrern). Es gibt deswegen auch immer wieder Lösungen zu diesem Problem. fischertechnik hat bereits im Kasten Computing von 1984 das Modell eines Hanoi-Roboters angeboten. Auf dieser Site finden sich ebenfalls einige Lösungen für das Problem :

- WinLogo Lösung für den Hanoi Robot von 1984
- VB6 Lösung für den Industry Robot von 1998
- VB6 Lösung für den LynxMotion Robot

Hier Lösungen für **VBA** mit vbaFish, **JScript** mit mscFish, **VB.NET**, **C#.NET**, **Python** und **Delphi**, alle nutzen FishFa30/umFish30.DLL :

Die manuelle Lösung

Das nachfolgende kleine Programm Hanoi.ftC (es wurde im PC Magazin Nr. 22 von 1988 als Quick Basic Lösung veröffentlich) gibt im Debug-Fenster eine Lösungsanweisung für das manuelle Umschichten aus :

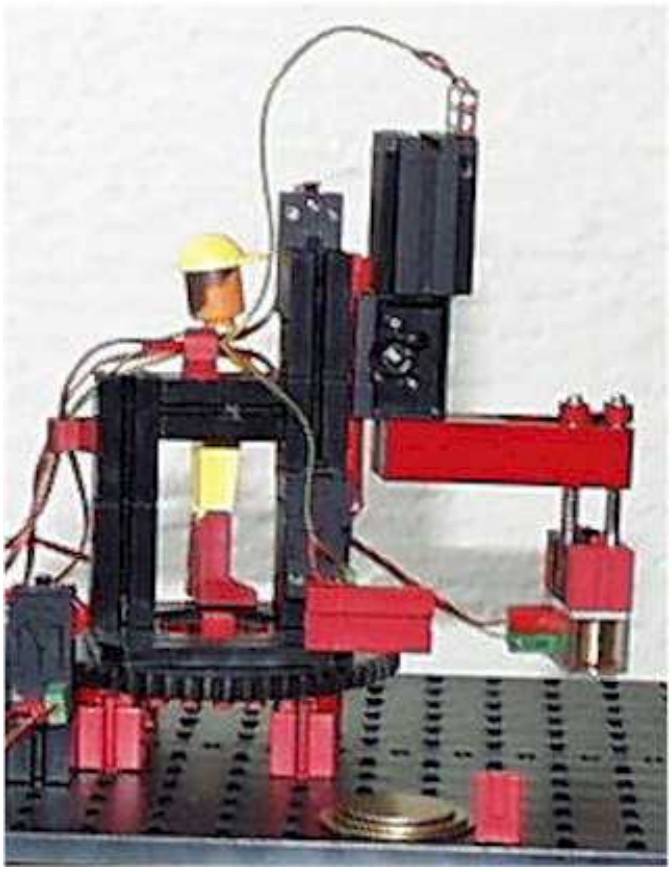
```
Sub Main
Dim n&
  n = InputBox("Anzahl Scheiben")
  Debug.Clear
  Debug.Print "Spiel mit" & n & " Scheiben"
  Hanoi n, "Links ", "Mitte ", "Rechts"
End Sub

Sub Hanoi(n&, TAnf$, TMit$, TEnd$)
  If n = 1 Then
    Debug.Print "Scheibe von " & TAnf & " -> " & TEnd
  Else
    Hanoi n-1, TAnf, TEnd, TMit
    Debug.Print "Scheibe von " & TAnf & " -> " & TEnd
    Hanoi n-1, TMit, TAnf, TEnd
  End If
End Sub
```

Kern ist das Unterprogramm Hanoi, das aus einer einzigen If-Anweisung besteht. Wenn die Anzahl der Scheiben auf dem Stapel bei 1 angekommen ist, wird direkt vom aktuellen Quellstapel zum Zielstapel geschichtet, sonst wird dreimal hin und her geschichtet. Trick dabei ist der rekursive Aufruf von Hanoi (das Unterprogramm ruft sich selber auf).
Achtung : Zur Anzeige der Lösung ist im Menü Ansicht die Option 'Immer teilen' einzustellen

