Einführung in die Programmierung von

Industry Robots mit LLWin

Ulrich Müller

Inhaltsverzeichnis

Bändigen des Robots

Allgemeines

Das Dokument gibt eine schrittweise Beschreibung der Programmierung eines Industry Robots mit LLWin. Begonnen mit einfachen Routinen zum Betrieb der Robot-Säule bishin zu einem TeachIn-Programm.

Die Programmierung wird im Detail beschrieben, auf das Arbeiten mit LLWin selber wird aber nur am Rande eingegangen. Die vorgestellten Programme bauen aufeinander auf. Sie entsprechen in manchen Teilen der LLWin-Vorlage für Robots, sind aber eine eigenständige Lösung, die man ebenfalls als Vorlage für die Erstellung eigener Programme nutzen kann.

Hauptzweck dieser Lösung ist die Darstellung und Beschreibung einzelner Programmiertechniken zum Betrieb von Robots. Die Programme sind einfach gehalten, reizen aber die Möglichkeiten von LLWin weitgehend aus.

Da die Mehrzahl der grundlegenden Operationen für alle Komponenten (Säule, Arm waagerecht, Arm senkrecht, Greifer) gleich sind, wird anfangs nur der Umgang mit der Säule beschrieben.

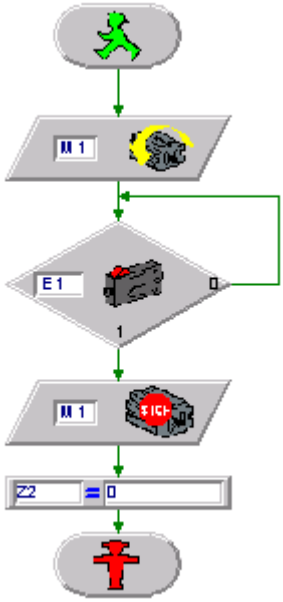
Voraussetzungen

Installiertes LLWin 3.0. Anfangskenntnisse im Umgang mit LLWin

Einen nach Bauanleitung gebauten und verdrahteten Industry Robot (Modelle Säulen- oder Knickarmrobot). Siehe auch Anhang.

Home : Auf Ausgangsstellung

Die Säule allein



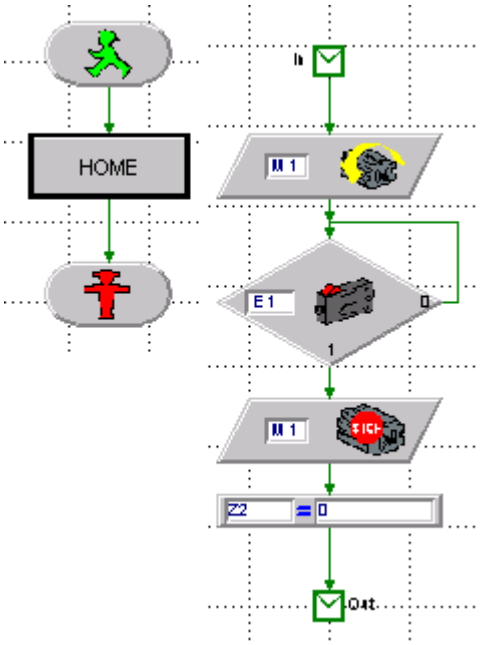
Die Robots nutzen zur Bestimmung der aktuellen Position Impulsrädchen, die auf dem Getriebe des Antriebsmotors der jeweiligen Komponente sitzen und einen Taster betätigen. Das Impulsrädchen hat vier Nocken und - dazu gehörend – vier Nockentäler. Gibt pro Umdrehung acht Impulse. Die aktuelle Position 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
- In einer Schleife abfragen, ob der Endtaster erreicht ist (E1 = 1 ist).
- Motor M1 wieder abschalten
- Den zugehörenden Impulszähler Z2 auf 0 setzen.

Das wars.

