INHALTSVERZEICHNIS

1 Einleitung – fischertechnik Modelle steuern mit ROBO Pro

Sicherlich hast du dich auch schon manchmal gefragt, wie das funktioniert, wenn Roboter, wie von Geisterhand bewegt, bestimmte Aufgaben ausführen. Aber nicht nur bei echten Robotern, in vielen anderen Bereichen begegnen wir der Steuerungs- und Automatisierungstechnik; auch bei fischertechnik. Bereits im übernächsten Kapitel werden wir gemeinsam ein kleines Steuerungsprogramm für ein automatisches Garagentor entwerfen und dabei lernen, wie man mit Hilfe der Software ROBO Pro für Windows, solche Steuerungsaufgaben lösen und testen kann. Die Bedienung von ROBO Pro ist dabei sehr einfach. Auf der grafischen Bedienoberfläche lassen sich die Steuerungsprogramme, genauer gesagt die Ablaufpläne und später Datenflusspläne, wie wir noch lernen werden, fast ausschließlich mit Hilfe der Maus erstellen.

Damit du deine fischertechnik Modelle über den PC ansteuern kannst, benötigst du neben der Steuerungssoftware ROBO Pro außerdem noch ein Interface als Bindeglied zwischen Rechner und Modell. Es wandelt die Befehle der Software so um, dass beispielsweise Motoren angesteuert und Signale von Sensoren verarbeitet werden können. Von fischertechnik gibt es das ROBO Interface Art.-Nr. 93293 und das ältere Intelligent Interface Art.-Nr. 30402. Beide Interfaces kannst du zusammen mit ROBO Pro verwenden. Allerdings unterstützt ROBO Pro nur den Onlinemodus des Intelligent Interface. Das alte parallele Interface Art.-Nr. 30520 wird von ROBO Pro nicht mehr unterstützt.

Noch ein paar Worte zum Aufbau dieses Handbuches. Es unterteilt sich in zwei Teile. Der erste Teil von Kapitel 1 bis Kapitel 4 beschreibt die grundsätzliche Vorgehensweise beim Programmieren mit ROBO Pro. Dabei bekommst du viele Informationen und Hintergründe zum Programmieren allgemein und zur Bedienungsweise der Software ROBO Pro.

Der zweite Teil umfasst die Kapitel 5 bis 7 und führt in die Funktionen für fortgeschrittene Programme ein.

Die Kapitel ab Kapitel 8 sind eher ein Nachschlagewerk. Wenn du also nach dem ersten Teil mit der Bedienung von ROBO Pro vertraut bist und ganz spezielle Informationen suchst, findest du dort die ausführliche Erklärung zu den einzelnen Programmelementen.

Nun aber los! Sicherlich bist du schon sehr gespannt, welche Möglichkeiten du mit der Software ROBO Pro zum Programmieren deiner fischertechnik Modelle hast. Viel Spaß!

1.1 Installation von ROBO Pro

Voraussetzungen zur Installation von ROBO Pro sind:

- ein IBM-kompatibler PC mit Pentium-Prozessor mit mindestens 600 MHz Taktfrequenz, 32 MB RAM und ca. 20 MB freier Speicherkapazität auf der Festplatte
- Ein Monitor und eine Grafikkarte mit einer Auflösung von mindestens 1024x768 Bildpunkten. Bei Monitoren mit Bildröhre sollte die Bildwiederholrate bei mindestens 85 Hertz liegen um ein flimmerfreies Bild zu erhalten. TFT Flachbildschirme liefern bei jeder Bildwiederholrate ein flimmerfreies Bild, so dass bei einem TFT Flachbildschirmen die Bildweiderholrate unerheblich ist.
- Microsoft, Windows, Version Windows 95, 98, ME, NT4.0, 2000 oder XP
- Eine freie USB-Schnittstelle oder eine freie RS232-Schnittstelle COM1 bis COM4 zum Anschluss des ROBO Interface — Art.-Nr. 93293 — oder eine freie RS232-Schnittstelle COM1 bis COM4 zum Anschluss des älteren Intelligent Interface — Art.-Nr. 30402

Zunächst musst du natürlich den Computer starten und warten, bis das Betriebssystem (Windows) vollständig geladen ist. Das ROBO–Interface sollte erst nach erfolgreicher Installation an den Computer angeschlossen werden. Lege nun die Installations-CD in das CD-ROM Laufwerk ein. Das Installationsprogramm auf der CD wird dann automatisch gestartet.

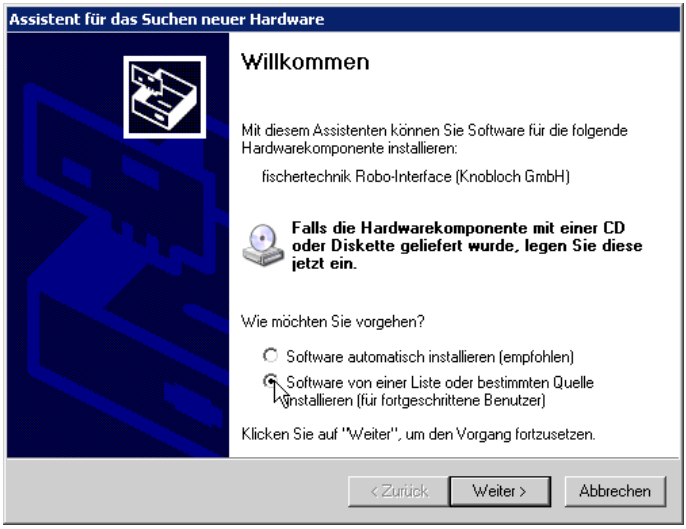
- Im ersten Willkommen-Fenster des Installationsprogramms betätigst du den **Weiter** Knopf.
- Das zweite Fenster **Wichtige Hinweise** enthält wichtige aktuelle Hinweise zur Installation des Programms oder zum Programm selbst. Auch hier betätigst du den **Weiter** Knopf.
- Im dritten Fenster **Lizenzvereinbarungen** ist der Lizenzvertrag zu ROBO Pro wiedergegeben. Du musst den Lizenzvertrag mit **Ja** akzeptieren, bevor du mit **Weiter** zum nächsten Fenster gehen kannst.
- Im folgenden Fenster **Anwenderinformationen** gibst du bitte deinen Namen ein.
- Im Fenster **Installationstyp** kannst du zwischen der **Expressinstallation** und einer **Benutzerdefinierten Installation** wählen. Bei der benutzerdefinierten Installation kannst du einzelne Komponenten von der Installation ausschließen. Wenn du eine neue Version von ROBO Pro über eine ältere Version installierst und einige der Beispielprogramme der älteren Version verändert hast, kannst du in der benutzerdefinierten Installation die Beispiele von der Installation ausschließen. Anderenfalls werden veränderte Beispielprogramme bei der Installation ***ohne Warnung überschrieben***. Wenn du die benutzerdefinierte Installation auswählst und **Weiter** drückst, erscheint ein zusätzliches Fenster, in dem du die Komponenten auswählen kannst.
- Im Fenster **Zielverzeichnis** kannst du den gewünschten Ordner bzw. Verzeichnispfad auswählen, in welchem das Programm ROBO Pro installiert werden soll. Normalerweise ist dies der Pfad C:\Programme\ROBO Pro. Du kannst aber auch ein anders Verzeichnis eingeben.
- Wenn du im letzten Fenster auf **Fertig stellen** drückst, wird die Installation durchgeführt. Sobald die Installation abgeschlossen ist — das dauert normalerweise nur wenige Sekunden — meldet das Programm die erfolgreiche Installation. Wenn es Probleme gibt, wird eine Fehlermeldung angezeigt, die dir helfen sollte das Problem zu lösen.

1.2 Installation des USB-Treibers für das ROBO–Interface

Dieser Schritt ist nur erforderlich, wenn das neue ROBO–Interface an der USB-Schnittstelle angeschlossen werden soll. Das ROBO–Interface kann auch an die serielle Schnittstelle COM1–COM4 angeschlossen werden. Für das ältere Intelligent Interface ist dieser Schritt nicht erforderlich, da das Intelligent Interface ausschließlich seriell angeschlossen werden kann. Die älteren Windows Versionen Windows 95 und Windows NT 4.0 unterstützen die USB-Schnittstelle nicht. Bei Verwendung von Windows 95 oder NT 4.0 kann das ROBO–Interface nur über die serielle Schnittstelle angeschlossen werden. Ein Treiber braucht dann ebenfalls nicht installiert zu werden.

Wichtiger Hinweis für die Installation unter Windows 2000 und Windows XP:
Der USB-Treiber kann nur von einem Anwender installiert werden, der an dem PC Administratorrechte besitzt. Sollte das Installationsprogramm melden, dass du den USB-Treiber nicht installieren darfst, musst du entweder deinen Systemadministrator bitten, den Treiber zu installieren oder ROBO Pro ohne diesen Treiber installieren. Dann kannst du dein Interface aber nur über die etwas langsamere serielle Schnittstelle anschließen.

