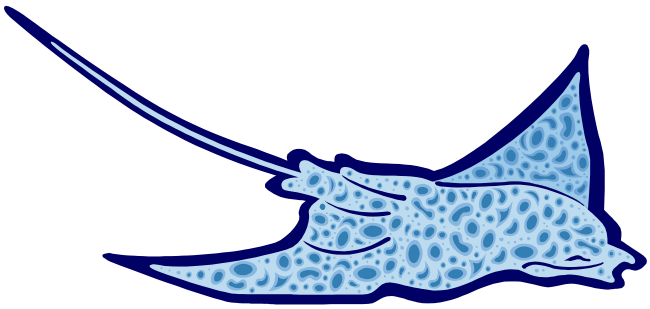
ftComputing

ROBO Pro : Tutorial

2. überarbeitete Auflage

Ulrich Müller



Inhaltsverzeichnis

Tips & Tricks

Allgemeine Hinweise

Version

Die erste stabile Version von ROBO Pro war 1.1.2.26C mit der die im Katalog angebotenen Katalog-Kästen ausgeliefert wurden, die aktuelle Version ist 1.1.2.40 (Sep. 05).

ROBO Pro ist eine vollständige Neuentwicklung und deswegen noch in Bewegung. Bei der Erstellung des Tutorial wurde mit Version 1.1.2.40 gearbeitet. Updates von der gelieferten CD ROM auf diesen oder einen höheren Stand können jederzeit von der fischertechnik Site geladen werden.

Die erste Auflage des Tutorial wurde auf Basis von Version 26C erstellt. Dies ist eine Überarbeitung des Tutorials besonders im Bereich der Listenverarbeitung, die in Version 26C noch gravierende Schwächen hatte.

Es wird durchgängig Level 3 von ROBO Pro verwendet auch wenn es im Einzelfall vielleicht nicht erforderlich ist.

Sinn

Der Schwerpunkt dieses Tutorials liegt nicht so sehr auf fertigen Programmen, sondern in der Einführung in den Umgang mit ROBO Pro. Das Tutorial legt den Schwerpunkt auf den Bereich Industry Robots. Weitere Programm Beispiele finden sich bei den genannten Beipielen und – wieder mit Erläuterungen – in der ROBO Pro-Ecke von ftComputing.de (siehe Links).

Das Tutorial will nicht das ROBO Pro Handbuch bzw. die Hilfe-Datei ersetzen, sondern im praktischen Teil ergänzen. Ebenso sollten die Anleitung der Computing Kästen durchgearbeitet werden.

Links

Die RPP-Files für die Beispiel sind in www.ftcomputing.de/zip/robopro.zip zu finden.

Die zugehörende ROBO Pro-Ecke ist unter www.ftcomputing.de/robopro.htm zu finden.

Die fischertechnik Seite www.fischertechnk.de

Und die Update Seite für ROBO Pro : http://www.fischertechnik.de/robopro/update.html

Über den Rechtsklick

auf ein Element in der Funktionseite der ROBO Pro IDE kommt man meist auf eine Eigenschaftsseite auf der man die Eigenschaften des betreffenden Elements einstellen kann. Das geht so weit, daß man auch die Haupteigenschaft von im linken Fenster noch eigenständigen Elementen ändern kann : z.B. kann dann aus Warten auf "J" ein Impulszählen werden.

Betrieb eines oder mehrerer Interfaces

Der Anschlußart Haupt(Default)-Interfaces (ROBO Interface bzw. Intelligent Interface, Simulation) kann über das Icon COM – USB festgelegt werden.

Programmspezifisch können auf der Eigenschaftsseite eines Programmes die dort genutzten Interfaces (Haupt-Interfaces, Extension ...) festgelegt werden.

Auf der Form Interface-Test können auf der Seite Info die Eigenschaften des aktuellen Interfaces angezeigt und verändert werden.

Hinweise

Namen

Sind CaseSensitive. D.h. es wird zwischen Groß- und Kleinschreibung einzelner Buchstaben eines Namens unterschieden. NaMe ist etwas anderes als nAmE.

Namenräume

Globale Variable, Lokale Variable, Unterprogramme und Bedienelemente haben eigene Namensräume. D.h. eine Globale Variable kann den gleichen Namen haben wie ein Unterprogramm. Die Namensgleichheit von Anzeige Elemente und Variable kann sehr sinnvoll sein.

Globale Variable

Globale Variable Variable gelten im gesamten Programm. Eine globale Variable kann mehrfach im Programm vorkommen (Variablen Element mit exakt gleichem Namen), sie repräsentiert immer nur einen Speicherplatz. Die Initwert des zuletzt eingefügten bzw. geänderten Variablen Elements überschreibt alle anderen Angaben.

Lokale Variable

Lokale Variable gelten nur in dem Unterprogramm, in dem sie angelegt wurden. Sie können im gleichen Unterprogramm mehrfach auftreten (Option Namensgebunden). Sie werden bei Eintritt in das Unterprogramm initialisiert und bei Verlassen wieder aufgelöst. Lokale Variable sind im Hauptprogramm nicht zulässig.

Listen

In den Beispielprogrammen werden, wenn erforderlich, Listen mit Konstantenwerten verwendet. Meist existieren die gleichen Werte auch in Form von Dateien, die über das Menü Datei | CSV ... bei Bedarf manuell geladen werden können, ebenso können die Listenwerte über das gleiche Menü nach Programmende manuell abgespeichert werden.

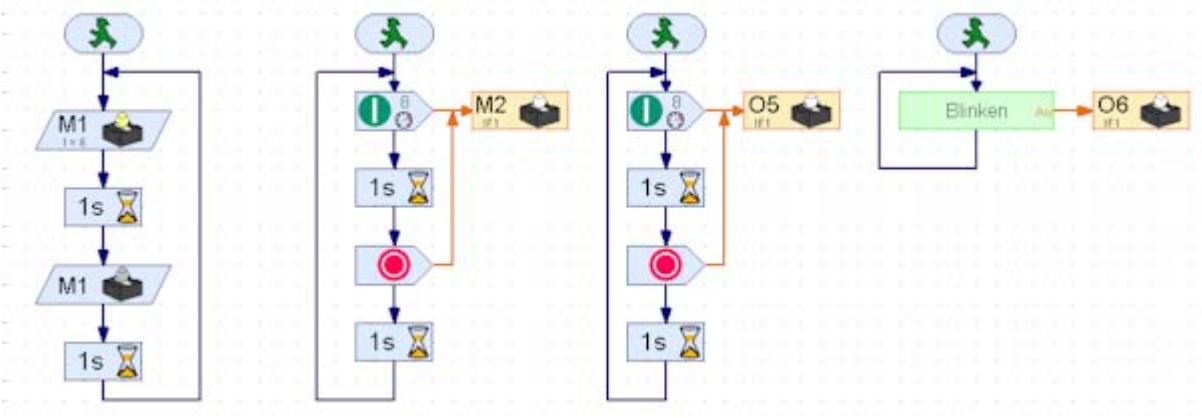
Die Dateinamen können auch fest in die Eigenschaftsseiten der jeweiligen Liste eingetragen werden, dann werden sie automatisch geladen bzw. gespeichert. Allerdings ist dann meist eine feste Pfadangabe erforderlich (Default : ..\Programme\ROBOPro).

Wenn mehrere Listen in einer Datei gespeichert werden, müssen die zusammengehörenden (in eine Zeile gehörenden) Werte durch einen "Separator" getrennt werden, das ist per Default ein "," (Komma), ein Semikolon bzw. Tab ist möglich. Bei Verwendung von Excel zur Be- und Verarbeitung von Listen ist das mit den Excel-Einstellungen abzustimmen.

Alle Listen sind als global definiert.

Techniken

Blinker : Schleifen/Unterprogramme/Ausgänge 1



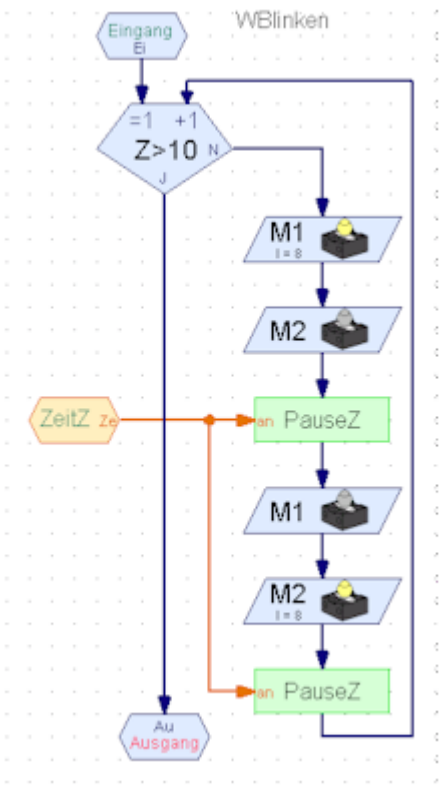
Angeschlossen ist jeweils eine Lampe an einen M-Ausgang bzw. einen O-Ausgang und Masse.