Die Säule über Unterprogramm Home

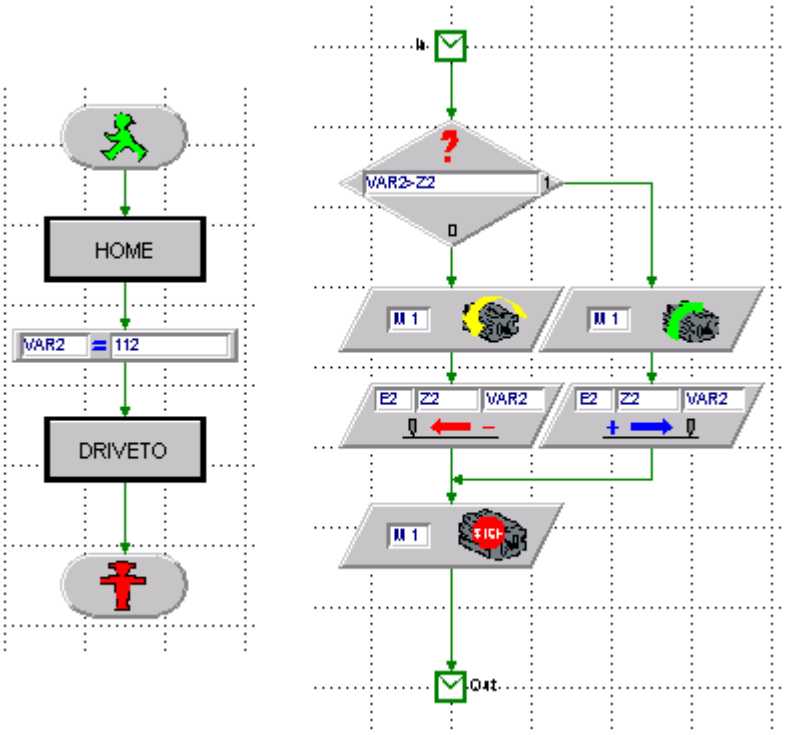


Noch fehlen zwar die Bausteine für die weiteren Robot-Komponenten, aber um das Gesamtprogramm übersichtlicher zu machen, werden die Bausteine in ein Unterprogramm (UP) ausgegliedert. Die grünen / roten Männchen werden dabei durch die Ein- und Ausgangs-Symbole eines Unterprogramms ersetzt, die anderen Bausteine bleiben erhalten. Hinzu kommt ein neues $MAIN mit den Männchen und einem Aufruf des Unterprogramms. Die Gesamtfunktion bleibt gleich.

Man sollte es allerdings mit den Unterprogrammen nicht übertreiben, denn in LLWin werden, im Gegensatz zu anderen Programmiersprachen, die Bausteine des Unterprogramms intern wieder anstelle des Unterprogrammaufrufs eingesetzt. Es sind also keine "echten" Unterprogramme sondern Makros – Kurzschreibweisen für eine Folge von Bausteinen. Deswegen sind die LLWin-Variablen (hier Z2) auch von allen Programmteilen gleichermaßen zugreifbar.

Positionieren

Die Säule auf Position : DriveTo



Nach Ende des UPs HOME steht die Säule des Robots auf der Home (Null) Position. Danach soll der Robot nun zur Position 112 (112 Impulse ab Null) fahren. Das wird wieder in einem Unterprogramm gelöst.

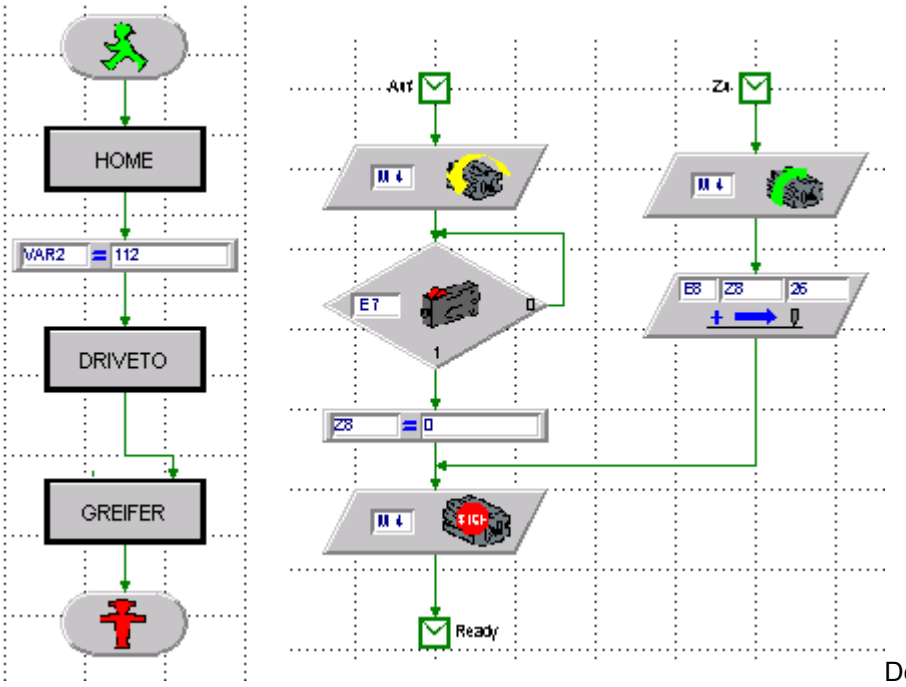
Dazu wird zunächst in der neuen Variablen VAR2 der Wert 112, die Zielposition, gespeichert. Im Unterprogramm DriveTo wird dann abgefragt, ob die Zielposition größer als die augenblickliche (Z2) ist. Wenn das der Fall ist wird Motor M1 rechtsdrehend – weg von der Homeposition – gestartet, andernfalls linksdrehend – in Richtung Home.

Anschließend wird im Positions-Baustein das Erreichen der Zielposition überwacht. Die erfordlichen Angaben für den Positions-Baustein sind Impulstaster (E2), aktuelle Position (Z2) und Zielposition (VAR2). Bei linksdrehendem Motor werden dann die festgestellten Impulse von der aktuellen Position abgezogen, bei rechtsdrehenddem Motor auf addiert. Stimmt der Wert von Z2 mit dem Wert von VAR2 überein, ist die Zielposition erreicht. Anschließend wird der Motor wieder abgeschaltet.

Anstelle der angegebenen Variablen können natürlich auch andere angegeben werden, für die Zielposition könnte auch gleich die Konstante 112 angegeben werden. Da die Zielposition später aber häufiger geändert werden soll, ist die Angabe in einer Variablen aber vorteilhafter.

Das Testen von Unterprogramme ist immer ein wenig schwierig. Einfacher ist es, die Bausteine zunächst mal in das $MAIN einzubauen und erst wenns läuft in ein UP zu verlagern.

Greifer auf und zu : Greifer, UP mit zwei Eingängen



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 Position.