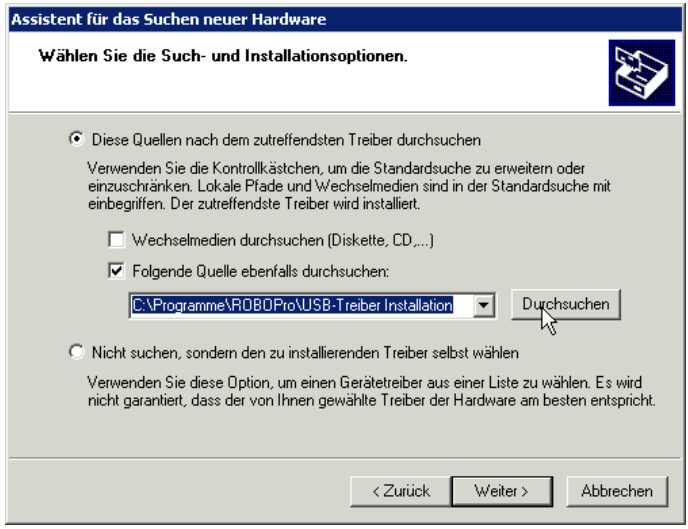
Um den USB-Treiber zu installieren musst du zunächst das ROBO–Interface mit einem USB-Kabel an deinen Computer anschließen und mit Strom versorgen. Windows erkennt automatisch, dass das Interface angeschlossen ist und zeigt folgendes Fenster an:



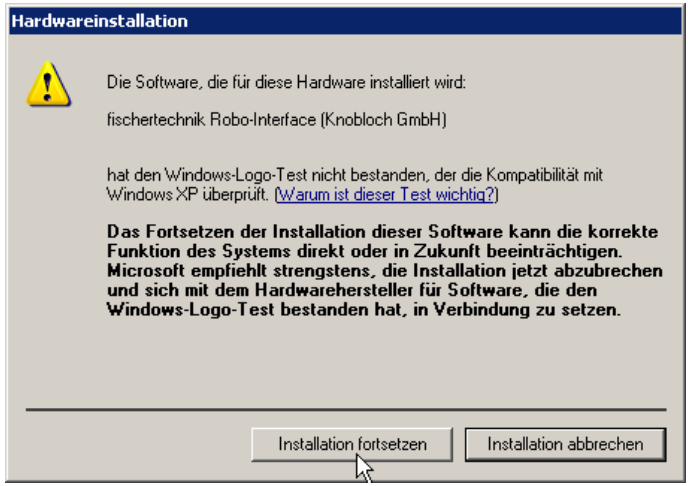
Das Fenster kann je nach Betriebssystem etwas anders aussehen als oben abgebildet!

Hier musst du **Software von einer Liste oder bestimmten Quelle installieren** auswählen und **Weiter** drücken.

Im folgenden Fenster deaktivierst du **Wechselmedien durchsuchen** und aktivierst **Folgende Quellen ebenfalls durchsuchen**. Dann drückst du auf **Durchsuchen** und wählst das Unterverzeichnis ***USB-Treiber Installation*** in dem Verzeichnis, in dem ROBO Pro installiert ist (das Standardverzeichnis ist C:\ROBOPro\):

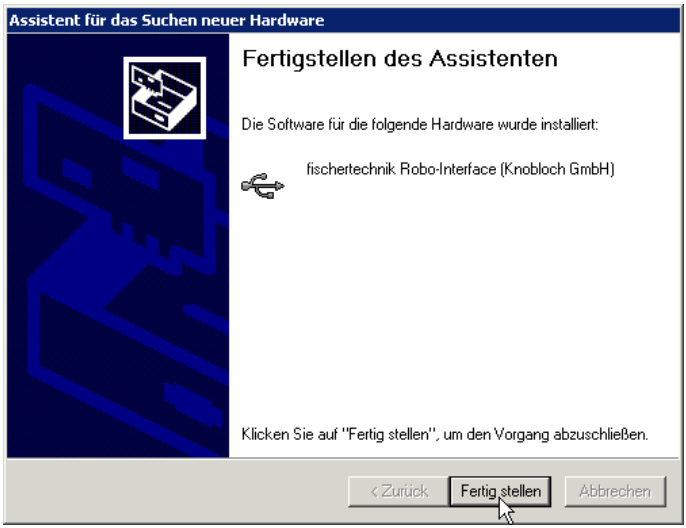


Nachdem du auf **Weiter** gedrückt hast, erscheint unter Windows XP möglicherweise folgende Meldung:



Der USB-Treiber wird noch von Microsoft überprüft. Sobald die Überprüfung abgeschlossen ist wird der Treiber von Microsoft signiert, so dass diese Meldung nicht mehr erscheint. Um den Treiber zu installieren drücke auf **Installation fortsetzen**.

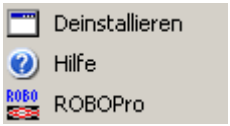
Schließlich erscheint folgende Meldung:



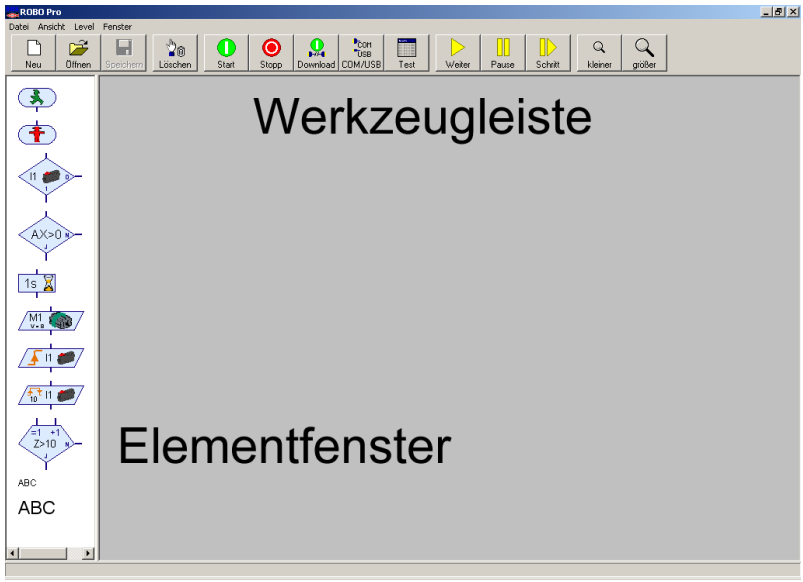
Drücke auf **Fertig stellen** um die USB-Treiber Installation abzuschließen.

1.3 Erste Schritte

Neugierig? Dann starte doch einfach mal das Programm ROBO Pro. Hierfür klickst du auf den Startknopf in der Taskleiste und wählst anschließend ***Programme*** oder ***Alle Programme*** und ***ROBO Pro***. In diesem Ordner des Startmenüs findest du folgende Einträge:



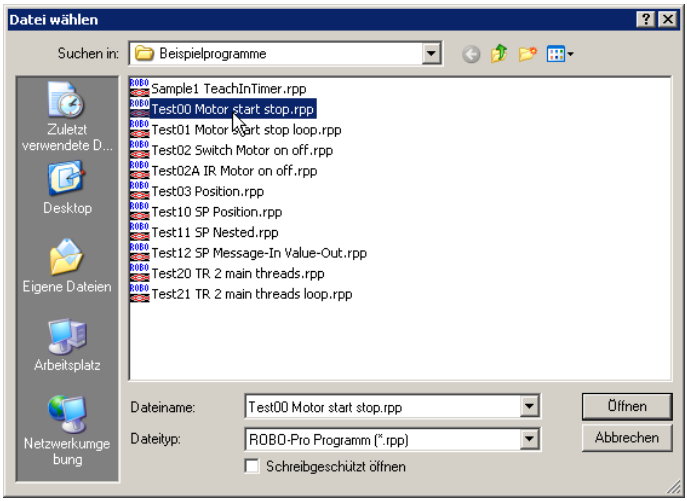
Mit dem Deinstallieren Eintrag kannst du ROBO Pro deinstallieren. Der Hilfe Eintrag öffnet die Hilfedatei zu ROBO Pro und der ROBO Pro Eintrag öffnet das ROBO Pro Programm. Wähle nun den Eintrag **ROBO Pro** aus um die Software zu starten.