Im linken Diagramm wird Level 1 verwendet, in den anderen Level 3. Bei Level 1 stehen nur die Grundelemente zur Verfügung, bei Level 2 zusätzlich Unterprogramme (ohne Parameter) und bei Level 3 Variable und die gelben Datenfluß-Linien, zusätzlich eine ganze Anzahl weiterer Funktionen. Nach einer Einarbeitungungphase wird man allgemein sicher in Level 3 arbeiteten. Hier wird von einer Level 3-Einstellung ausgegangen. Die Grundelemente können in bunter Mischung mit Elementen höherer Level genutzt werden, das ist in erster Linien eine Frage der Bequemlichkeit (und dann auch noch eine der Weltanschauung). Hier werden bevorzugt Elemente von Level 3 verwendet, da dort die Zuordnung zu den einzelnen Ein- und Ausgänge einfacher zu handhaben ist. Bei einfachen Unterprogrammen werden gelegentlich auch Level 1 Elemente eingesetzt (sie sind bequemer).

Das obige Programm ist ein einzelnes Hauptprogramm in dem es ums Blinken geht, von links :

1. Lampe an M1 1 Sekunde an, 1 Sekunde aus, Betrieb in einer Endlosschleife, Abbruch über Bedieneroberfläche. Nur Grundelemente
2. Lampe an M2, wie (1.) aber mit Elementen von Level 3. Hier gibt es einen Ein- und einen Aus-Befehl, der über eine gelbe Linie das entsprechende Kommando an einen Ausgangs-Baustein sendet. Der Ausgangsbaustein sollte nur einmal im Haupt-/Unterprogramm auftauchen, kann aus Gründen der Übersicht (Strippenverhau) auch mehrfach angegeben werden, gemeint ist immer der gleiche Interface Ausgang.
3. Lampe an O5, wie(2.)
4. Lampe an O6, funktional wie (3.), jedoch wird das Blinken in ein Unterprogramm verlegt und der zu schaltende Ausgang als Parameter übergeben.

WechselBlinker : Schleifen/Unterprogramme 2

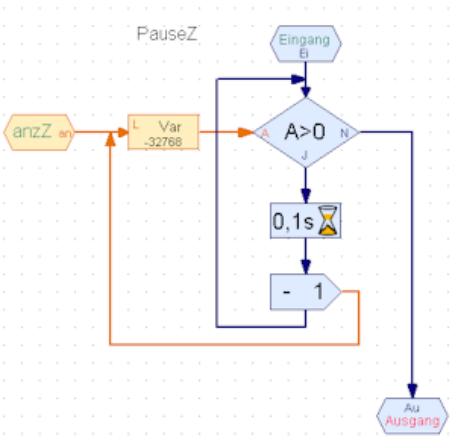


Wenn man das Blinken nicht als einzigen Zweck des Programms sieht, so ist es besser, das Blinken samt Schleife in ein Unterprogramm zu legen.

Hier werden zwei an M1 und M2 angeschlossene Lampen mit Level 1 Befehlen angesprochen.

Die Blinkschleife wird über eine Zählschleife (Level 1) gesteuert und 10mal durchlaufen. Dazu wird der Schleifenzähler bei Eintritt in das Unterprogramm auf den Wert 1 gesetzt und bei jedem Schleifenende um 1 erhöht und mit dem Maximalwert 10 verglichen.

Da die Standardpause nur einen festen Wert zuläßt, wurde in ein Unterprogramm verlegt, das eine Standardpause von 0,1 Sek. n mal durchläuft. Der Wert n wird dem Unterprogramm WBlinken beim Aufruf übergeben. Das Unterprogramm reicht es dann an PauseZ weiter.

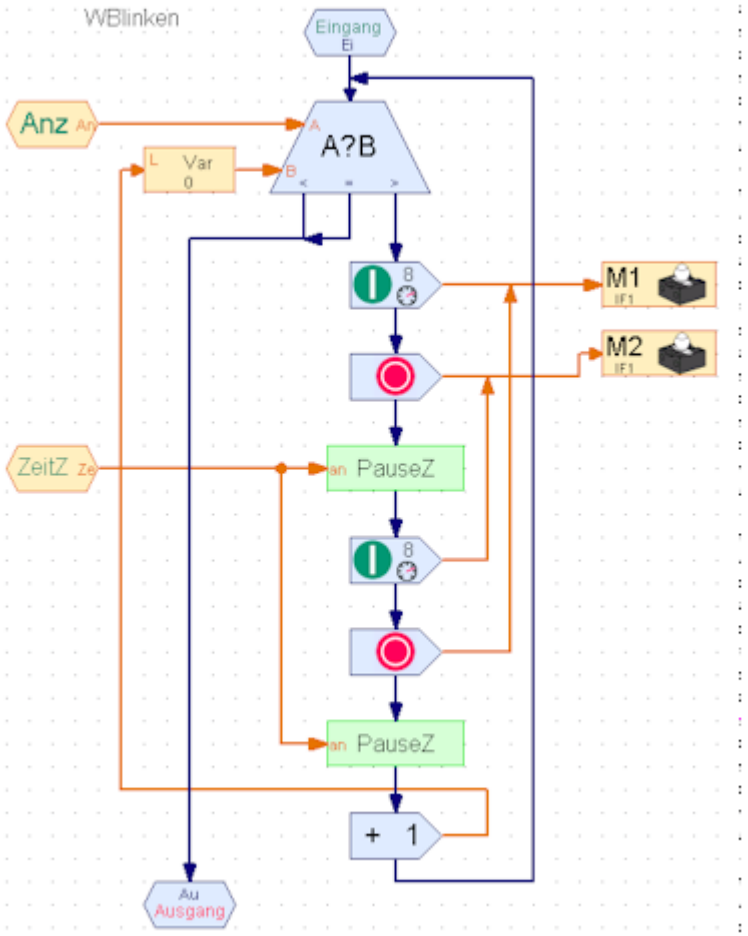


Hier eine Schleife mit einer variablen Anzahl von Durchläufen. Die Anzahl der Durchläufe wird durch einen Parameter übergeben und in einer lokalen Variablen gespeichert und vor Ausführung der Pause mit 0 verglichen.Nach Pause wird der Zähler um 1 heruntergezählt.

Die lokale Variable Var wird beim Eintritt in das Unterpramm angelegt und bei dessen Verlassen wieder aufgegeben, sie kann nicht von außen zugegriffen werden.

Das Hauptprogramm muß dann das Unterprogramm WBlinken nur noch mit einer passenden Blinkfrequenz aufrufen. Das geschieht am besten über eine Konstante

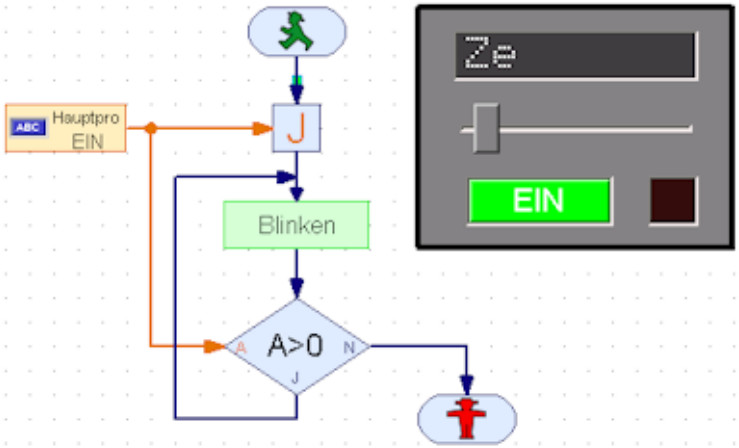
WechselBlinker : Schleifen/Unterprogramme 3

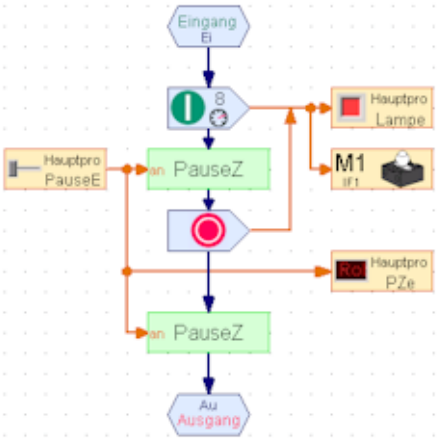


Das WechselBlinken wurde hier mit einem zusätzlichen Parameter für die Anzahl der Blinkdurchläufe ausgestattet. Hier werden im Befehl A?B der Parameter Anz mit dem Wert der lokalen Variablen Var (Schleifenzähler) verglichen, der bei Eintritt in das Programm auf 0 gesetzt wird. A?B hat drei Ausgänge von denen zwei zusammengelegt wurden. Bei Anz <= Var wird das Programm also verlassen. Diese Abfrage hätte auch mit dem A?0 von PauseZ realisiert werden können, da Anz auch auf 0 herunter-gezählt werden könnte.

In diesem Beispiel wurden Level 3 Befehle zum Schalten der Lampen verwendet.