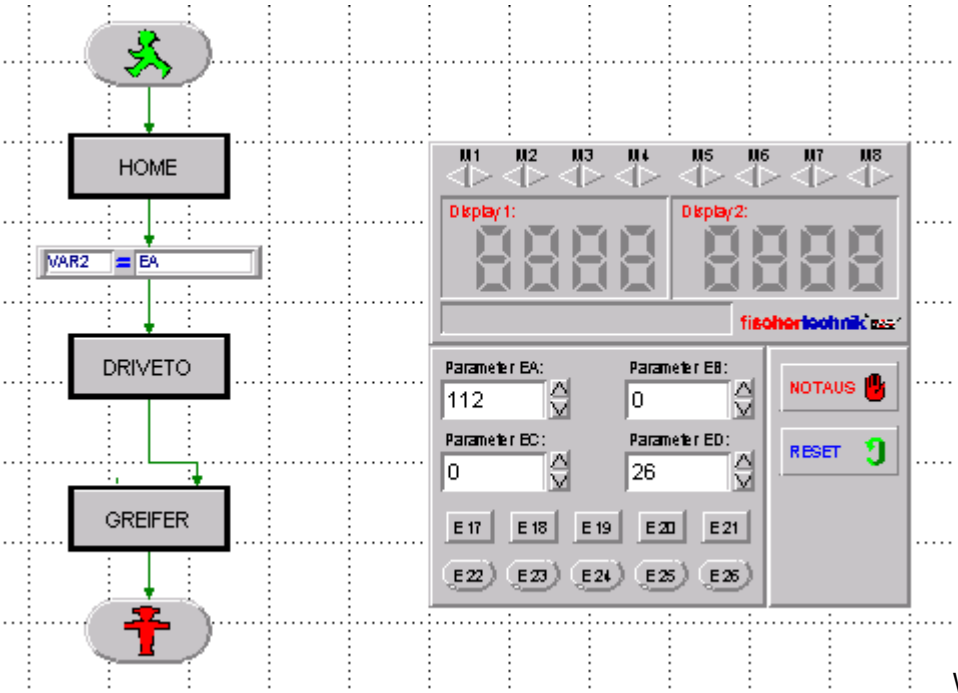
Das Öffnen des Greifers geschieht mit einer Bausteinfolge wie beim Home, das Schließen entspricht dem Rechtsdrehen von DriveTo.

Die Bausteine des Greifers werden wieder in ein UP : GREIFER gepackt. Besonderheit dieses UPs sind seine zwei Eingänge "Auf" und "Zu". Der Baustein muß dazu mit Menü "Design" entsprechend bearbeitet werden.

Hier wurden die Funktionen Greifer Auf und Greifer zu in einem UP untergebracht. Das sieht schön aus, kostet aber zusätzlichen Platz, da bei jedem Aufruf alle Bausteine an den Ort des Aufrufs kopiert werden. Wenns eng wird ist eine Lösung mit zwei UPs GREIFERAUF und GREIFERZU platzsparender.

Hier nicht zu sehen : die Verwendung von UP Greifer in HOME.

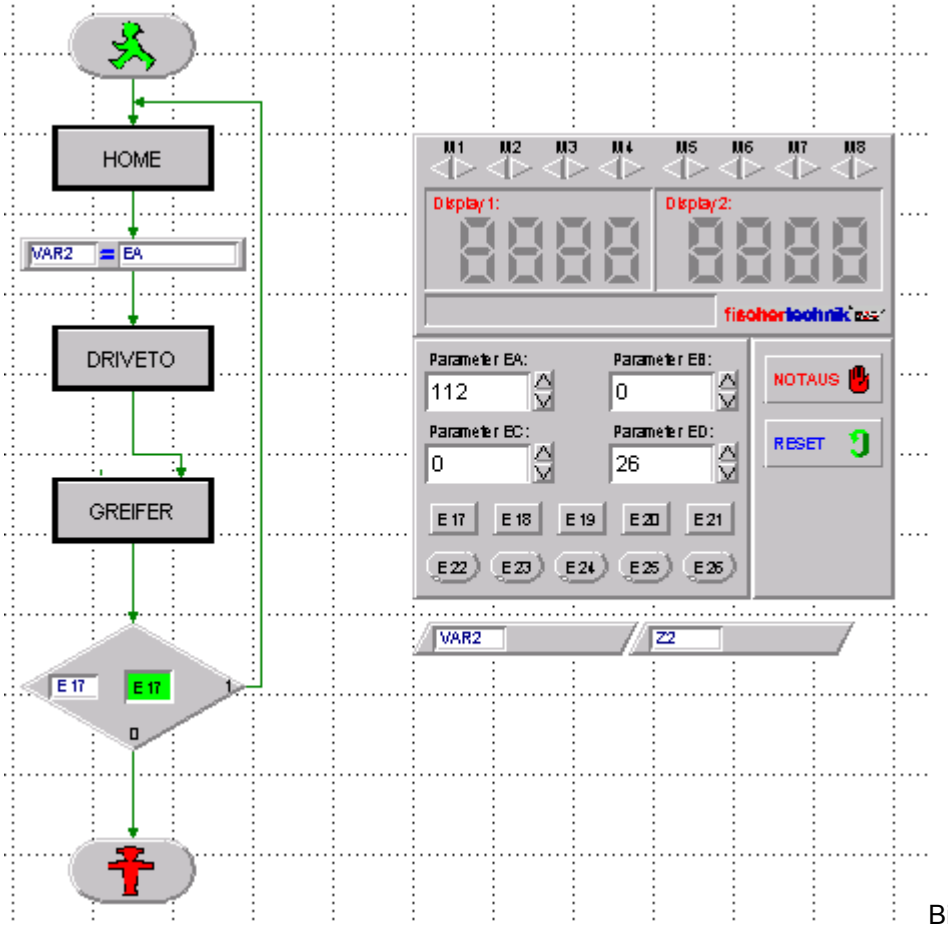
Positionsangabe über den Terminalbaustein