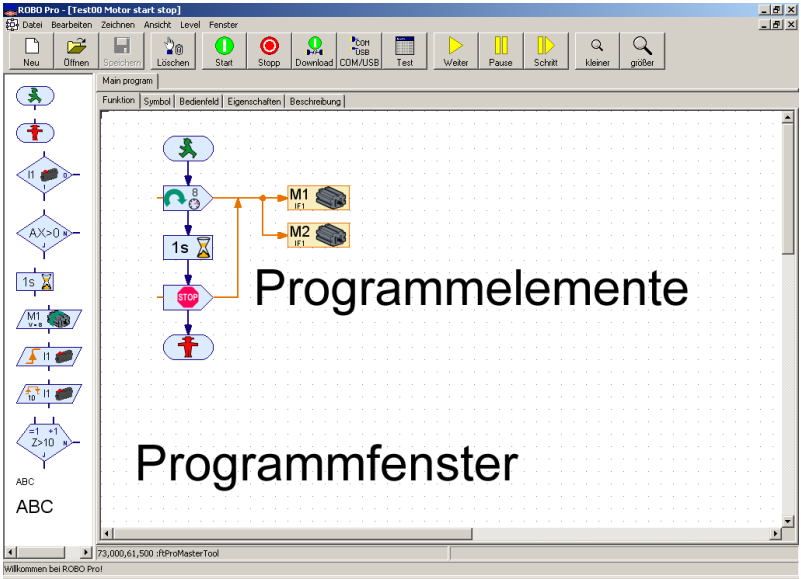
Das Fenster hat oben eine Menüleiste und eine Werkzeugleiste mit verschiedenen Bedienknöpfen sowie auf der linken Seite ein Fenster mit Programmelementen. Wenn Du in der linken Randspalte zwei Fenster übereinander siehst, ist ROBO Pro nicht auf **Level 1** eingestellt. Um die Funktionalität von ROBO Pro an den wachsendes Wissen anzupassen, kannst du ROBO Pro auf Level 1 für Einsteiger bis Level 5 für Experten einstellen. Kontrolliere nun im Menü **Level**, ob bei **Level 1: Einsteiger** ein Haken ist. Falls nicht, schalte bitte auf Level 1 um.


Öffnen

Du hast nun die Möglichkeit, entweder eine neue Programmdatei zu erstellen, oder aber eine bereits existierende Programmdatei zu öffnen. Eine neue Programmdatei wollen wir erst im Kapitel 3 erstellen, wenn wir unser erstes Steuerungsprogramm anlegen. Um die Bedienoberfläche kennen zulernen, öffnen wir eines der bereits existierenden Beispielprogramme. Dazu klickst du im Menü **Datei** auf den Eintrag **Öffnen** oder verwendest den **Öffnen** Knopf in der Werkzeugleiste. Die Beispieldateien befinden sich im Verzeichnis ***C:\Programme\ROBO Pro\Beispielprogramme***.



Öffne die Datei **Test00 Motor start Stopp.rpp**:



Hier kannst du sehen, wie ein einfaches ROBO Pro Programm aussieht. Mit den Programmelementen aus dem Elementfenster werden beim Programmieren in dem Programmfenster die Ablaufpläne der Steuerungsprogramme erstellt. Die fertigen Ablaufpläne können dann überprüft

und mit einem angeschlossenen fischertechnik Interface getestet werden. Aber immer mit der Ruhe, wir werden in den nächsten Kapiteln schrittweise das Programmieren kennen lernen! Nachdem du so einen ersten Eindruck von der Bedienoberfläche bekommen hast, schließt du die Programmdatei über den Befehl **Beenden** im Menü **Datei** wieder. Die Abfrage, ob du die Datei speichern möchtest, kannst du mit **Nein** beantworten.

2 Vor der Programmierung ein kurzer Test der Hardware

Damit wir die Steuerungsprogramme, die wir später erstellen werden, auch testen können, muss das Interface an den PC angeschlossen werden, das ist klar. Aber je nach verwendetem Interface (ROBO–Interface Art.-Nr. 93293 oder älteres Intelligent Interface Art.-Nr. 30402) muss auch die Software entsprechend eingestellt und die Verbindung getestet werden. Dies wollen wir im folgenden Kapitel tun.

2.1 Anschluss des Interface an den PC

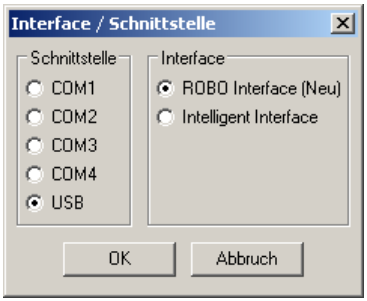
Dies dürfte kein größeres Problem sein. Das mit dem Interface mitgelieferte Verbindungskabel wird am Interface und an einer Schnittstelle des PCs angeschlossen:

- Beim ROBO–Interface (Art.-Nr. 93293) kann eine USB Schnittstelle oder eine serielle Schnittstelle COM1 bis COM4 verwendet werden.
- Beim Intelligent Interface (Art.-Nr. 30402) muss eine serielle Schnittstelle COM1 bis COM4 verwendet werden.

Die Anschlüsse dieser Schnittstellen befinden sich in der Regel auf der Rückseite deines Computers. Die genaue Lage der verschiedenen Anschlüsse ist in der Bedienungsanleitung deines PCs genau beschrieben, bitte dort nachlesen. USB Anschlüsse finden sich häufig auch an der Frontseite des PCs. Vergiss nicht, das Interface mit Strom zu versorgen (Netzgerät oder Akku). Die einzelnen Anschlüsse des Interfaces sind in der Bedienungsanleitung des jeweiligen Interfaces genau beschrieben.

2.2 Damit die Verbindung stimmt – die Interfaceeinstellungen

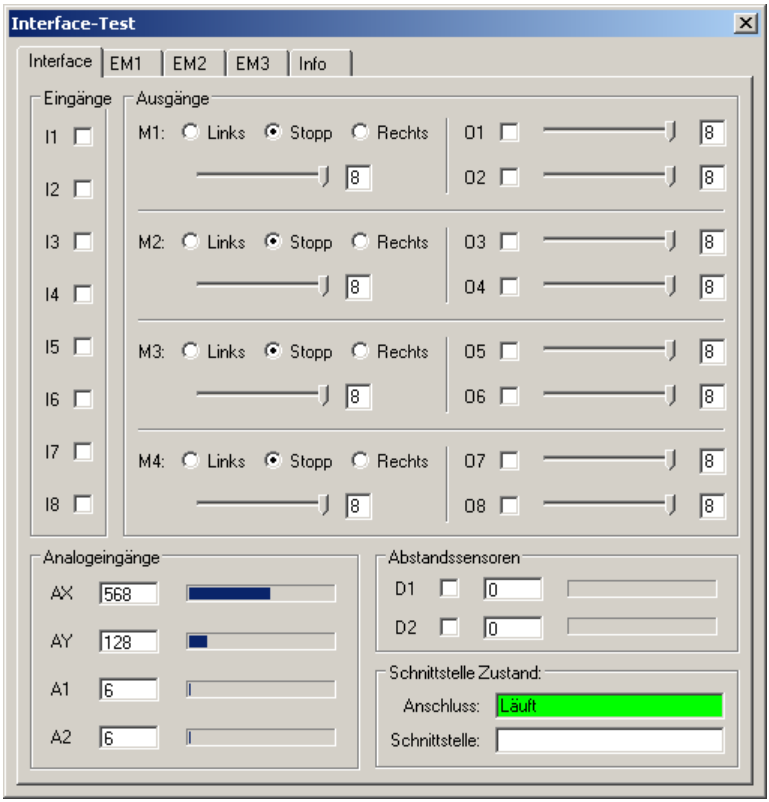
Damit die Verbindung von PC und Interface korrekt funktioniert, muss das jeweils verwendete Interface in ROBO Pro eingestellt werden. Starte dazu ROBO Pro über den Eintrag **ROBO Pro** im Startmenü unter ***Programme*** oder ***Alle Programme*** und ***ROBO Pro***. Drücke dann in der Werkzeugleiste auf den Knopf **COM/USB**. Es erscheint folgendes Fenster:



Hier kannst du sowohl die Schnittstelle als auch den Interfacetyp auswählen. Das Intelligent Interface unterstützt wie bereits erwähnt nur die seriellen Schnittstellen COM1 – COM4.


Test

Nachdem du die richtigen Einstellungen vorgenommen hast schließe das Fenster mit OK. Öffne nun das Fenster für den Interface-Test mit dem **Test** Knopf in der Werkzeugleiste:



Es zeigt die am Interface vorhandenen Eingänge und Ausgänge. Der grüne Balken unten rechts zeigt den Verbindungsstatus vom PC zum Interface an:

- **Verbindung zum Interface OK** bestätigt eine korrekte Verbindung zum Interface
- **Keine Verbindung zum Interface** deutet darauf hin, dass die Verbindung nicht korrekt eingestellt wurde und der PC keine Verbindung zum Interface aufbauen konnte. Der Balken erscheint dann in roter Farbe.