Bedienung : Bedienfeldelemente





Die Bedienfeldelemente traulich vereint zur Anzeige und Steuerung eines simplen Blinkers. Sie wurden auf einer grauen, rechteckigen Form (Abschnitt Zeichnen) plaziert, das Rechteck muß dazu in den Hintergrund gestellt werden (Menü Zeichnen | Objekt in den Hintergrund). Die Bedienelemente wurden im Funktionsbereich der Hauptform untergebracht, da sieht dort zusammen mit dem Hauptprogramm im Blickfeld sind (vorgesehen ist eigentlich Tab Bedienfeld).

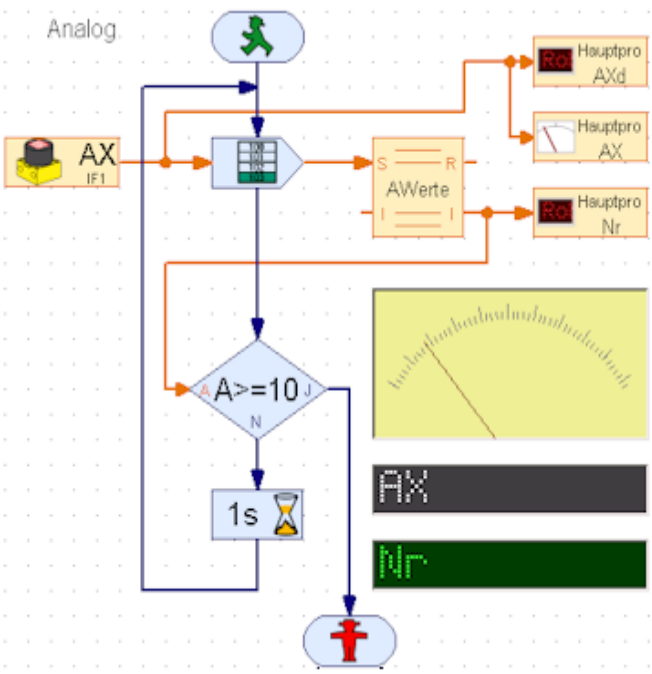
Parallel zu den Bedienelementen gehen entsprechende Ein- und Ausgabe-Elemente. Hier sind das EIN für den EIN-Button, PauseE für den Scrollbar zur Steuerung der Blinkfrequenz, Lampe zur Anzeige des Lampenstatus und PZe zur Anzeige der eingestellten Pausenzeit.

Wie man sieht können die Ein- und Ausgänge munter im ganzen Programm verteilt sein und zentral angezeigt werden.

Da die Handhabung des Buttons ein wenig hakelig ist (der letzte Status wird gespeichert), wird er im Hauptprogramm gleich zweimal abgefragt, einmal zum Starten des Blinkens und einmal zu dessen Beenden. Man sieht, er kann wie ein Taster an einem I-Eingang gehandhabt werden. "J" ist ein Befehl, der auf Eingang = 1 (true) wartet.

Im Unterprogramm Blinken wird der Scrollbar (PauseE) gleich dreimal angezapft, zur Bestimmung der Pausenzeiten beim Aufruf von Blinken und zu deren Anzeige in PZe.

Analog : Füllen einer Liste, Analoganzeige



Analog liest im Sekundenabstand den aktuellen Wert des AX-Einganges, an den ein Photowiderstand angeschlossen ist, aus. Er wird auf dem Analog-instrument und zusätzlich noch digital angezeigt. Zusätzlich wird die aktuelle Listengröße angezeigt.

Das Fülllen der Liste AWerte geschieht über den Befehl "Wert Anhängen" am S-Eingang der Liste. Die Analog-Werte werden dazu über den Dateneingang (RechtsKlick – Haken) bereitgestellt. Zu Beginn des Programms ist die Liste in diesem Fall leer und für max. 100 Werte ausgelegt(default), sie könnte mit Werten vorbelegt werden (RechtsKlick auf Liste, im Dialog einfügen). Ebenso kann die max. Größe geändert werden.

Bei jeder Veränderung der Liste über den S-Eingang steht am I-Ausgang die neue Anzahl der Listenelemente zur Verfügung (auch beim Programmstart). Wird die max. mögliche Anzahl (default = 100) überschritten, wird 0 ausgegeben. Hier werden exakt 10 Werte eingegeben (A>=10), die in der Liste mit 0 – 9 durchnumeriert sind. Damit man auch etwas davon hat : Immer 1 Sek. Pause.

Liste AWerte

Maximal 100 Werte, beim Programmstart leer.

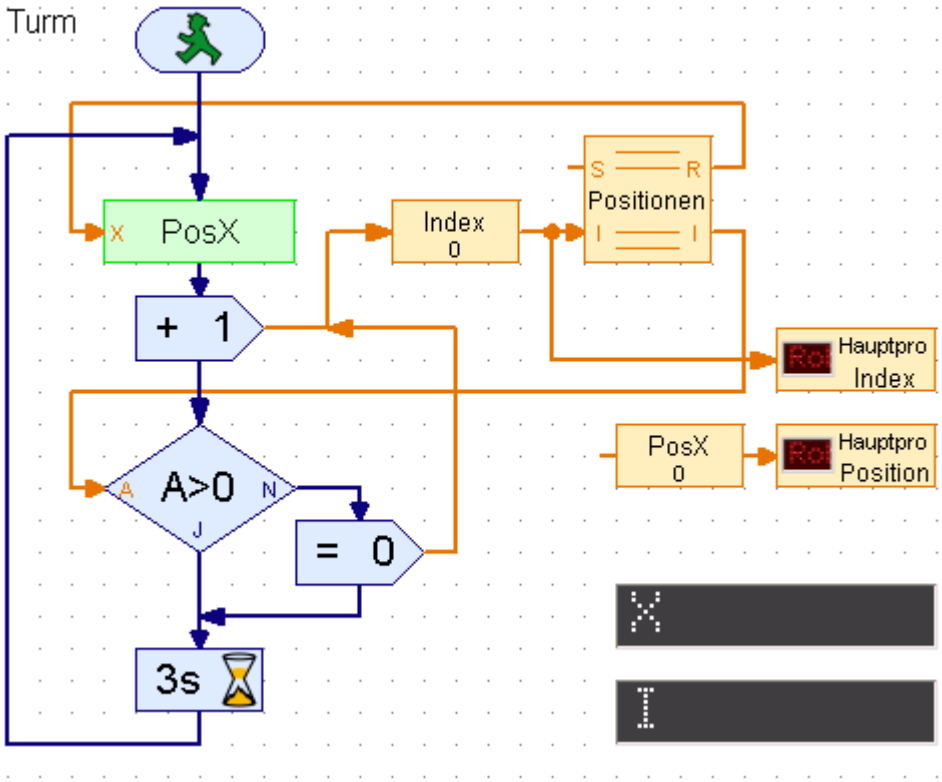
Bei Programmende sollen die gemessenen Werte in eine Datei Analog.CSV gespeichert werden. Sie können dann anschließend mit z.B. Excel ausgewertet werden. Dazu müssen in dem Kasten "In .CSV-Datei schreiben" Einträge gemacht werden (alternativ) :

1. Haken an "In CSV Speichern", die "1" in der Spalte darüber bleibt (Erklärung bei den Robots) zusätzlich Spaltenname angeben.
2. Oder in dem linken Textfeld einen Dateinamen angeben (Wenn kein Pfadname angegeben wird, landet die Datei im Verzeichnis Programme\ROBOPro). "1" und Spaltenname wie gehabt.

Im Fall 2. wird bei Programmende (Ende-Form) automatisch in das anzugegebende File gespeichert.

Im Fall 1. stehen die Daten in einem speziellen CSV-Speicher zum Speichern bereit, sie können dann manuell über das Datei-Menü CSV-Speichern in ein wählbares File gespeichert werden.

Turm : Zugriff auf den Listeninhalt



Turm bewegt die Säule eines Industry Robots (Rob 2, 3, 4) oder Computing Starter (SchweißRobot) in einer Endlosschleife von Position 0 nach 3. Dazu wurde die Liste Postionen mit festen Werten (RechtsKlick, Tabelle) gefüllt (0, 45, 150, 30). Das Bewegen der Säule geschieht über die Bibliotheks-Routine PosX / Pos.

Über die Variable Index wird der aktuelle Positions-Wert ausgewählt und über den R-Ausgang der Liste als Parameter an PosX übergeben. Am R-Ausgang steht immer dann eine neuer Wert an, wenn sich der I-Eingang (hier über die Var Index) verändert hat. Das gilt auch für den Programmstart. Es steht dann der zum Init-Wert von Index (0) gehörende Wert (0) am R-Ausgang an. Deswegen erfolgt die Inkrementierung von Index erst nach der ersten Verarbeitung eines R-Wertes.

Da bei PosX die Position 0 einen Sonderfall bildet (Anfahren der Home-Position), kann hier auf das eigentliche fällige Home verzichtet werden.

Nach dem Inkrementieren von Index wird auf A>0 abgefragt, um festzustellen, ob noch Listenwerte vorliegen. Wenn nicht wird 0 zurückgegeben (Der R-Ausgang enthält dann –32767, also Achtung).

Für den Endlos-Betrieb wird der Index jetzt wieder auf 0 gesetzt (beim Programmstart geschah das automatisch) und sich in die Schleife über die Positionen eingeklinkt. Dort wird immer nach Erreichen eine Position 3 Sek. geruht, sonst kann man das Spektakel nicht richtig genießen.