Wenn man die Zielposition oder den Öffnungswinkel des Greifers häufiger ändern will, ist eine Angabe des aktuellen Werte zentral im Terminal-Baustein sinnvoll. Hier wurde der Parameter EA für die Angabe der Zielposition und der Parameter ED für den Schließwinkel des Greifers genutzt. VAR2 wird dann folgerichtig mit EA besetzt. Im UP Greifer wird dann anstelle der Konstanten 26 die Variable ED angeben.

Einfacher Zyklus

Pendeln zwischen Home und Position A



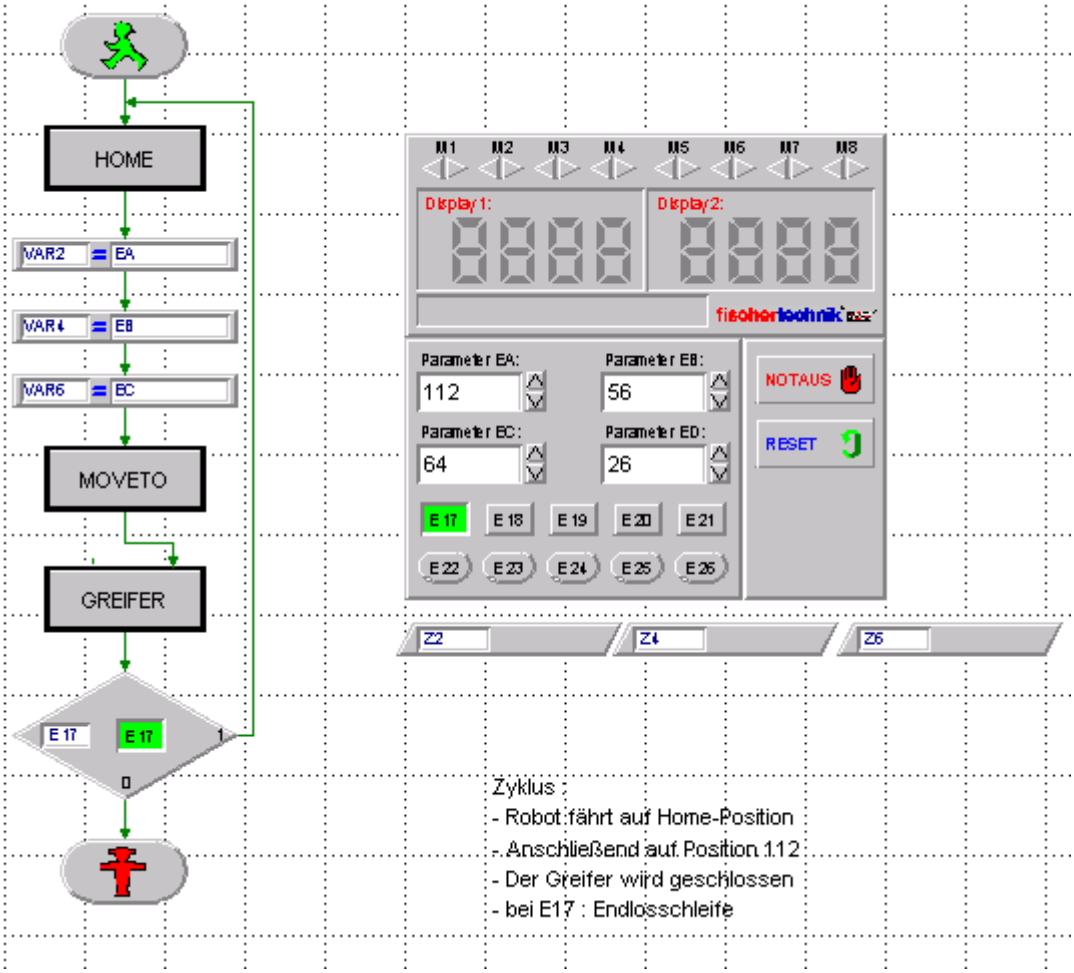
Bislang war das Programm eine einmal Veranstaltung : HOME – DRIVETO – GREIFER : Ende.

Jetzt wird der "virtuelle" E-Eingang E17 genutzt, um über den Terminal-Baustein eine bedingte Schleife zu konsturuieren. Ist E17 gedrückt (grün) wird die angeführte Bausteinfolge wiederholt, bis E17 wieder gelöscht wird (oder die Ampel auf rot geschaltet wird). Der Robot werkelt jetz also endlos vor sich hin. Säule auf Position Null, Greifer auf, fahren nach Postion 112, Greifer zu.

Wenn man dem Robot bei diesem Ablauf schon gelbe Tonnen zu "fressen" gibt, wird er sie vom Sockel stoßen, da der Arm immer die gleiche Höhe "über Grund" hat. das wird sich im nächsten Abschnitt ändern.