Um die Interface- oder die Verbindungseinstellungen verändern zu können, musst du das Test Fenster schließen (mit dem X oben rechts) und wie zuvor beschrieben über den COM/USB Knopf in der Werkzeugleiste eine andere Schnittstelle oder einen anderen Interfacetyp auswählen.

Wenn du so die Verbindung von PC und Interface einstellen konntest und im **Test** Fenster der grüne Balken erscheint, kannst du das nächste Kapitel getrost überspringen.

Wenn nicht, können dir vielleicht die Tipps im nächsten Abschnitt weiterhelfen.

2.3 Falls die Verbindung nicht stimmt –keine Verbindung zum Interface!?

Falls beim ROBO oder Intelligent Interface trotz korrekt eingestellter serieller COM-Schnittstelle (s. oben) die Meldung **Keine Verbindung zum Interface** erscheint, solltest du nachfolgende Punkte überprüfen. Eventuell musst du hierfür auch einen "Computerkenner" zu Rate ziehen:

- ***Stromversorgung:***
  Wird das Interface richtig mit Strom versorgt? Verwendest du als Stromversorgung Batterien oder Akkus, dann besteht die Möglichkeit, dass leere Akkus nicht mehr genug Spannung liefern. Sinkt die Spannung der Batterie unter 6 V, funktioniert der Prozessor des ROBO Interfaces nicht mehr. In diesem Fall leuchtet entweder die rote Leuchtdiode dauerhaft oder gar keine Leuchtdiode. Wenn die Spannung in Ordnung ist blinken einige der grünen Leuchtdioden.
  Beim älteren Intelligent Interface kann man anhand der Leuchtdioden nicht einfach feststellen, ob die Spannung für den Prozessor ausreicht. Wenn die Spannung zu gering ist musst du die Akkus neu laden bzw. neue Batterien verwenden, oder aber das Interface falls möglich mit einem Netzgerät testen.
- ***Funktioniert die Schnittstelle überhaupt?***
  Dies kannst du herausfinden, indem du ein anderes serielles Gerät wie z.B. ein externes Modem an der Schnittstelle testest.
- Gibt es einen Konflikt mit einem anderen Gerätetreiber an derselben Schnittstelle (z. B. Modem)? Eventuell muss dieser Treiber deaktiviert werden (siehe Windows- oder Gerätehandbuch).
- Nur für Windows NT/2000/XP und das ältere Intelligent Interface: Ist ein älteres Intelligent Interface beim Hochfahren des PCs bereits mit dem Rechner und der Stromversorgung verbunden, wird es von Windows NT unglücklicherweise in den Download-Modus umgeschaltet. Um die Verbindung zum PC wieder herzustellen musst du einfach die Stromversorgung am Interface kurz unterbrechen. Beim neuen ROBO Interface kann das nicht passieren.
- Falls du immer noch keine Verbindung zum Interface herstellen kannst, ist wahrscheinlich das Interface oder das Verbindungskabel defekt. In diesem Fall wendest du dich an den fischertechnik Service (Adresse: siehe Menü: **„?“** / **Info Über**).

2.4 Funktioniert alles – der Interface-Test


Test

Nachdem die Verbindung korrekt eingestellt ist, können wir mit Hilfe des Interfacetests das Interface selbst und daran angeschlossene Modelle testen. Wie bereits erwähnt, zeigt das Testfenster die verschiedenen Eingänge und Ausgänge des Interfaces:

- ***Digitaleingänge I1–I8***
  I1-I8 sind die Digitaleingänge des Interfaces. Hier werden so genannte Sensoren angeschlossen. Digitaleingänge können nur die Zustände 0 und 1 oder Ja und Nein annehmen. An die Digitaleingänge können als Sensoren Schalter (Minitaster) aber auch Fototransistoren (Lichtsensoren) oder Reedkontakte (Magnetsensoren) angeschlossen werden.
  Du kannst die Funktion dieser Eingänge überprüfen, indem du am Interface z. B. an I1 einen Mini-Taster (Art.-Nr. 37783) anschließt. (verwende am Taster die Kontakte 1 und 3). Sobald du den Taster drückst, erscheint in der Anzeige von I1 ein Häkchen. Hast du den Taster anders herum angeschlossen (Kontakte 1 und 2), erscheint sofort das Häkchen und verschwindet bei Tastendruck

- ***Motorausgänge M1–M4***
  M1 – M4 sind die Ausgänge des Interfaces. Hier werden die so genannten Aktoren angeschlossen. Dies können z. B. Motoren, Elektromagneten oder Lampen sein. Die 4 Motorausgänge lassen sich sowohl in der Geschwindigkeit in 8 Stufen als auch in der Richtung steuern. Zur Steuerung der Geschwindigkeit dient der Schieberegler. Zudem wird die Geschwindigkeit neben dem Schieberegler als Zahl angezeigt. Wenn du einen Ausgang testen möchtest, schließt du an diesen Ausgang, z. B. M1, einen Motor an.
- ***Lampenausgänge O1–O8***
  Die Motorausgänge lassen sich alternativ auch als ein Paar von Einzelausgängen verwenden. Damit kann man Lampen, aber auch Motoren, die sich nur in eine Richtung bewegen müssen, (z.B. Förderband) ansteuern. Wenn du einen dieser Ausgänge testen willst, schließt du einen Anschluss der Lampe an den Ausgang, z. B. O1 an. Den anderen Anschluss der Lampe verbindest du mit der Massebuchse des Interfaces (⊥).
- ***Analogeingänge AX–AY***
  Die Analogeingänge AX und AY messen den Widerstand des angeschlossenen Sensors. Hier lassen sich NTC–Widerstände zur Temperaturmessung, Potentiometer, Fotowiderstände oder Fototransistoren anschließen.
- ***Analogeingänge A1–A2***
  Diese beiden Eingänge messen eine Spannung von 0–10V.
- ***Abstandssensoren D1–D2***
  And die Abstandssensoreingänge D1 und D2 lassen sich nur spezielle Abstandssensoren anschließen. Die Abstandssensoren D1 und D2 liegen sowohl als Digitaleingänge als auch als Analogeingänge vor.
- ***Extensionmodule EM1–EM3***
  Sofern an das Interface Erweiterungsmodule angeschlossen sind (bis zu drei I/O-Extension Art.-Nr. 93294 beim ROBO Interface, aber höchstens ein Extension Module Art.-Nr. 16554 beim Intelligent Interface), lassen diese sich steuern, indem über das Register am oberen Rand auf die Erweiterungsmodule umgeschaltet wird.

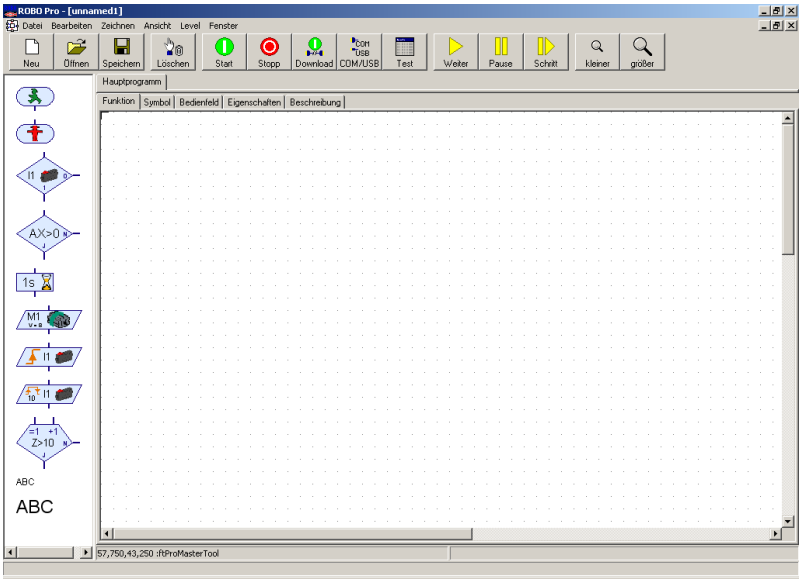
3 Level 1: Dein erstes Steuerungsprogramm

Nachdem du im letzten Kapitel die Hardware, also das Interface und daran angeschlossene Schalter und Motoren, getestet hast, wollen wir uns nun mit dem Programmieren befassen. Was aber bedeutet "Programmieren" eigentlich? Nun stell dir einmal vor, dass an unserem Interface z. B. ein Roboter angeschlossen ist. Dieser Roboter ist aber so dumm, dass er von alleine nicht funktioniert. Zum Glück sind wir da ein bisschen schlauer. Wir können dem Roboter ganz genau sagen, was er tun soll. Wie? Nun, was passierte, als wir im letzten Kapitel mit der Maustaste den Motorausgang M1 auf „Links“ gestellt haben? Richtig, wir haben den Motor eingeschaltet. Würde dieser Motor z. B. die Greifzange unseres Roboters bewegen, hätten wir nichts anderes getan, als dem Roboter gesagt: "Greife den Gegenstand!" Nun wollen wir aber nicht jeden Schritt von Hand auslösen, sondern der Roboter soll dies "automatisch" tun. Dazu müssen wir die einzeln auszuführenden Schritte so abspeichern, dass der Roboter sie nacheinander abarbeiten kann, d. h. wir müssen ein Programm erstellen, welches an unserer Stelle den Roboter steuert. In der Fachsprache nennt man das dann logischerweise ein Steuerungsprogramm.