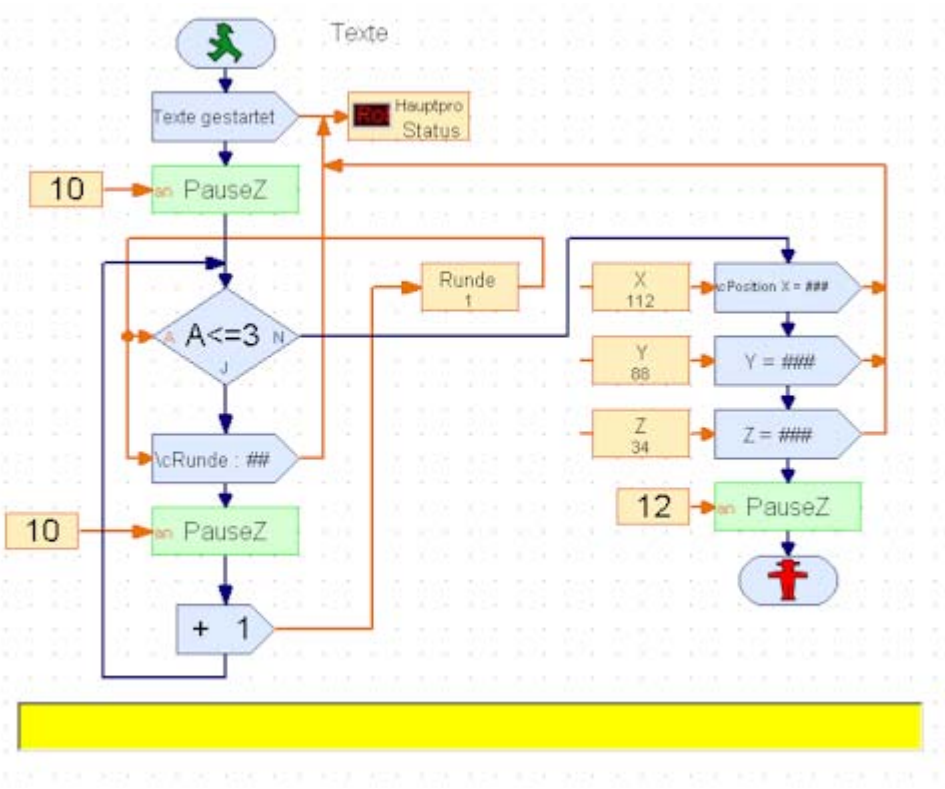
Hinweis : man sollte Füllen / Zugriff einer Liste in getrennten Routinen erledigen, da es sonst leicht bei der Belegung der Listen-Ein- und –Ausgänge zu Konfusionen kommen kann.

Liste Positionen

In der Liste Positionen sind die von Turm anzusteuernden Positionen enthalten. Das sind konstante Werte in "Liste der Anfangswerte" der entsprechenden Eigenschaftsseite. Alternativ können sie auch aus einer CSV-Datei gelesen werden. CSV-Dateien kann man mit Excel, aber auch einfach mit NotePad (Editor), erstellen bzw. ändern.

Ebenso kann man die Werte der Eigenschaftsseite manuell editieren und dann bei Programmende manuell, oder bei Eintrag eines Dateinamens automatisch, in eine Datei abspeichern.

Texte : Wie sage ich es dem Anzeige Element



Mit dem Befehl Texte können angezeigt werden :

-- reine Texte "\cTexte gestartet", Anzeige Element wird vorher gelöscht
ohne \c : Text an Inhalt Anzeige Element nahtlos anfügen
– Text und formatierte Zahlen "\cRunde : ##"
– mehrere Teiltexte in ein Anzeige Element "\cPosition X = ### Y = ### Z = ###"

Industry Robots

Allgemeines

Übliche Zuordnung der Motoren / M-Ausgänge und Taster / I-Eingänge

X-Achse (Säule) : M1, I1 (Endtaster), I2 (Impulstaster)
Y-Achse (Arm Horizontal) : M2, I3, I4
Z-Achse (Arm Vertikal) : M3, I5, I6
4-Achse (Greifer) : M4, I7 (Endtaster, Greifer Offen), I8 (Impulstaster)

Der Impulstaster ist ein besonderes (vierZahn) Rädchen, das auf einer Motorwelle sitzt. Es schaltet einen an einem I-Eingang angeschlossenen Taster dessen Wechsel von 0 auf 1 und umgekehrt gezählt und zur Positionsbestimmung herangezogen werden können.

Arbeitsraum

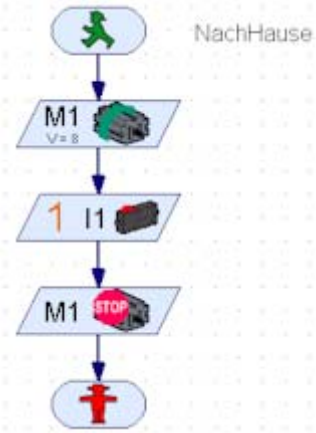
Der Arbeitsraum eine Industry Robots wird in Impulsen ab Endtaster gezählt. Er liegt zwischen ca. 90 – 230, je nach Komponente und Kabelführung.

Um einen Bezugspunkt zu haben, muß der Robot mindestens einmal beim Start des Programmes die zugehörenden Endtaster anfahren, das geschieht **linksdrehend** (TestPanel : Klick auf links). Tut er das nicht, so muß der entsprechende Motor umgepolt werden.

Bei Erreichen der Endtaster sollten die zugehörenden (globalen) PositionsVariablen auf 0 gesetzt werden. Der umgekehrte Fall (Setzen auf maximalen Arbeitsraum und abarbeiten auf 0) ist eigentlich technisch besser, aber wohl nicht üblich, hier wird von Home gleich 0 ausgegangen.

TurmSolo

NachHause : Fahren bis zum Anschlag



Die Robots nutzen zur Bestimmung der aktuellen Position Impulsrädchen, die auf dem Getriebe des Antriebsmotors der jeweiligen Komponente sitzen und einen Taster betätigen. Das Impulsrädchen hat vier Nocken und - dazu gehörend – vier Nockentäler. Gibt pro Umdrehung acht Impulse. Die aktuelle Position wird in Anzahl Impulse ab Null angegeben. Null ist die Position in der der zugehörende Endtaster bei Linksdrehung der Komponente betätigt wird. Bei der Säule wird der Motor an M1 angeschlossen, das Impulsrädchen an E2 und der Endtaster an E1.

Bevor die Position so bestimmt werden kann, muß die Komponente (die Säule) erstmal in die Nullposition gebracht werden. Das geschieht mit dem kleinen Programm oben :

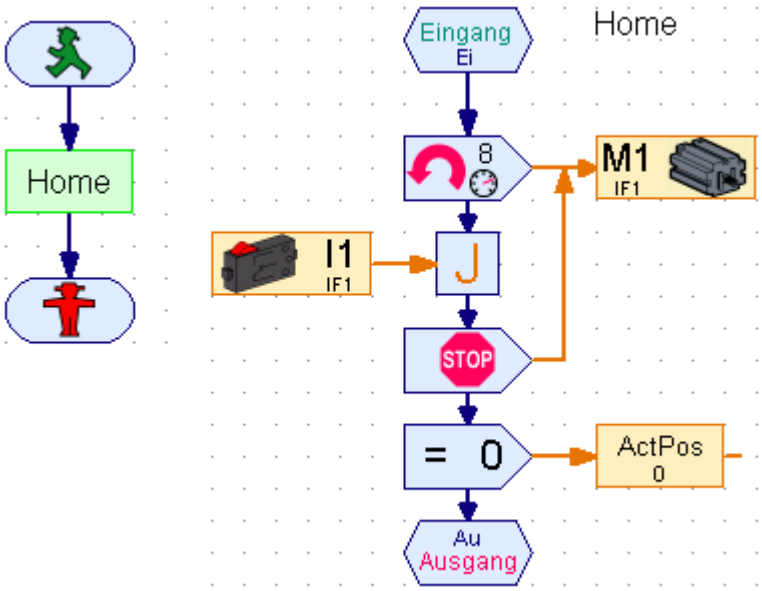
Motor an M1 linkdrehend einschalten

Mit "Warten auf Eingang" das Erreichen von I1 abwarten(I1 = 1).

Motor M1 wieder abschalten.