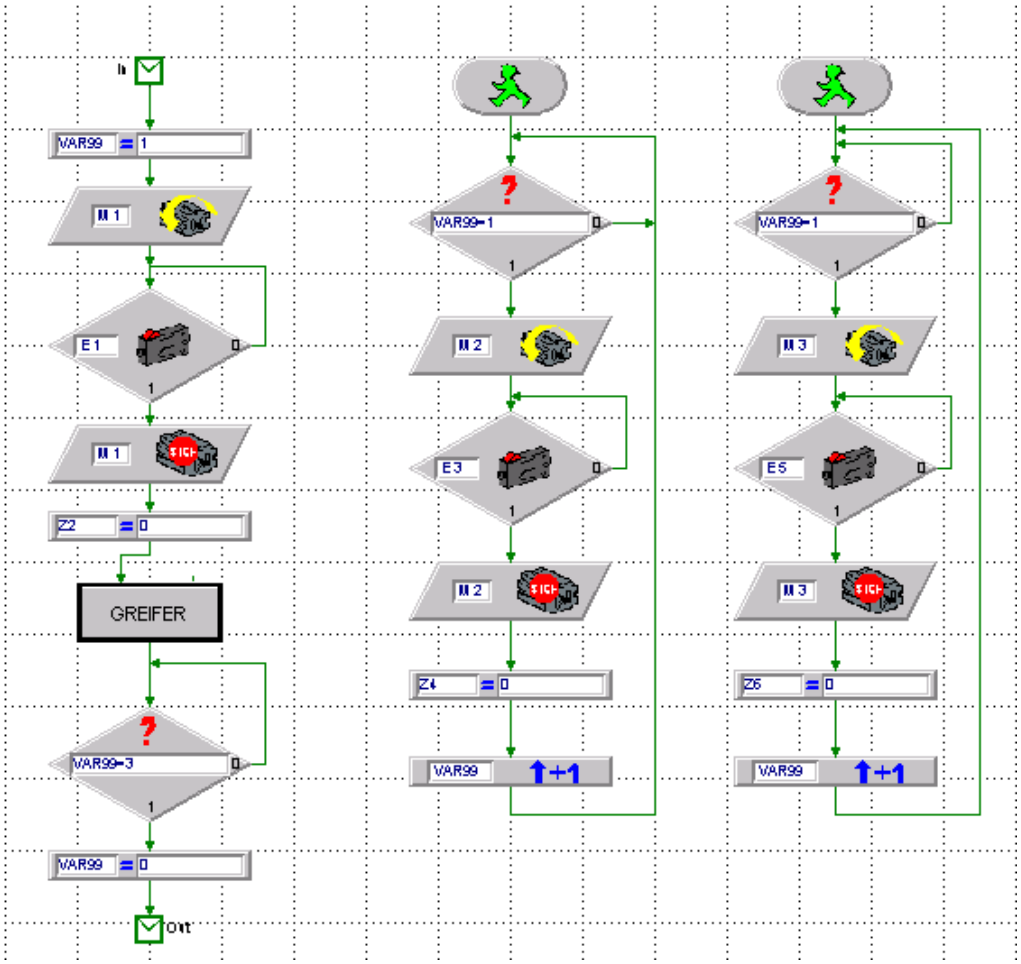
Mehrere Abläufe

Der Ablauf $MAIN



Der Zykus (Ablauf) ist zunächst erstmal der gleiche wie gehabt, es fällt nur auf, das in allen Parameterfeldern jetzt Werte stehen. Außerdem haben sich die Variablen in $MAIN vermehrt : VAR2 für die Zielposition der Säule, VAR4 für Arm horizontal und VAR8 für Arm vertikal. Sonst ist nichts zu sehen. Es wird dann wohl in den UPs liegen.

Home : alle Robot-Komponenten auf einmal



Hier geht mit allen Komponenten simultan (gleichzeitig, Ausnahme Greifer, der klappert nach) nach Home (Null). Der Pfad links ist weitgehend bekannt. Hinzugekommen sind die Abläufe (grüne Männchen, Threads) für die Motoren an M2 und M3.

Abläufe sind Folgen von Bausteinen, die eigenständig laufen. sie werden bei Beginn des Programms parallel zum $MAIN-Ablauf gestartet. Sie können durch ein rotes Männchen beendet werden. Hier laufen sie endlos. Die Unterbringung auf der Seite des UP HOME ist imgrunde zufällig, aber sinnvoll. Formal haben sie nichts mit dem linken Pfad zu tun.

Die beiden Abläufe enthalten zum großen Teil die gleich Baustein wie der Pfad links, aber mit Werten für die Motoren M2 und M3. Während der linke Pfad einen (UP)Anfang und ein Ende hat, besitzen die Abläufe rechts hier eine Endlosschleife. D.h. sie fangen gleich mit Start des Programm an und hören nicht mehr auf. Wenn man sie ließe würden sie den Robot auf seiner Home-Position festnageln (den Arm).

Es ist deswegen eine Synchronisation (Abstimmung) zwischen dem linken Pfad und den beiden Abläufen rechts erforderlich. Das geschieht über VAR99. VAR99 hat bei Programmstart und bei Erreichen der Home-Position den Wert 0 und bekommt erst bei Start von UP HOME den Wert 1. Das wird am Anfang der Abläufe abgefragt. Nur bei VAR99 = 1 wird die jeweilige Home-Position angesteuert. Ist sie erreicht, wird der Wert von VAR99 um 1 erhöht, ist also nicht mehr 1.

Am Ende des linken Pfades wird abgefragt und gewartet, bis alle Abläufe fertig sind. D.h. VAR99 = 3 ist. Dann ist Home erreicht, VAR99 wird dann wieder auf 0 gesetzt.