Die Anweisung Debug.Print "Scheibe von " & TAnf & " -> " & TEnd, die zweimal im Programm auftaucht, ist nun durch eine Anweisung Ziehe zu ersetzen, die einen Robot überredet tätig zu werden, ganz wie im manuellen Verfahren. Dazu ist dann erstmal ein Robot erforderlich :

Der Hanoi Robot



Der Hanoi Robot ist ein Eigenbau der auf dem Schweißroboter-Chassis des Computing Starter Kits basiert. Er entspricht hier weitgehend dem Hanoi Robot von 1984. Allerdings wurde die Technik modernisiert :

M1 : Säulenmotor mit Impulsrad und Endtaster an E1, Impulstaster an E2

M2 : Armmotor mit Endtaster oben E3 und unten E4

M3 : Gefedert gelagerter Magnet (neuere Version bei Knobloch : 32363 - 17,20 Euro)

Gestapelt werden die Eisenscheiben, die im Bild vorn (PosA) neben dem Winkel liegen. Verdrahtet wurde "fliegend".

Die Routinen

-------	Hanoi-Logik
Ziehe(Von, Nach)	Ersatz für die manuelle Arbeitsanweisung : Zerlegt in Hole(Von) und Bringe(Nach)
Bringe(Pos)	Ablegen einer Scheibe auf der gewünschten Zielposition : SauleNach(Pos), ArmSenken, ScheibeGreifen, ArmHeben
Hole(Pos)	Aufgreifen einer Scheibe von der angegebenen Quellposition SauleNach(Pos), ArmSenken, ScheibeLegen, ArmHeben
-----	Robot-Ansteuerung
SauleNach(ZielPos)	Fahren auf die Zielposition. Die Position wird durch eine Impulsrad bestimmt
ArmSenken	Senken des Greiferarms bis zum Endtaster
ArmHeben	Heben des Greiferarms bis zum Endtaster
ScheibeLegen	Einfach : Magnet aus
ScheibeGreifen	Auch einfach : Magnet ein
-----	DrumRum
Grundstellung	Fahren auf Grundstellung (Robot auf PosA, Armhoch, Magnet aus)
Init	Besetzen der globalen Variablen. Dabei auch festlegen der Werte für die drei Stapelpositionen PosA, PosB und PosC

Die Sources

Das komplette VBA Programm gibts gleich zweimal :

- HanoiRobot.ftC : Eine all-in-one Lösung. Alle beschriebenen Routinen sind in einer Source enthalten
- HanoiMain.ftC/HanoiRobot.CLS : Die Robot-Ansteuerung wurde in eine separate Klasse ausgelagert. Hier wohl kaum erforderlich. Mehr um zu zeigen, wie man mit vbaFish auch Klassen einsetzen kann. Die Programme sind funktional gleich

eine Perl-Lösung :

- Hanoi.PL : Ein File mit dem kompletten Programm.

dann gleich noch einmal für Python :

- HanoiRobot.PY : Ein File, aber wie bei HanoiMain/HanoiRobot wurde die Robot-Ansteuerung in eine Klasse ausgelagert. Getestet wurde mit PythonWin. Ablauf auch als rreine

Konsolapplikation.

für Delphi :

- poorHanoiRob.DPR : Eine all-in-one Lösung auf Basis des Templates poorFish30. Alle beschriebenen Routinen sind in einer Source enthalten, Konsolprogramm.

und für JScript :

- HanoiRob.JS : Eine Lösung auf Basis von Unterprogrammen (function)
- HanoiRobC.JS : Eine Lösungsvariante unter Verwendung eines Objects (Klasse), das die Robot-Methoden kapselt.

und dann noch VB.NET und C#.NET:

- HanoiRobot : Eine Windows Lösung unter Nutzung der Klasse HanoiRob.

und jetzt auch noch eine ROBO Pro-Lösung

Alles zusammengefaßt in HanoiRob.ZIP. Zusätzlich ist noch vbaFish30Setup.EXE (VBA) bzw. PythonFish30.ZIP (Python), vbFish30Setup.EXE (Perl), umFish30.ZIP (C#, VB.NET) oder delphiFish30Setup.EXE (Delphi) erforderlich (Auf der jeweiligen Seite wird das genauer erklärt). Außerdem die zugehörenden Entwicklungssysteme - Ausnahme : vbaFish30, da ist schon alles dabei.

Stand : 11.12.2004