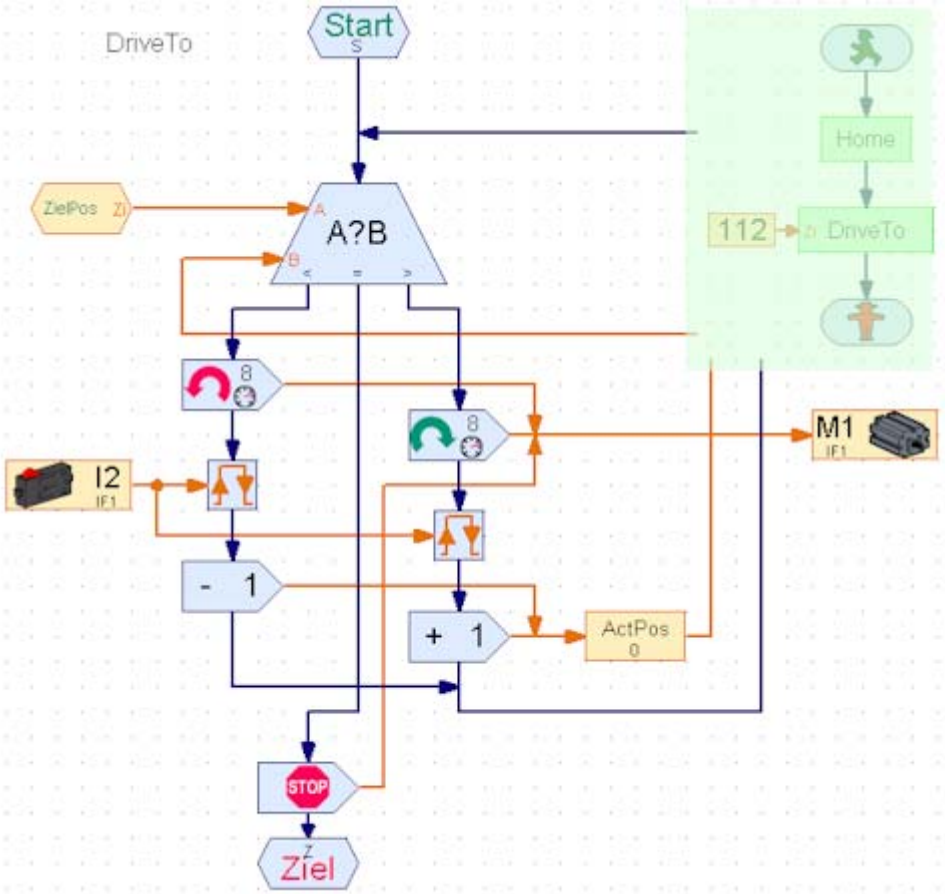
Und dann müßte noch eine globale Variable (ActPos) auf 0 gesetzt werden, wenn man denn noch mehr machen möchte.

Home : Anfahren Home in einer Subroutine



Die Funktion "Anfahren Home" wurde in ein Unterprogramm verlegt, außerdem wurden Level 3 Befehle verwendet. Hinzu kommt die globale Variable ActPos. Bei größeren Programmen wird die Angelegenheit so deutlich übersichtlicher. Alternative : Nutzung der Bibliotheks-Routine PosX (siehe dort).

DriveTo : Anfahren einer vorgegebenen Position

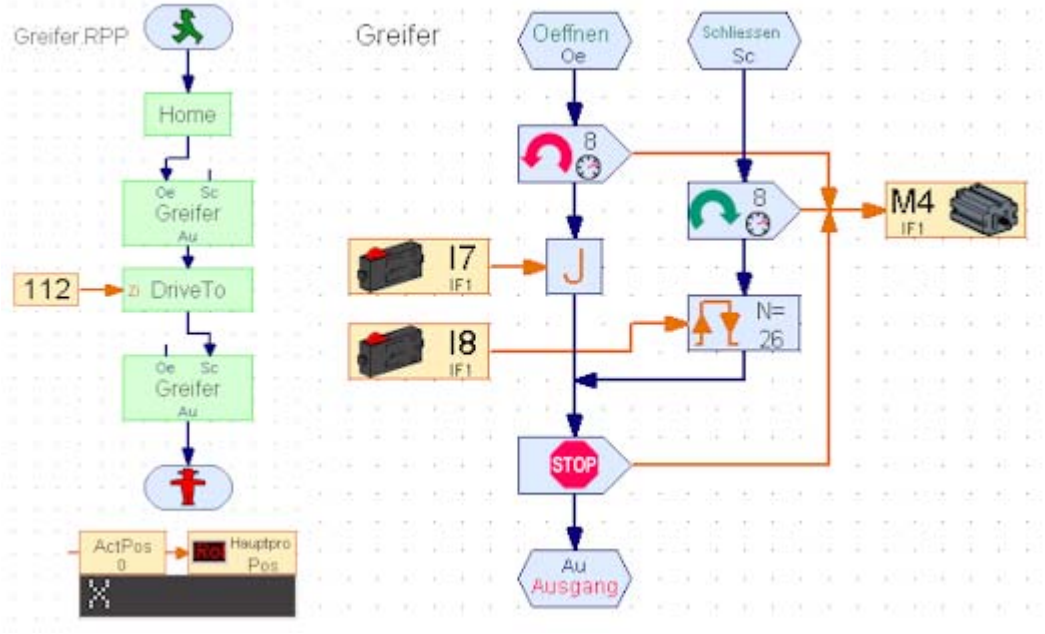


Die anzufahrende Position wird über den Parameter ZielPos vorgegeben. Die aktuelle Position wird in der globalen Variablen ActPos gehalten, sie muß im Hauptprogramm nicht in Erscheinung treten, sehrwohl aber in einer zugehörenden Sub Home. Zentraler Befehl ist die Abfrage A?B mit der ZielPos und ActPos verglichen werden. Bei Gleichheit ist die ZielPos erreicht, Motor Aus, Return.

Ist ZielPos > ActPos geht's nach rechts mit M1, es wird auf einen an I2 auftretenden Impulse gewartet und ActPos jeweils um einen erhöht. Bei ZielPos < ActPos geht's nach links in Richtung Endtaster.

Man könnte hier noch vorsichtshalber eine Abfrage auf den zugehörenden Endtaster und den maximalen Fahrweg einbauen, oder einfach die Bibliotheksroutine PosX / Pos nehmen.

Greifer : Subroutine mit zwei Eingängen, Position Anzeigen



Der Greifer weicht im Einsatz etwas von den anderen Komponenten ab, er wird meist nur geöffnet und geschlossen. Allerdings kann der Schließwinkel in Form von Impulsen vorgegeben werden. Wenn man stets die gleichen "gelben Tonnen" transportiert, kann das ein fester Wert sein, hier 26 im Baustein Impulszähler.

Das Öffnen des Greifers geschieht über ein "Warten auf J".

In diesem Fall wurde auf Parameter für Öffnen und Schließen verzichtet und stattdessen entsprechende Eingangspunkte ins Programm eingebaut. Diese Art des Aufrufs ist eher veraltet.

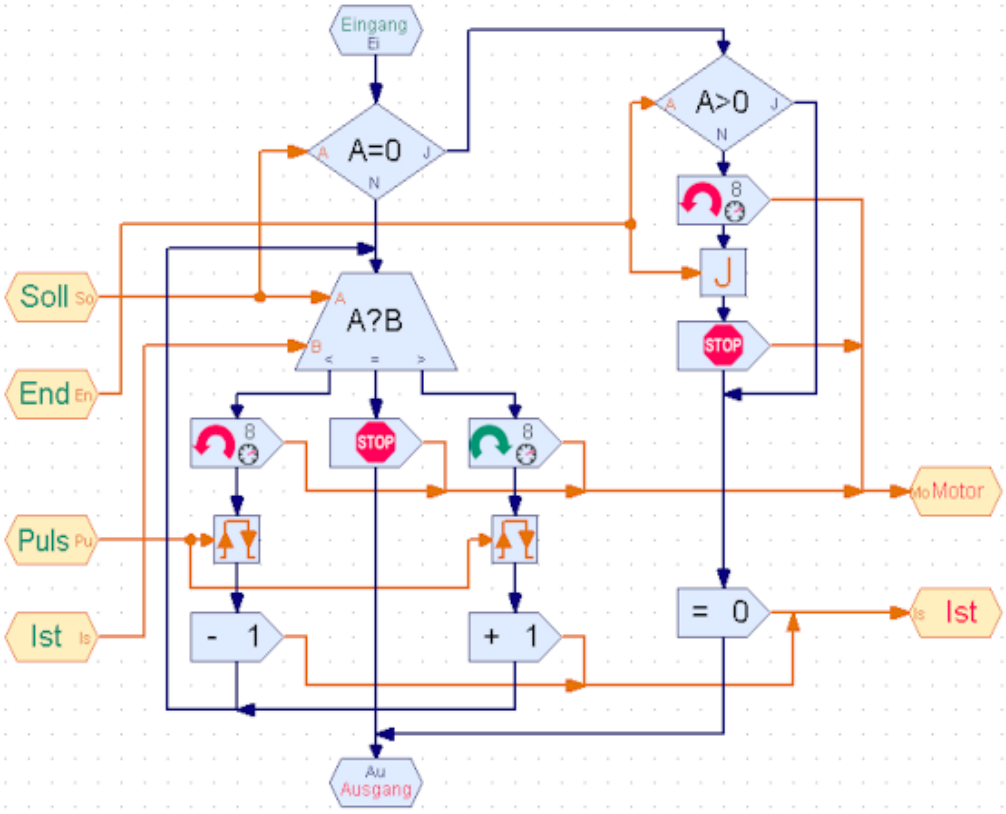
Hier wurde das Home (ohne Greifer auf) vom vorherigen Punkt genutzt, man kann natürlich auch noch ein Greifer auf im UP unterbringen.