Anmerkung :

Die mit LLWin gelieferte Robot-Vorlage setzt hier pro Komponete eine eigene Home-Routine ein. Das hat den Vorteil, daß nicht ständig zwei zusätzliche Abläufe auf Arbeit warten. Außerdem kann man den Ablauf besser beeinflussen, wenn z.B. zwischen weiteren fischertechnik Modellen durch manövriert werden muß.

Diese Lösung sieht im Betrieb aber imposanter aus.

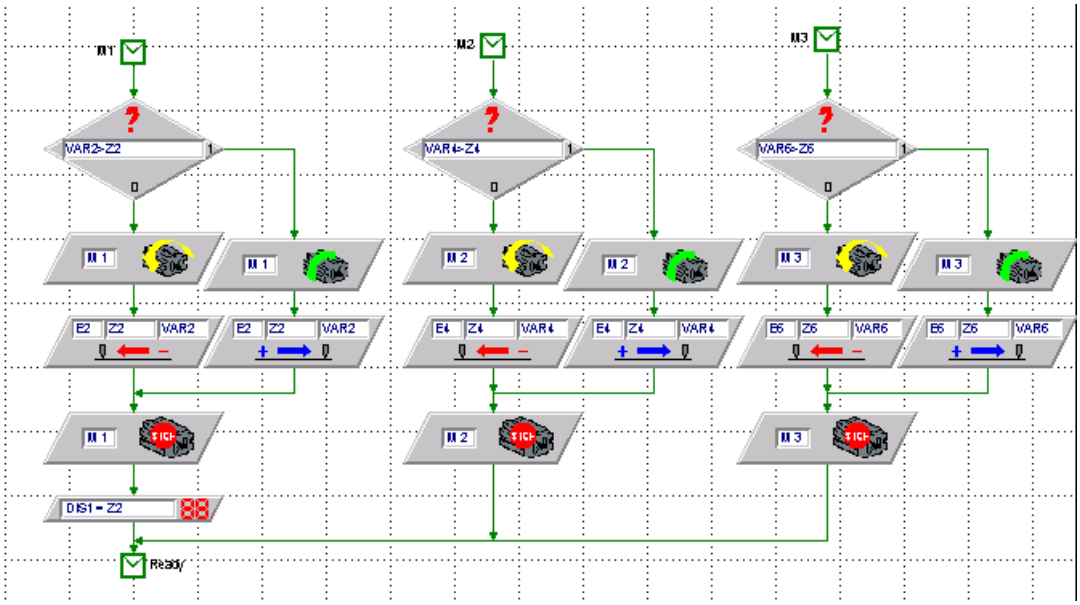
Anmerkung 2 :

Beim UP POSITIONIEREN verwendet die Vorlage noch einen Warte-Baustein um der Synchronisations-Variablen (hier VAR98) Gelegenheit zu geben, ihre Wirkung zu entfalten. D.h. die Zeit spielt hier eine Rolle und damit die Frage wer ist schneller VAR98=1 oder die VAR98=1 Abfrage.

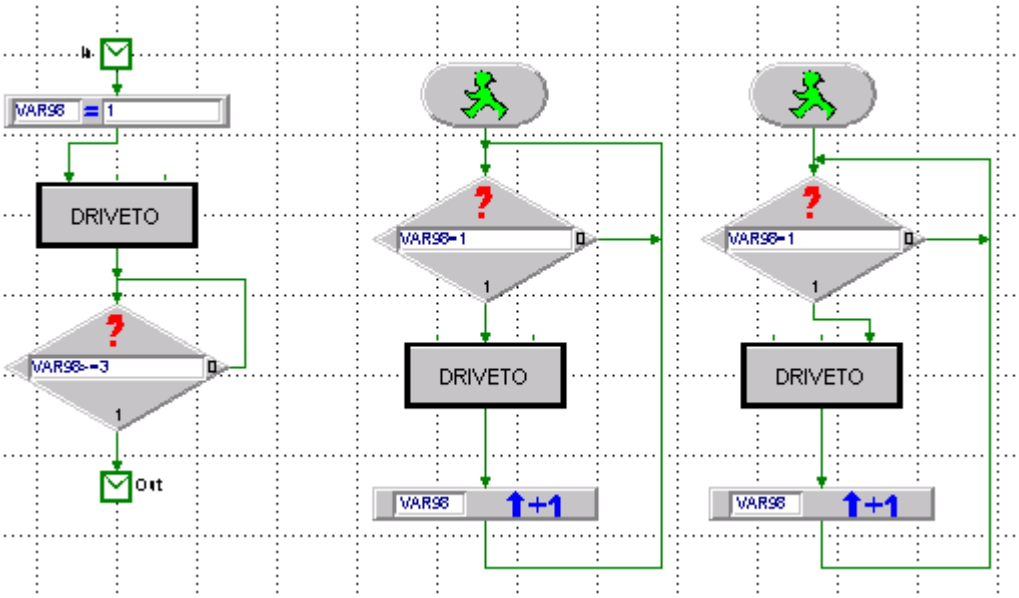
Im Projekt SAU10 wurde die Zeitverzögerung ebenfalls eingesetzt, in SAU12 wurde schlicht auf VAR98 >= 3 abgefragt das geht auch. Beides hat seine Tücken. Man sollte bei unerklärlichem Verhalten von parallelen Abläufen daran denken.