3.1 Ein neues Programm erstellen


Neu

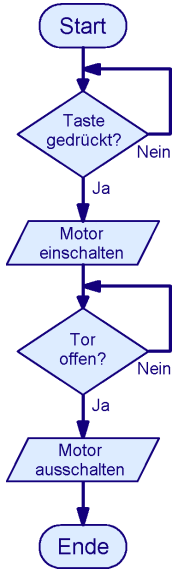
Mit der Software ROBO Pro haben wir nun ein tolles Werkzeug zur Hand, um solche Steuerungsprogramme zu entwerfen und mit Hilfe eines angeschlossenen Interfaces zu testen. Keine Angst, wir wollen nicht gleich einen Roboter programmieren. Wir begnügen uns zunächst mit einfachen Steuerungsaufgaben. Hierzu müssen wir ein neues Programm erstellen. In der Werkzeugleiste findest du den Eintrag „Neu“. Wenn du mit der linken Maustaste darauf klickst, wird ein neues leeres Programm erstellt:



Du siehst nun eine große weiße Zeichenfläche, in die du gleich dein erstes Programm eingeben wirst. Falls du am linken Rand zwei Fenster übereinander siehst, stelle bitte im Menü **Level** auf **Level 1: Einsteiger** um.

3.2 Die Elemente des Steuerungsprogramms

Nun können wir uns an die Aufgabe machen, unser erstes Steuerungsprogramm zu erstellen. Dieses wollen wir anhand eines konkreten Beispiels tun:

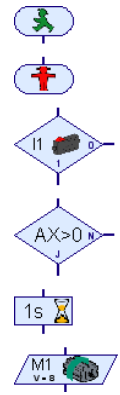


Funktionsbeschreibung:

Stell Dir ein Garagentor vor, das sich automatisch öffnen lässt. Vielleicht besitzt ihr sogar ein solches zu Hause! Kommt man mit dem Auto zur Garage, genügt ein Tastendruck beim Sender und das Garagentor wird, von einem Motor angetrieben, geöffnet. Der Motor muss so lange laufen, bis das Garagentor vollständig geöffnet ist.

Nun ist es recht umständlich und auch nicht sehr anschaulich, eine Steuerung in Worten zu beschreiben. Deshalb verwendet man für die Darstellung der nacheinander auszuführenden Aktionen und die Bedingungen, die für diese Aktionen erfüllt sein müssen, so genannte ***Ablaufpläne***. Die Bedingung für die Aktion "Einschalten des Motors" ist im Fall unserer Steuerung, dass der Taster gedrückt wird. Das Lesen eines solchen Ablaufplanes ist ganz einfach: Immer schrittweise den Pfeilen nach! Diese ergeben dann die genaue Funktionsweise der Steuerung – die einzelnen Schritte können nur in der durch die Pfeile vorgegebenen Reihenfolge ausgeführt werden, niemals anders. Sonst bräuchten wir uns die ganze Arbeit nicht zu machen – oder?

Mit Hilfe unserer Software ROBO Pro können wir nun genau diesen Ablaufplan zeichnen und damit das ***Steuerungsprogramm*** für die angeschlossene Hardware (Interface, Motoren, Schalter, etc.) erstellen. Den Rest übernimmt die Software, was im Übrigen bei großen, industriellen Anwendungen auch nicht anders ist! Damit können wir uns ganz auf die Erstellung des Ablaufplans konzentrieren.



Den Ablaufplan setzt du aus Programmelementen zusammen. Wieder ein neuer Begriff? Halb so wild! In ROBO Pro werden die einzelnen Elemente, mit denen der Ablaufplan erstellt wird, Programmelemente genannt. Die Aktion "Motor einschalten" heißt doch nichts anderes, als dass das Interface tatsächlich diesen Motor, der am Interface angeschlossen ist, einschalten soll! Die verfügbaren Programmelemente findest du im Elementfenster am linken Rand.

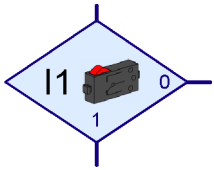
3.3 Programmelemente einfügen, verschieben und ändern

Jetzt geht es darum, mit den im Elementfenster enthaltenen Programmelementen den Ablaufplan für unsere Garagentorsteuerung zu erstellen. Alle verfügbaren Programmelemente können aus dem Elementfenster geholt und im Programmfenster eingefügt werden.

Einfügen von Programmelementen

Dazu bewegst du die Maus auf das Symbol des gewünschten Programmelements und klickst mit der linken Maustaste einmal darauf. Dann bewegst Du die Maus in das Programmfenster (das ist die große weiße Fläche) an die gewünscht Stelle und klickst noch einmal. Du kannst das Programmelement auch

mit gedrückter Maustaste aus dem Elementfenster in das Programmfenster ziehen. Ein Programm beginnt immer mit einem Startelement. Das Startelement ist das abgerundete Element mit dem grünen Ampelmännchen. Am besten probierst du es gleich mal mit diesem Programmelement aus: Klicke einmal mit der linken Maustaste auf das Startelement im Programmfenster, bewege die Maus oben in das Programmfenster und klicke dort noch einmal die linke Maustaste.



Als nächstes kommt im Programmlaufplan ein Element, das einen Eingang abfragt und je nach dem Zustand des Eingangs in einen oder einen anderen Programmpfad verzweigt. Klicke im Elementfenster auf das rechts abgebildete Element und bewege dann die Maus unter das zuvor eingefügte Startelement. Wenn der obere Eingang des Verzweigungselements ein oder zwei Rasterpunkte unter dem Ausgang des Startelements steht, erscheint im Programmfenster eine Verbindungslinie. Wenn Du nun noch mal die linke Maustaste klickst, wird das Verzweigungselement eingefügt und automatisch mit dem Startelement verbunden.

Verschieben von Programmelementen und von Gruppen

Ein Programmelement lässt sich auch nach dem Einfügen bei gedrückter linker Maustaste an die gewünschte Stelle verschieben. Wenn Du mehrere Elemente zusammen verschieben möchtest, kannst du zunächst mit gedrückter linker Maustaste einen Rahmen um die Elemente aufziehen. Du musst dazu die linke Maustaste in einem ***leeren*** Bereich klicken, die Taste gedrückt halten und mit der Maus ein Rechteck aufziehen, das die gewünschten Elemente enthält. Die Elemente im Rechteck werden nun mit einem roten Rand dargestellt. Wenn Du nun eines der roten Elemente mit der linken Maustaste verschiebst, werden alle roten Elemente mit verschoben. Du kannst auch einzelne Elemente rot markieren, indem du mit der linken Maustaste bei gedrückter Umschalttaste (das ist die Groß-Kleinschreibtaste) auf die Elemente klickst. Wenn du mit der linken Maustaste in einen leeren Bereich klickst, werden alle rot markierten Elemente wieder normal dargestellt.

Kopieren von Programmelementen und von Gruppen

Zum Kopieren von Programmelementen gibt es zwei Möglichkeiten. Du kannst genauso wie beim Verschieben vorgehen, aber bevor du die Elemente verschiebst, drückst du die **STRG** Taste an der Tastatur. Dadurch werden die Elemente nicht verschoben, sondern kopiert. Mit dieser Funktion kannst du Elemente aber nur innerhalb eines Programms kopieren. Wenn du Elemente von einem Programm in ein anderes kopieren möchtest, kannst du die ***Zwischenablage*** von Windows verwenden. Wähle zunächst einige Elemente aus, so wie es im vorigen Abschnitt beim Verschieben von Elementen beschrieben ist. Wenn du nun **STRG+C** an der Tastatur drückst oder den Menüpunkt **Bearbeiten** / **Kopieren** aufrufst, werden alle ausgewählten Elemente in die Windows Zwischenablage kopiert. Du kannst nun zu einem anderen Programm wechseln und die Elemente dort mit **STRG+V** oder **Bearbeiten** / **Einfügen** wieder einfügen. Du kannst einmal kopierte Elemente auch mehrfach einfügen. Wenn du Elemente von einem Programm in ein anderes Programm verschieben möchtest, kannst du am Anfang statt **STRG+C** oder **Bearbeiten** / **Kopieren** die Funktion **STRG+X** oder **Bearbeiten** / **Ausschneiden** verwenden.