Der aktuelle Wert der globalen Variablen ActPos (aktuelle Position der Säule) wird im Hauptprogramm ebenfalls angezeigt. Man könnte dann noch je eine Anzeigelampe für den Status des Greifers einbauen. Das könnte man z.B. über den "=" Befehl machen.

Anmerkungen zur Bibliothek Position ES

Routinen zur Ansteuerung von Positionen, die durch Zählen der Impulse beim Betrieb eines Motors mit Impulsrad / Impulstaster auftreten. Zur Feststellung einer Null-(Home)Position wird ein Endtaster benötigt, bei dessen Betätigung die Home-Position angenommen wird. Der Motor dreht beim Anfahren des Endtaster links im Sinne der ROBO Pro-Befehle. Der Zusatz ES im Bibliotheksnamen betont die Auswertung des entsprechenden Endtaster.

Pos : Anfahren einer vorgegebenen Position

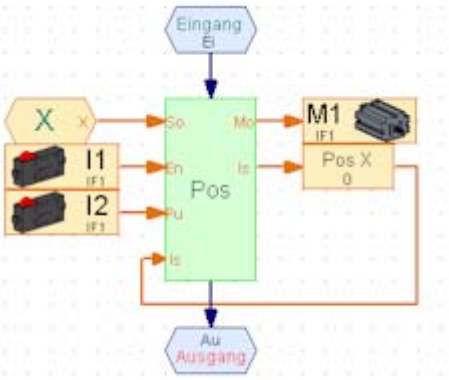


Parameter
- Soll : Anzufahrende Position, Variable oder Konstante
– Ist : Aktuelle Position, muß eine globale Variable sein.
gleichzeitig Eingangs- und Ausgangsparameter, Trick um Pos flexibel einsetzen zu können, siehe auch PosX
– End : Zugehörender Endtaster
– Puls : Zugehörender Impulstaster
– Motor : Zugehörender Motor

Es wird unterschieden zwischen dem Anfahren einer allgemeinen Position und der Position 0. Dort werden keine Positionen gezählt, sondern der Endtaster direkt (linksdrehend) angefahren und nach Erreichen die aktuelle Position (Ist) auf 0 gesetzt.

Anders das Anfahren einer allgemeinen Position. Hier wird in einer Schleife zunächst auf das Erreichen der Zielposition geprüft und gleichzeitig die Drehrichtung bestimmt. Anschließend wird nach Einschalten des Motor auf jeweils einen Impuls gewartet und dann die aktuelle Position inkrementiert / dekrementiert.

PosX : Anfahren einer vorgegeben Position auf der "X-Achse"

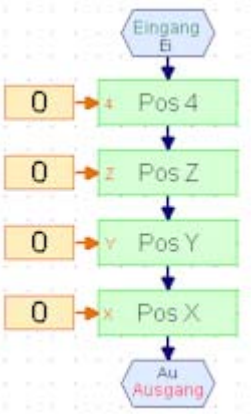


Die Arbeit wird in Pos getan, hier werden lediglich die für die X-Achse spezifischen Parameter gesetzt. Die Ziel-Position wird dabei von außen durchgereicht.

Interessant ist die Handhabung der aktuellen Position. Sie wird in der globalen Variablen Pos X gespeichert. Jede Veränderung des Parameters Ist von Pos zieht eine Veränderung von Pos X nach sich, diese wird dann als Vorgabe wieder an Pos zurückgereicht.

So kann Pos die Arbeit für Pos X, Pos Y, Pos Z und Pos 4 erledigen.

PosInit : Anfahren von Home



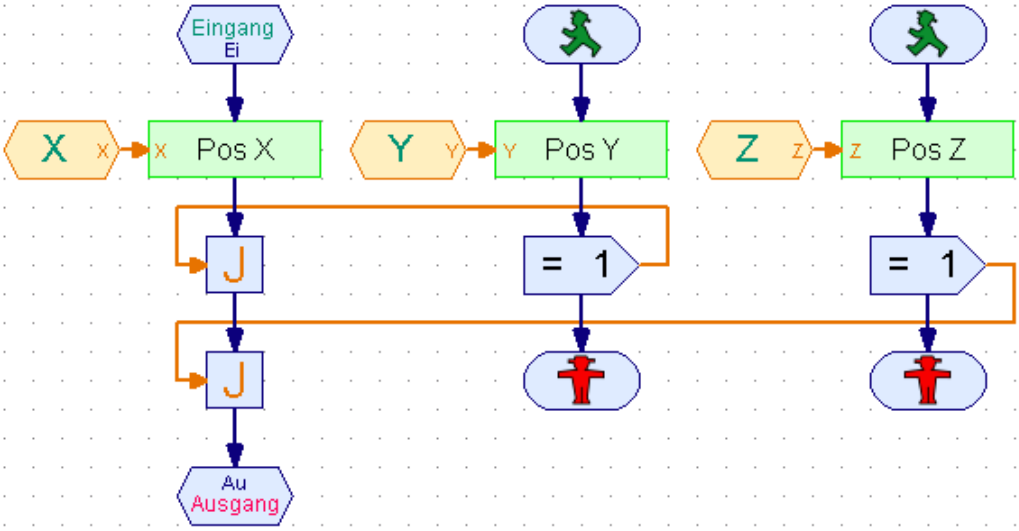
Die Home-Position des Gesamtrobots wird, Komponente für Komponente, nacheinander angefahren. So besteht die größte Chance den Robot aus einer räumlich engen Position heraus zu lotsen.

PosInit ist sehr einfach : es nutzt die Sonderfunktion von Pos zum direkten Anfahren des Endtaster bei Vorgabe der Zielposition 0.

Anzeige : Bedienfeld

Ist eher eine Komfort-Funktion zur Anzeige der aktuellen Position. Dazu werden die globalen Variablen Pos X, Pos Y, Pos Z und Pos 4 über Anzeige-Ausgänge an Anzeige-Bedienelemente auf dem Bedienfeld weitergeleitet. Hier werden gleichzeitig alle benutzten globalen Variablen gelistet.

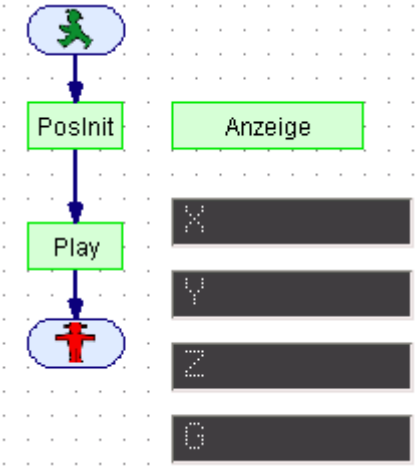
Pos XYZ : Simultanes Anfahren einer vorgegeben Position



Die für X, Y, und Z-Achse vorgegebenen Positionen werden gleichzeitig angefahren. Das geschieht in drei verschiedenen Prozessen : Dem des Hauptprogramms und zweier neuer, die in Pos XYZ gestartet und auch wieder beendet werden. Im jeweiligen Prozeß wird eine Routine zur Positionierung einer Komponente ausgeführt. Dabei wartet der Hauptprozeß auf die Fertigmeldung durch die "Hilfs"-Prozesse. Das geschieht durch Senden einer 1 an den Warte-Befehl "J".

SäulenRobot nach Liste

iRobListe : Hauptprogramm



Ist sehr schlicht gehalten. Auch hier wieder die Anzeigen direkt auf der Hauptform. Angezeigt werden die aktuellen Positionen der Robot-Komponenten.

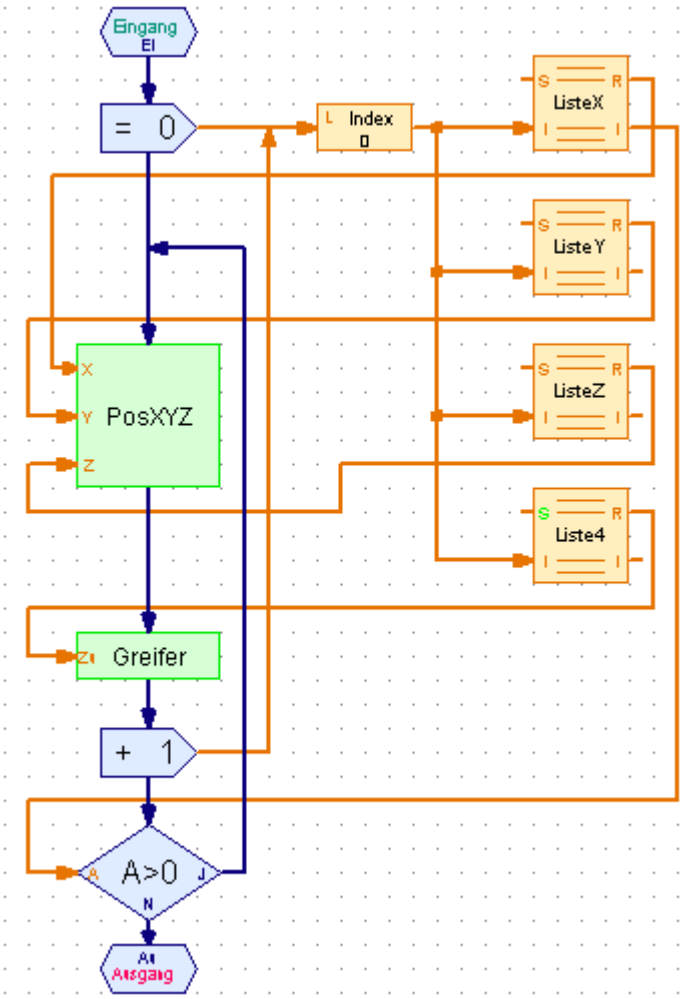
Die modifizierte Bibliotheks-Routine Anzeige enthält die Zuweisungen der betroffenen Globalen Variablen zu den entsprechenden Bedien-Ausgängen, die Anzeigen im Bedienfeld wurden gelöscht.