MoveTo : Auf Position gleichzeitig mit allen Komponenten

DriveTo



MoveTo



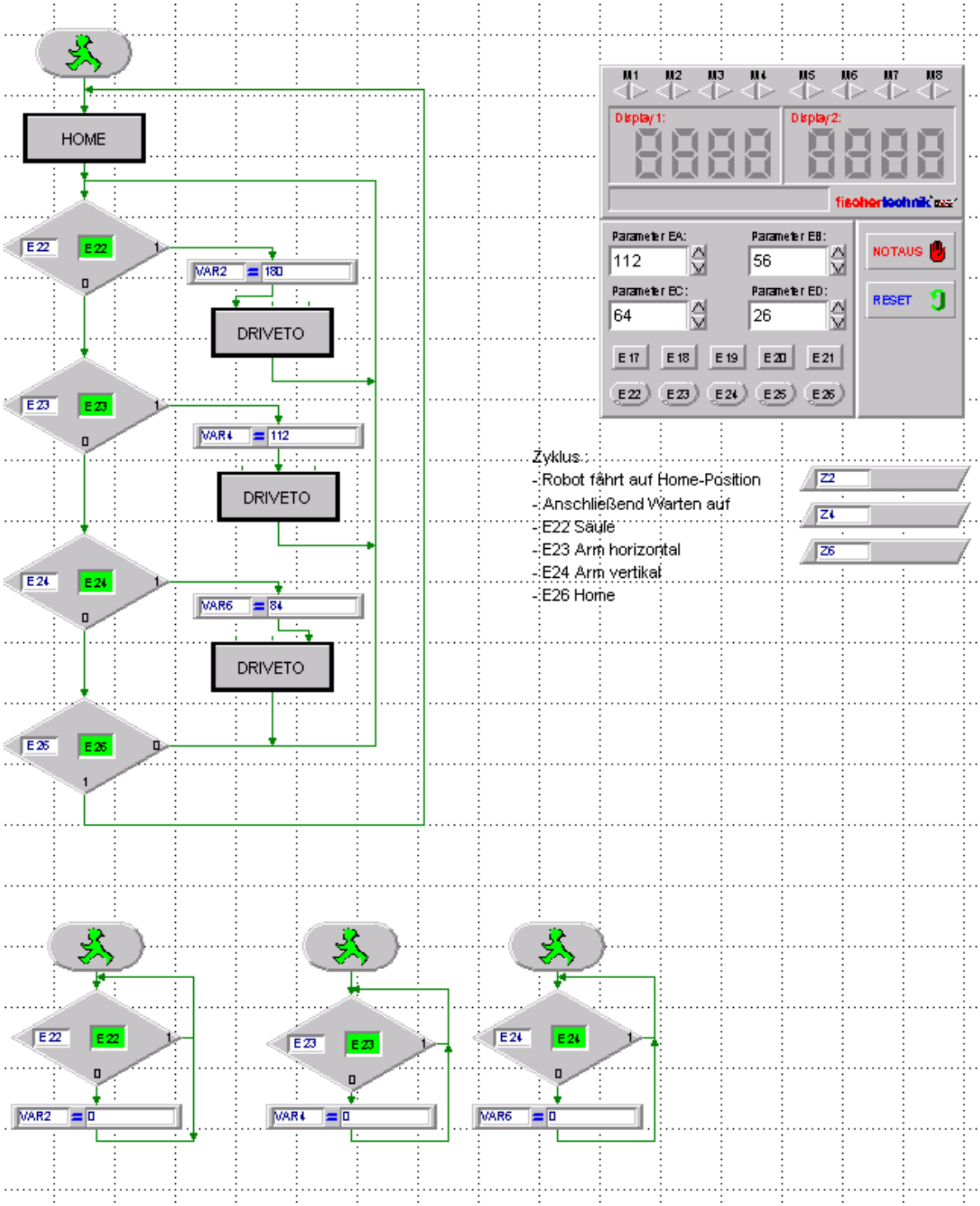
Hier wird erstmal das vorhandene UP DriveTo erweitert. Es bekommt drei Eingänge für die Motoren an M1 (Säule), M2 und M3 (Arm), alle ihre jeweilige Komponente auf Position fahren, immer eine auf einmal, ganz wie gehabt. Zusätzlich wird noch die Position der Säule im Terminal-Baustein angezeigt.

Neu ist hier das UP MOVETO das seinerseits die Komponenten in unabängigen Abläufen simultan auf Position fährt. Der Mechanismus ist der gleiche wie bei HOME. Her wird VAR98 zur Synchronisation genutzt.

Man hätte natürlich auch – platzsparender – die benötigten Bausteine von DriveTo hier einkopieren können. So sieht es schöner aus und man kann DriveTo unabhängig von MoveTo zum Bewegen einer Komponente nutzen.