Löschen von Elementen und Rückgängig-Funktion

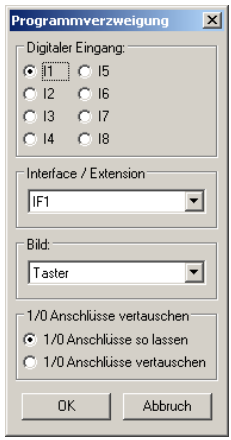

Löschen

Elemente zu löschen ist auch ganz einfach. Du kannst alle rot markierten Elemente (siehe vorheriger Abschnitt) löschen, indem du die Entfernen Taste (**Entf**) auf der Tastatur drückst. Du kannst auch einzelne Elemente mit der Löschfunktion löschen. Klicke dazu zunächst auf den abgebildeten Knopf in der Werkzeugleiste und dann auf das Element, das du löschen möchtest. Probiere es gleich einmal aus. Du kannst das gelöschte Element dann wieder neu zeichnen. Um das gelöschte

Element wieder zurück zu holen kannst du aber auch die Funktion **Rückgängig** im Menü **Bearbeiten** verwenden. Über diesen Menüpunkt kannst du alle Änderungen am Programm wieder rückgängig machen.

Eigenschaften von Programmelementen bearbeiten

Wenn du mit der ***rechten*** Maustaste auf ein Programmelement im Programmfenster klickst erscheint ein Dialogfenster, in dem du die Eigenschaften des Elements verändern kannst. Das Eigenschaftsfenster für ein Verzweigungselement ist rechts abgebildet.

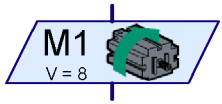


- Mit den Knöpfen **I1** bis **I8** kannst du eingeben welcher Eingang des Interface abgefragt werden soll.
- Die Auswahl **Interface / Extension** wird erst in Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51 erklärt.
- Unter **Bild** kannst du eine Bild für den am Eingang angeschlossenen Sensor auswählen. Digitaleingänge werden meistens mit Tastern verwendet, häufig aber auch mit Fototransistoren oder Reed-Kontakten.
- Unter 1**/0 Anschlüsse vertauschen** kannst du die Position der 1 und 0 Ausgänge der Verzweigung vertauschen. Normalerweise ist der 1 Ausgang unten und der 0 Ausgang rechts. Oft ist es aber praktischer wenn der 1 Ausgang rechts ist. Drücke auf **1/0 Anschlüsse vertauschen**, dann werden die 1 und 0 Anschlüsse vertauscht, sobald du das Fenster mit OK schließt.

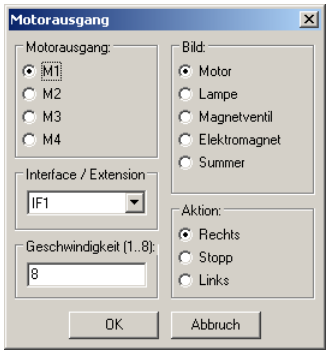
Hinweis: Wenn du einen Minitaster als Schließer am Anschluss 1 und 3 des Tasters anschließt, geht die Programmverzweigung zum 1 Zweig wenn der Schalter gedrückt ist, sonst zum 0 Zweig.

Wenn du einen Minischalter als Öffner am Anschluss 1 und 2 des Tasters anschließt, geht die Programmverzweigung zum 1 Zweig, wenn der Schalter nicht gedrückt ist, sonst zum 0 Zweig.

Das nächste Programmelement in unserer Garagentorsteuerung ist ein Motorelement. Füge es wie die beiden Elemente zuvor in das Programm ein, und zwar unter dem Verzweigungselement. Am besten platzierst du das Element wieder so, das es automatisch mit dem Element darüber verbunden wird.



Mit dem Motorelement kannst du sowohl einen Motor als auch eine Lampe oder einen Elektromagneten ein- oder ausschalten. Das Eigenschaftsfenster für das Motorelement öffnest du wieder mit einem rechten Mausklick auf das Element.



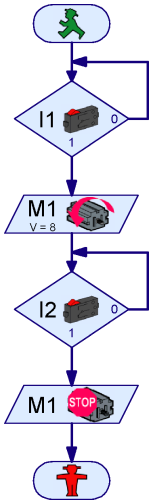
- Über die Knöpfe **M1** bis **M4** kannst du auswählen welcher Ausgang des Interface angesteuert werden soll.
- Unter **Bild** kannst du ein Bild auswählen, das den am Ausgang angeschlossenen fischertechnik Baustein darstellt.
- Die Auswahl **Interface / Extension** wird erst in Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51 erklärt.
- Unter **Aktion** kannst du auswählen, wie der Ausgang beeinflusst werden soll. Einen Motor kannst du mit Drehrichtung links oder rechts starten oder stoppen. Eine Lampe kannst du ein oder ausschalten.
- Unter **Geschwindigkeit/Intensität** kannst du einstellen, mit welcher Geschwindigkeit sich der Motor drehen soll, bzw. wie hell die Lampe leuchten soll. Mögliche Werte sind von 1 bis 8.

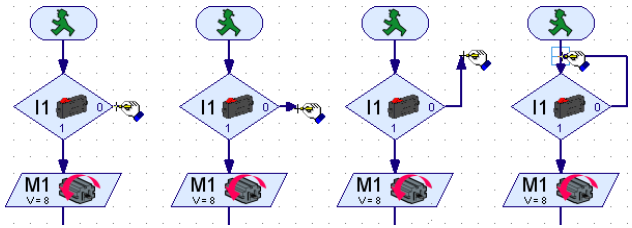
Für unseren Programmlaufplan benötigen wir den Befehl ***Motor M1 links mit Geschwindigkeit 8.***

3.4 Verbinden der Programmelemente

Nachdem du nun weißt, wie man Elemente in ein Steuerprogramm einfügt, können wir uns an die Aufgabe machen, unser Steuerungsprogramm fertig zu stellen. Denk mal zurück an die Funktionsbeschreibung der Garagentorsteuerung - fehlt da nicht noch etwas? Richtig, zwar haben wir den Motor per Tastendruck eingeschaltet, nachdem das Tor geöffnet ist, muss er aber wieder automatisch abgeschaltet werden! In der Praxis geschieht dies mit einem so genannten Endschalter. Das ist ein Taster, der so am Garagentor angebracht ist, dass er in dem Moment betätigt wird, in dem der Motor das Tor ganz geöffnet hat. Und wie beim Einschalten des Motors kann dieses Signal verwendet werden, um den Motor wieder auszuschalten. Für die Abfrage des Endschalters können wir wieder das Verzweigungselement verwenden.

Füge also in dein Steuerprogramm noch ein Verzweigungselement ein, das den Endschalter am Eingang I2 abfragt. Vergiss nicht mit der rechten Maustaste auf das Element zu klicken und den Eingang auf I2 umzustellen. Sobald das Garagentor offen und der Endschalter gedrückt ist, soll der Motor wieder anhalten. Dies wird über ein Motorelement erreicht. Verwende zunächst das gleiche Element wie zum Einschalten des Motors. Wenn du mit der rechten Maustaste auf das Element klickst, kannst du die Funktion des Elements auf **Motor stoppen** ändern. Abgeschlossen wird das Programm durch ein Ende-Element. Dein Programm sollte nun fast so wie rechts abgebildet aussehen. Wenn du die Elemente immer mit einem Abstand von ein oder zwei Rasterpunkten direkt untereinander platziert hast, sind die meisten Ein- und Ausgänge bereits durch Programmflusspfeile miteinander verbunden. Der Nein (N) Ausgang der beiden Verzweigungen ist aber noch nicht angeschlossen. Solange der Taster am Eingang I1 nicht gedrückt ist, soll das Programm wieder zurückgehen und den Schalter noch mal abfragen. Um diese Linie zu zeichnen, klicke mit der Maus nacheinander auf die im Bild unten gezeigten Stellen.





Hinweis: Sollte einmal eine Linie nicht korrekt mit einem Anschluss oder einer anderen Linie verbunden sein, wir dies durch ein grünes Rechteck an der Pfeilspitze dargestellt. In diesem Fall musst du die Verbindung durch Verschieben der Linie oder durch Löschen und neu zeichnen herstellen. Sonst funktioniert der Programmablauf an dieser Stelle nicht.

Löschen von Programmflusslinien

Das Löschen von Linien geht genau so wie das Löschen von Programmelementen. Klicke einfach mit der linken Maustaste auf die Linie, so dass sie rot markiert wird. Drücke nun die Entfernen Taste (**Entf**) auf der Tastatur, um die Linie zu löschen. Du kannst auch mehrere Linien auswählen, wenn du die Umschalttaste (das ist die Taste zur Umschaltung zwischen Groß- und Kleinbuchstaben) gedrückt hältst und dann nacheinander mit der linken Maustaste auf die Linien klickst. Außerdem kannst du zum Markieren mehrerer Linien auch einen Rahmen um diese Linien herum aufziehen. Nun kannst du alle rot markierten Linien durch drücken der **Entf** Taste auf einmal löschen.