Verwendet werden die Globalen Variablen PosX (Säule), PosY (Arm horizontal), PosZ (Arm vertikal) und Pos4 (Greifer), die in Anzeige schon mal alle genutzt werden.

Mit PosInit wird die Home-Position angefahren. Das ist die Position am zugehörenden Endtaster, sie wird linksdrehend und nacheinander in der Reihenfolge 4, Z, Y, Y angefahren.

In Play spielt dann die Musik, die Listen mit den Positionen für die Rob-Komponenten wird abgearbeitet. Nur einmal, wenn das nicht genügt, endlos durch eine Schleife um Play. Mißtrauische werden werden auch noch PosInit in die Schleife einbeziehen um die Ausgangslage, allen Eventualitäten zum Trotz, wiederherzustellen.

Play : Abarbeiten der Liste



Pro Rob-Komponente ist eine Liste mit der jeweils anzusteuernden Position vorhanden. Unveränderte Positionen müssen nochmal aufgeführt werden.

Ein Index (Lokale Variable), der bei Start auf 0 gesetzt wird, enthält einen Hinweis auf die aktuelle Position in den einzelnen Listen. Jede Veränderung des Index bewirkt eine Ausgabe des zugehörenden Listenwertes über den Ausgang R des Listenelements und weiter an PosXYZ und Greifer. Die neue Position wird simultan angefahren und anschließend der Greifer getätigt. Das wird separat gemacht, da ein simultanes Betätigen des Greifer zum Verlust des gegriffenen Teils, irgendwo auf dem Weg, führen würde. Der Greifer erhält die Werte 0 für Öffnen und 1 für Schließen. Die Routine Greifer bildet das intern auf die Bibliotheksroutine Pos4 ab, dort wird dann der tatsächliche Schließweg angegeben.

Nach Greifer wird dann der Index um 1 erhöht. Der I-Ausgang liefert eine 0, wenn der Index zu groß geworden ist (Ende der Liste), sonst die Anzahl der gespeicherten Listenelemente (Das Handbuch sagt etwas anderes).

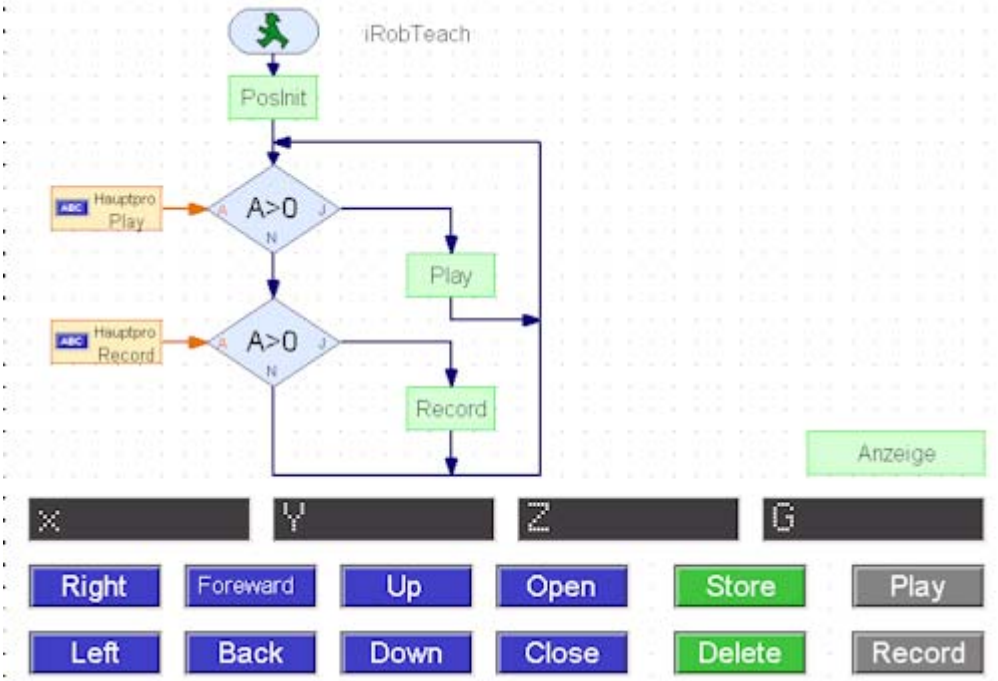
Achtung : Die Bibliotheks-Routinen verwenden Namen, die ein Leerzeichen enthalten. Das geht irgendwann mal schief, ich habe deswegen das Leerzeichen entfernt (oder waren es sogar zwei oder gar keins?).

Listen ListeX, -Y, -Z und -4

Die genannten Listen enthalten konstante Vorgabewerte im Feld "Liste der Anfangswerte" der jeweiligen Eigenschaftsseite. Sie können auch in einer einzigen Datei gespeichert werden. Dazu bekommen sie Spaltentitel und eine Spaltennummer. Außerdem muß einer Separator (default Komma, siehe auch Hinweise oben) angegeben werden. Sie können dann vor Start des Programmes manuell (Menü Datei | CSV, Haken bei CSV-Speicher in Eigenschaftsseite) oder automatisch (Dateiname in Eigenschaftsseite) geladen werden. Ebenso können sie nach Programmende gespeichert werden.

Ein TeachIn für den SäulenRobot

iRobTeach : Das Hauptprogramm



Diagramm

Der Robot (KnickarmRobot ist genauso möglich) wird auf Home-Positione gefahren. Anschließend werden in einer Endlosschleife die (grauen) "Play" und "Record" Buttons abgefragt und die zugehörenden Routinen aufgerufen. Die Buttons sind Druckschalter, sie bleiben also unten bis sie nochmal angeklickt werden. Mit der (modifizierten) Bibliotheksroutine Anzeige wird die aktuelle Position über die globalen Variablen PosX ... angezeigt. Die aufgezeichneten Positionen werden in den globalen Listen ListeX ... gehalten. Sie enthalten schon eine Reihe von Werte – ready to play.

Genutzt wurden auch hier wieder Routinen der Bibliothek Position ES (wieder von blanks befreit). und zusätzlich Routinen von iRobListe.