TeachIn

Robot auf Position bringen



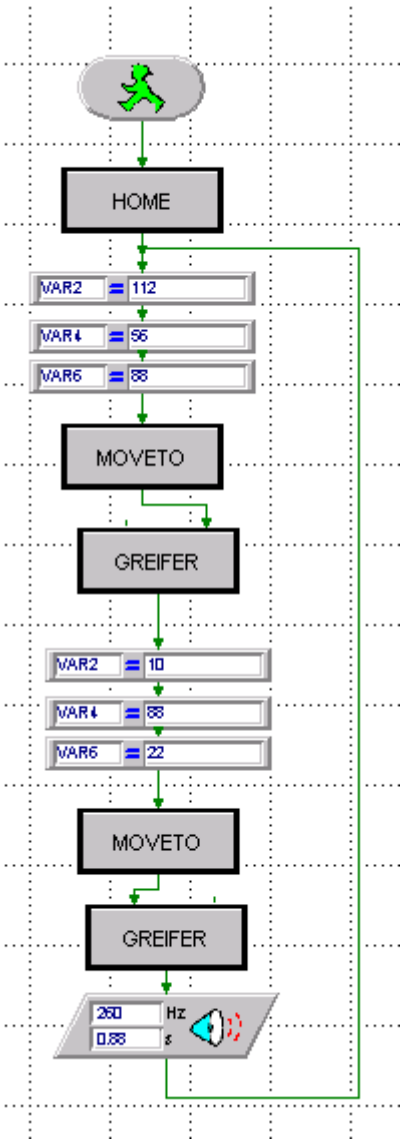
Über die Maus und E22 (Säule) oder E23/24 (Arm) des Terminal-Bausteins können die Robot-Komponenten einzeln positioniert werden. Immer ab Home-Position und solange die Maus gedrückt ist. Die erreichten Positionen werden in Z2, Z4 und Z6 angezeigt und können für einen späteren Automatik-Einsatz notiert werden. Die Home-Position wird über E26 erreicht, hier genügt ein kurzes Klicken.

In dem Haupt-Ablauf oben werden in einer Endlosschleife die "virtuellen" Eingänge E22/23/24 und E26 des Terminal-Bausteins abgefragt. Bei E26 = 1 geht’s zurück zu HOME, das auch beim Start als erstes durchlaufen wurde. E22/23/24 setzen die zugehörende Positionsvariable auf einen Maximalwert und starten dann das entsprechende DRIVETO – immer wieder.

Die weiteren Abläufe fragen E2/23/24 auf 0 ab, wenn das zutrifft wurde die Maus wieder freigegeben und der zugehörende Eingang ging auf 0. In diesem Fall wird VAR2/4/6 = 0 gesetzt um das laufende DRIVETO zu beenden (der entsprechende Postions-Baustein bekommt dann eine Obergrenze 0, die er sofort auswertet.).

Hier wird im Interesse der Übersichtlichkeit eine einfache Lösung vorgestellt, die nur ein Positionieren in eine Richtung zuläßt (Die Eingänge des Terminal-Bausteins sind begrenzt). Man könnte aber z.B. E17 als Indikator für die Richtung hernehmen (gedrückt = rückwärts) und dann VAR2/4/6 auf 0 setzen und die anderen Abläufe entsprechend korrigieren ...

Arbeit nach Plan



Hier werden in einer Endlosschleife zwei unterschiedliche Postionen angefahren :

- Zunächstmal beim Start auf HOME
- Dann die VAR2/4/6 mit den gewünschten Positionswerten besetzen, sie wurden z.B. im vorhergehenden Projekt gewonnen.
- Und mit MOVETO ab nach Position E
- GREIFER zu – gelbe Tonne schnappen
- VAR2/4/6 mit neuen Positionswerten versorgen – MOVETO Position A
- GREIFER auf – Tonne fallen lassen
- freudig Beepen und zurück auf E.

Anfangs kann man den Robot ja von Hand füttern, später wird man sicher die Entnahme- und die Abgabe-Station mit fischertechnik-Teilen noch ausbauen.
Wenn das langweilig wird, kann man natürlich auch noch Zwischenstationen einbauen ...

Anhang

Modell, Programme

Interfacebelegung

M1	Säulenmotor	Endtaster E1 : links, Impulszähler : E2
M2	Motor Arm horizontal	Endtaster E3 : hinten, Impulszähler : E4
M3	Motor Arm vertikal	Endtaster E5 : oben, Impulszähler : E6
M4	Greifermotor	Endtaster E7 : offen, Impulszähler : E8
E1	Endtaster Säule	
E2	Impulszähler Säule	akt. Position Z2, Zielposition VAR2
E3	Endtaster Arm horizontal	
E4	Impulszähler Arm horizontal	akt.Position Z4, Zielposition VAR4
E5	Endtaster Arm vertikal	
E6	Impulszähler Arm vertikal	akt. Position Z6, Zielposition VAR6
E7	Endtaster Greifer	
E8	Impulszähler Greifer	akt. Position Z8

LLWin Parameter, Variable und Schalter

EA	Zielposition Säule (Impulse ab links)
EB	Zielposition Arm horizontal (Impulse ab hinten)
EC	Zielposition Arm vertikal (Impulse ab oben)
ED	Öffnung Greifer (Impulse ab geschlossen)
E17	Zyklus-Stop
Z2	akt. Position Säule
Z4	akt. Position Arm horizontal
Z6	akt. Position Arm vertikal
Z8	akt. Position Greifer
VAR2	Zielposition Säule
VAR4	Zielposition Arm horizontal
VAR6	Zielposition Arm vertikal
VAR98	Ablaufsynchronisation MoveTo
VAR99	Ablaufsynchronisation Home

Projekte

SAU01	Home Säule pur
SAU02	Home Säule mit UP
SAU03	Säule + DriveTo (absolute Positionierung über Variable)
SAU04	Säule + Greifer (UP mit zwei Eingängen)
SAU05	Säulenposition über Terminal
SAU06	Säule einfacher Zyklus
SAU10	Home/MoveTo Gesamt-Robot
SAU11	TeachIn
SAU12	Run

Die angeführten Projekte sind in www.ftcomputing.de/zip/llwin30.zip zu finden.