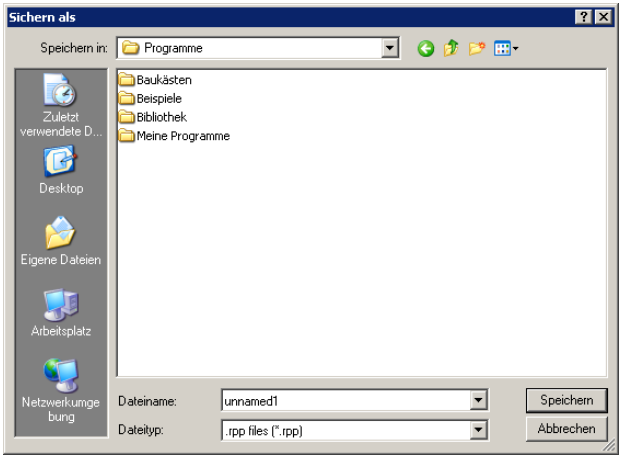
3.5 Testen des ersten Steuerungsprogramms

Um unser erstes Steuerungsprogramm testen zu können, solltest du ein kleines Modell aufbauen. Dazu genügt es, an das Interface an I1 und I2 einen Taster und an M1 einen Motor anzuschließen.

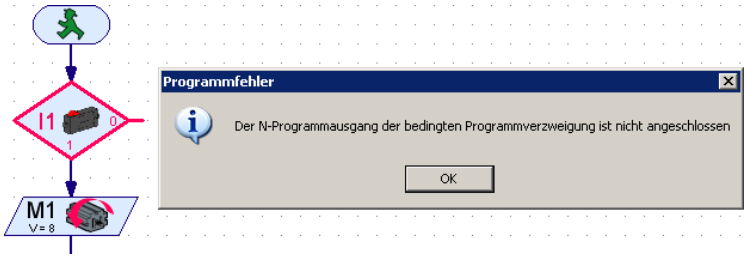
Hinweis: Der Anschluss des Interfaces an den PC und die Einstellung des Interfaces wurde bereits im vorigen Kapitel durchgeführt und kann dort nachgelesen werden.

Bevor du das Steuerungsprogramm testest, solltest du die Programmdatei auf der Festplatte deines Computers abspeichern. Klicke mit der Maus auf den Befehl **Speichern unter** im Menü **Datei**. Darauf erscheint folgendes Dialogfenster:

Wähle bei "Speichern in" das Verzeichnis, in dem du das Programm abspeichern willst. Gebe bei "Dateinamen" einen noch nicht vergebenen Namen ein, z. B. GARAGENTOR und bestätige mit einem linken Mausklick auf "Speichern".



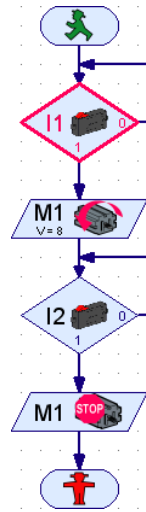
Um das Programm zu testen, drücke auf den links gezeigten Start-Knopf in der Werkzeugleiste. Zunächst testet ROBO Pro ob alle Programmelemente richtig verbunden sind. Sollte ein Element nicht richtig verbunden sein oder etwas anderes nicht in Ordnung sein, wird es rot markiert, und eine Fehlermeldung angezeigt, die beschreibt, was nicht in Ordnung ist. Wenn Du z.B. vergessen hast den Nein (N) Ausgang der Programmverzweigung anzuschließen, sieht das so aus:



Wenn du eine Fehlermeldung erhalten hast, musst du zunächst den gemeldeten Fehler korrigieren. Anderenfalls wird das Programm gestartet.

Hinweis: Eine ausführliche Erklärung zu dieser Betriebsart und zu der Betriebsart „Download-Betrieb“ findest du im Kapitel 3.7 auf Seite 24.

Das erste Verzweigungselement wird rot markiert. Es wird angezeigt, dass der Ablauf in diesem Programmelement auf ein Ereignis wartet, nämlich dass der Taster an I1, der das Garagentor öffnen soll, gedrückt wird. Solange der Taster nicht gedrückt ist, verzweigt das Programm zum Nein (N) Ausgang der Programmverzweigung und geht von dort wieder an den Anfang der Verzweigung. Drücke nun den Taster, der am Eingang I1 des Interface angeschlossen ist. Damit ist die Bedingung zum Weiterschalten erfüllt und der der Motor wird eingeschaltet. Der Ablauf wartet im nächsten Schritt darauf, dass der Endschalter am Eingang I2 gedrückt wird. Sobald du den Endschalter an I2 betätigst, verzweigt das Programm zum zweiten Motorausgang und schaltet den Motor wieder aus. Schließlich erreicht das Programm das Programmende. Es erscheint eine Meldung, dass das Programm beendet wurde.



Hat alles geklappt? Herzlichen Glückwunsch! Du hast damit Dein erstes Steuerungsprogramm erstellt und getestet. Falls es nicht funktioniert – nicht verzagen, einfach noch einmal alles genau nachprüfen, bestimmt hat sich irgendwo ein kleiner Fehler versteckt. Jeder Programmierer macht mal Fehler und aus Fehlern lernt man am meisten. Deshalb, nur Mut!!!

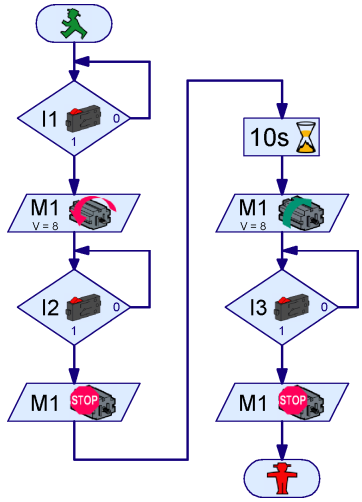
3.6 Weitere Programmelemente

Wenn du dein erstes Steuerprogramm an einem richtigen Garagentormodell ausprobiert hast, steht das Tor nun offen. Wie aber wird es wieder geschlossen? Natürlich könnten wir den Motor wieder durch Drücken eines Tasters starten! Wir wollen aber eine andere Lösung testen und dabei ein neues Programmelement kennen lernen. Hierfür speicherst du das Programm zunächst unter einem neuen Namen (den jetzigen Ablaufplan benötigen wir später noch einmal). Verwende dazu den Menüpunkt **Speichern unter …** im Menü **Datei** und gib dort einen noch nicht verwendeten Dateinamen ein.

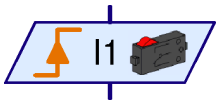
3.6.1 Wartezeit

Bevor wir den Ablaufplan erweitern können, musst du die Verbindung zwischen „Motor abschalten“ und „Programmende“ löschen und das Ende-Element nach unten verschieben. Die neuen Programmelemente kannst du nun zwischen diesen beiden Elementen einfügen. Das Garagentor soll nach einer Zeit von 10 Sekunden automatisch geschlossen werden. Hierfür kannst du das rechts abgebildete Programmelement ***Wartezeit*** verwenden. Die Wartezeit kannst du in weiten Grenzen beliebig einstellen, indem du wie üblich mit der rechten Maustaste auf das Element klickst. Gib die gewünschte Wartezeit von 10 Sekunden ein. Zum Schließen des Garagentores muss der Motor natürlich in die andere Richtung, also nach rechts, laufen. Abgeschaltet wird der Motor durch einen weiteren Endschalter an I3.

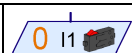
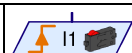
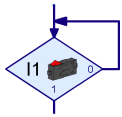
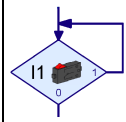
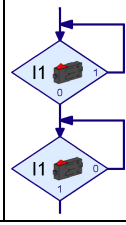
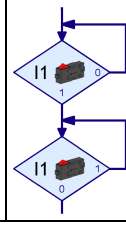
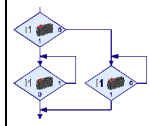
Dein fertiges Ablaufdiagramm sollte etwa so aussehen wie rechts dargestellt. Die neuen Programmelemente sind zur besseren Darstellung nach rechts verschoben. Sind im Ablaufplan keine Fehler mehr enthalten, kannst du die erweiterte Garagentorsteuerung wie gewohnt mit dem **Start** Knopf testen. Bei Betätigung des Tasters an I1 wird der Motor eingeschaltet, bei Betätigung von I2 wieder abgeschaltet. Damit ist das Garagentor geöffnet. Nun wird das Programmelement Wartezeit für 10 Sekunden, das ist unsere eingestellte Wartezeit, rot umrandet. Dann wird der Motor mit anderer Drehrichtung eingeschaltet bis der Taster an I3 betätigt wird. Versuche auch einmal die Wartezeit zu verändern.