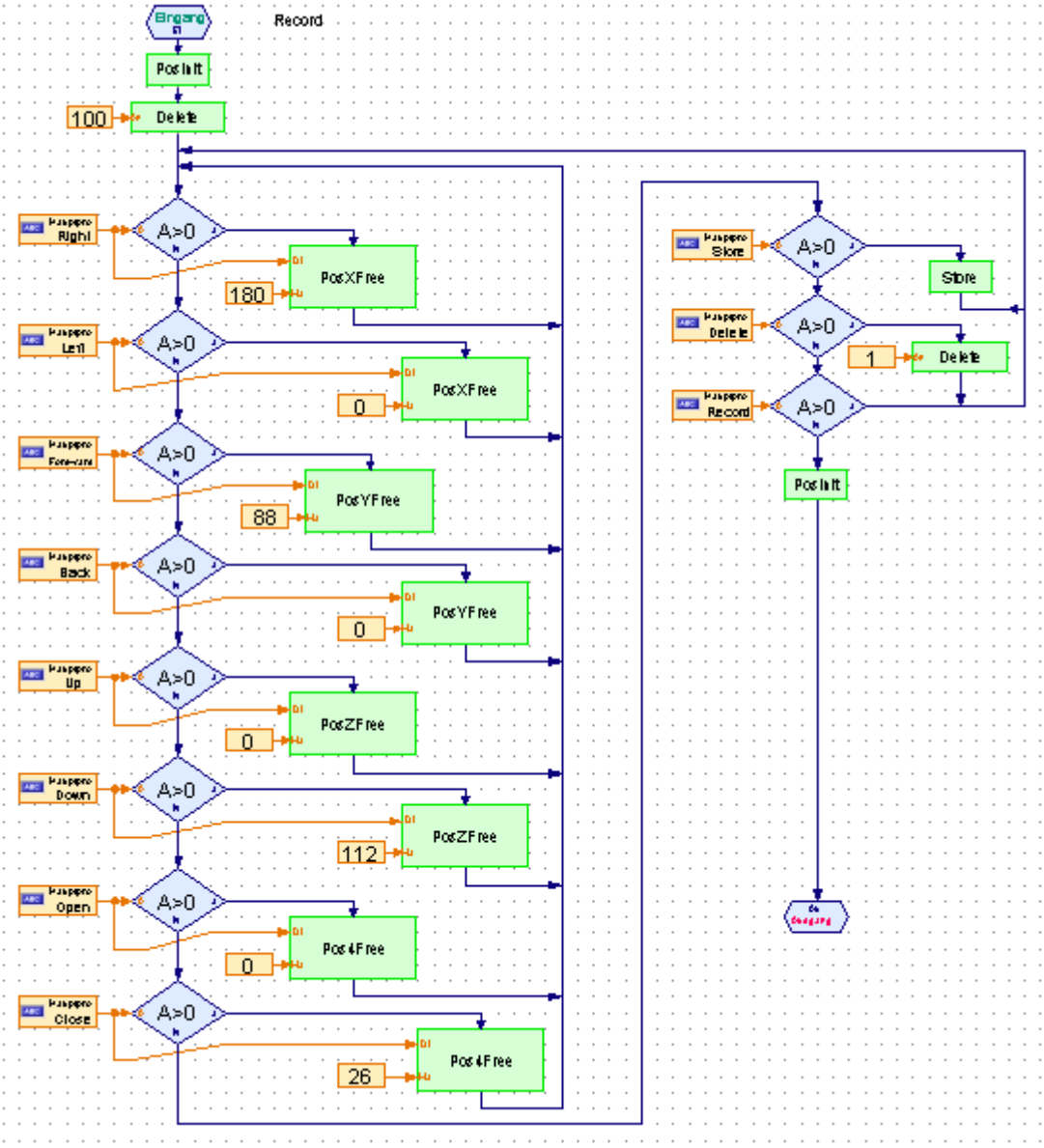
Bedienelemente

Die Bedienelemente wurden auch hier direkt auf der Hauptform plaziert, ist halt etwas bequemer. In der oberen Reihe wird die aktuelle Position angezeigt, darunter links der Block der TeachIn-Buttons, sie wirken nur, wenn auch draufgedrückt wird. Es folgen die Buttons für das Speichern der aktuellen Position und das Löschen der letzten gespeicherten Position. Dann folgen die Buttons für das PlayBack und und die Aufzeichnung.

Listen

Aufbau und Verwendung wie bei "SäulenRobot nach Liste". Beim TeachIn wird zunächst die gesamte Liste der Positionen gelöscht. Neue Positionen werden über Button Store / UP Store an das Ende aller Listen angehängt. Die jeweils letzte Position kann über Button Delete / UP Delete gelöschst werden. Da sich das Programm nicht selber beendet, muß es vor dem Speichern der Listen über den roten Button manuell beendet werden.

Record : Die Aufzeichnung des Bewegungsablaufes



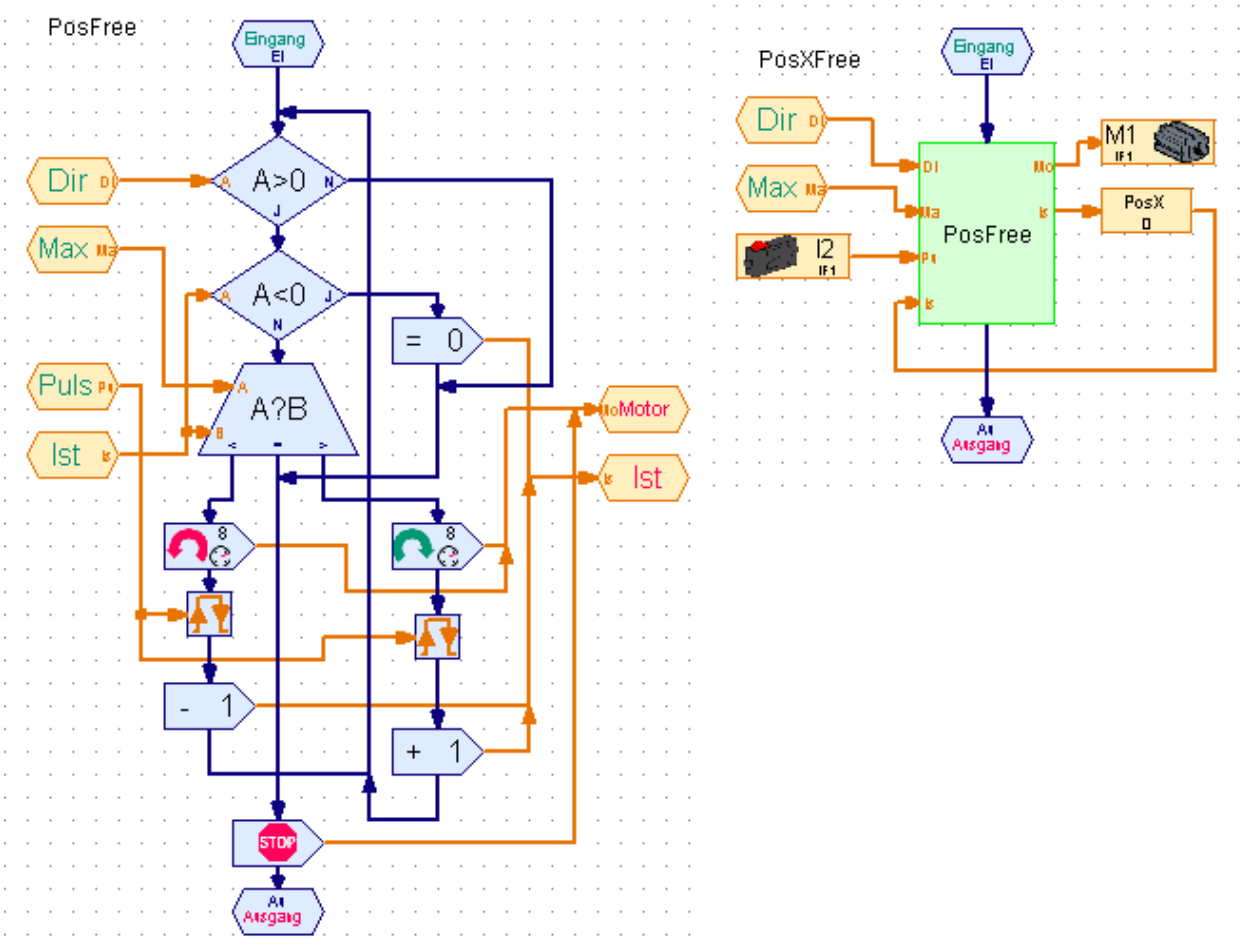
Beginnt mit dem Anfahren der Home-Position(PosInit) und dem Löschen der Positionslisten(Delete, eine einzelne Liste ist max. 100 Elemente lang). Der Rest ist eine große Schleife in der die TeachIn-Buttons abgefragt werden und die zugehörenden Routinen aufgerufenwerden. Die Routinen Pos.Free bekommen als Parameter den rufenden Button und die Maximal-Position übergeben. Die Maximal-Position ist gleichzeitig die Angabe der Bewegungsrichtung, die Aufrufe treten deswegen immer paarweise auf.

Das Ende der Schleife (für die Schweizer Leser : Schlaufe) bilden die Button für das Speichern bzw. Löschen der letzen Position. Sie enthalten eine Pause um ein Click-Ereignis nachzubilden. Auch so muß man sehr schnell den Finger von der Maus lassen, sonst wird mehrfach gespeichert (man kann die Pausendauer natürlich seinem Temperament anpassen).

Hinweis : Unabhängig von der aktuellen Schließposition wird beim PlayBack immer auf Position 26 geschlossen, man kann das in Routine Greifer ändern.

Wenn der Button Record nicht mehr gedrückt ist, wird die Schleife beendet, der Robot steuert wieder Home an.

PosFree / Pos_Free : Anfahren einer Position



PosFree ist die allgemeine Routine zum freien Anfahren einer Position, sie entspricht im Aufbau weitgehend der Bibliotheksroutine Pos (Details siehe dort), betrachtet aber die Positionsvorgaben als Maximal- bzw. Minimalwert und nicht als Zielvorgabe. Gefahren wird solange der zugehörende Button (Parameter Dir A>0) gedrückt ist oder der MiniMax-Wert erreicht wurde.

Auf eine Abfrage des zugehörenden Endtaster wurde verzichtet, da er sich sehr sperrig anstellt (er ist für die Dauer von mehrer Impulsen gedrückt), stattdessen führen Positionen (Ist < 0) zum Ende der Routine, die aktuelle Position wird in diesem Fall auf 0 gesetzt.

Pos_Free paßt PosFree dann der einzelnen Komponente an. Feste Angabe des Impulstasters und des Motors, Zuordnung der globalen Variablen, die die aktuelle Position (mit einem eleganten Schlenker – Logik der gelben Linien – wie auch schon bei Bibliothek Pos_) hält.

Punkte über die man sich ärgern sollte

1. Die aktuelle Position in der Liste wird nicht angezeigt.
2. Es fehlt ein Status-Text (Record Mode, Turm dreht ...)
3. Die Pause bei Store und Delete ist eher eine Krücke
4. Das in Record integrierte PosInit / Delete kann manchmal richtig nerven
5. u.v.a.

Das alles sollte man – nachdem man sich ausgeärgert hat – viel schöner machen.