3.6.2 Warten auf Eingang

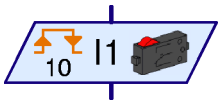


Neben dem Element Wartezeit gibt es noch zwei weitere Elemente, die auf etwas warten, bis sie die Programmausführung fortsetzen. Das links abgebildete **Warten auf Eingang Element** wartet bis ein Eingang des Interfaces einen bestimmten Zustand hat oder sich auf eine bestimmte Art ändert. Von diesem Element gibt es 5 Varianten:

Symbol					
Warten auf	Eingang=1 (geschlos-sen)	Eingang=0 (offen)	Wechsel 0-1 (offen nach geschlossen)	Wechsel 1-0 (geschlossen nach offen)	beliebiger Wechsel (1-0 oder 0-1)
Gleiche Funktion nur mit Verzwei-gung					

Man kann dafür auch eine Kombination aus Verzweigungselementen verwenden, aber mit dem Element **Warten auf Eingang** geht es einfacher und übersichtlicher.

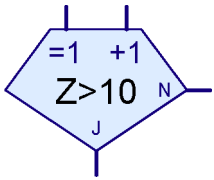
3.6.3 Impulszähler



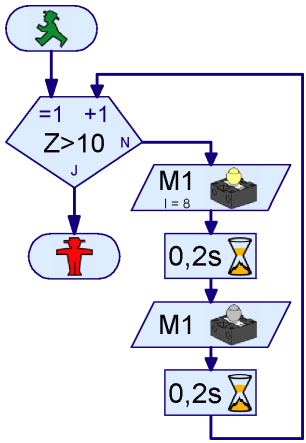
Viele Fischertechnik Robotermodelle verwenden auch Impulszahnräder. Diese Zahnräder betätigen einen Taster bei jeder Umdrehung 4 Mal. Mit solchen Impulszahnrädern kann man einen Motor statt einer bestimmten Zeit eine genau definierte Zahl von Umdrehungen einschalten. Dazu muss man die Zahl der Impulse an einem Eingang des Interface zählen. Zu diesem Zweck gibt es das links abgebildete ***Impulszählerelement***, das auf eine einstellbare Zahl von Impulsen wartet. Auch bei diesem Element kannst du einstellen, ob

beliebige Änderungen oder nur 0-1 oder nur 1-0 Wechsel als Impuls gewertet werden. Bei Impulsrädern wartet man in der Regel auf beliebige Änderungen, so dass man bei 4 Zähnen eine Auflösung von 8 Schritten pro Umdrehung erreicht.

3.6.4 Zählschleife



Mit dem Zählschleifenelement kannst du ganz einfach ein bestimmtes Programmstück mehrfach ausführen lassen. Das abgebildete Programm schaltet zum Beispiel eine Lampe an **M1** 10x ein und wieder aus. Das Zählschleifenelement hat einen eingebauten Zähler. Wenn die Zählschleife über den **=1** Eingang betreten wird, wird der Zähler auf 1 gesetzt. Wenn die Zählschleife dagegen über den **+1** Eingang betreten wird, wird zum Zähler 1 dazu gezählt. Je nach dem ob der Zähler größer einem von dir vorgegeben Wert ist oder nicht, verzweigt die Zählschleife zum Ja (**J**) oder zum Nein (**N**) Ausgang. Der Ja Ausgang wird also verwendet, wenn die Schleife so oft durchlaufen wurde, wie du es im Zählerwert vorgegeben hast. Wenn noch weitere Schleifendurchgänge notwendig sind, verzweigt die Zählschleife dagegen zum Nein Ausgang. Wie beim Verzweigungselement kannst du den Ja und Nein Ausgang über das Eigenschaftsfenster auch vertauschen.



3.7 Online- oder Download-Betrieb – Wo ist denn da der Unterschied?


Start

Bisher haben wir unsere Steuerungsprogramme im so genannten ***Online-Betrieb*** getestet. Du konntest dabei den Ablauf der Programme am Bildschirm mitverfolgen, weil das jeweils aktive Element am Bildschirm rot markiert worden ist. Den Online-Betrieb verwendest du um Programme zu verstehen oder Fehler in Programmen zu suchen.


Pause

Im Online-Betrieb kannst du das Programm auch anhalten und wieder fortsetzen, indem du den **Pause** Kopf drückst. Das ist sehr praktisch, wenn du bei deinem Modell etwas untersuchen möchtest, ohne das Programm vollständig zu stoppen. Auch wenn du versuchst den Ablauf eines Programms nachzuvollziehen, kann die Pause Funktion sehr hilfreich sein.


Schritt

Mit dem **Schritt** Knopf kannst du das Programm in Einzelschritten Element für Element ausführen. Jedes mal, wenn du den Schritt Knopf drückst, geht das Programm zum nächsten Programmelement. Wenn du ein ***Wartezeit*** Element oder ein ***Warten auf*** Element ausführst, kann es natürlich eine Weile dauern, bis das Programm beim nächsten Element ankommt.


Download

Wenn du ein ROBO Interface (kein Intelligent Interface) besitzt, kannst du statt dem Online-Betrieb auch den ***Download-Betrieb*** verwenden. Im Online-Betrieb werden die Programme von deinem Computer ausgeführt. Dein Computer sendet dabei Steuerbefehle wie „Motor einschalten“ an das Interface. Dazu ist es notwendig, dass das Interface mit dem Computer verbunden ist, solange das Programm läuft. Im Download-Betrieb wird dagegen das Programm vom Interface selbst ausgeführt. Dein Computer speichert das Programm im Interface. Sobald das geschehen ist, kann die Verbindung zwischen Computer und Interface getrennt werden. Nun kann das Interface das Steuerungsprogramm unabhängig vom Computer ausführen. Wichtig ist der Download-Betrieb z. B. bei

der Programmierung von mobilen Robotern, bei denen ein Verbindungskabel zwischen PC und Roboter sehr hinderlich wäre. Trotzdem sollten Steuerungsprogramme zuerst im Online-Betrieb getestet werden, da sich hier mögliche Fehler besser finden lassen. Das ausgetestete Programm kann dann per Download auf das ROBO Interface übertragen werden. Beim ROBO Interface kann das störende Kabel durch die Funkverbindung ***ROBO RF Data Link***, Art.-Nr. 93295 ersetzt werden. Damit ist das Modell auch im Onlinebetrieb uneingeschränkt mobil.

Der Online-Betrieb hat gegenüber dem Download-Betrieb aber auch Vorteile. Ein Computer hat im Vergleich zum Interface sehr viel mehr Arbeitsspeicher und kann viel schneller rechnen. Bei großen Programmen ist dies von Vorteil. Zudem können im Online-Betrieb mehrere Interfaces parallel angesteuert werden, auch ROBO Interfaces und Intelligent Interfaces gemischt.

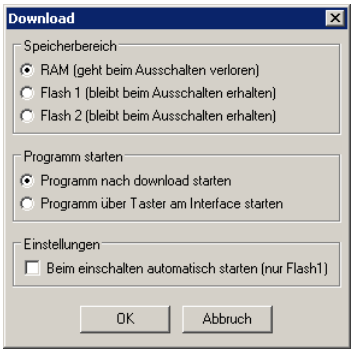
Die zwei Betriebsarten im Überblick

Betriebsart	Vorteil	Nachteil
Online	• Die Programmausführung kann am Bildschirm verfolgt werden • Die Ausführung auch großer Programme ist sehr schnell • Es können mehrere Interfaces parallel angesteuert werden • Das ältere Intelligent Interface wird unterstützt • Bedienfelder können verwendet werden • Das Programm kann angehalten und fortgesetzt werden.	• Computer und Interface müssen miteinander verbunden bleiben
Download	• Computer und Interface können nach dem Download voneinander getrennt werden	• Das ältere Intelligent Interface wird nicht unterstützt • Die Programmausführung kann nicht am Bildschirm verfolgt werden. • Das Programm kann nur bis zu 3 Erweiterungsmodule ansteuern

Den Download-Modus verwenden


Download

Solltest du also das neue ROBO Interface besitzen, kannst du deine Garagentorsteuerung mit dem **Download** Knopf in der Werkzeugleiste auf das Interface übertragen. Zunächst wird das rechts zu sehende Dialogfenster angezeigt. Das ROBO Interface verfügt über mehrere Programmspeicherplätze, einem ***RAM*** (Random Access Memory) Bereich und zwei ***Flash***-Bereiche. Ein Programm im RAM geht verloren, sobald du das Interface von der Stromversorgung trennst oder der Akkupack leer ist. Ein im Flash gespeichertes Programm bleibt dagegen auch ohne Strom jahrelang auf dem Interface gespeichert. Natürlich kannst du Programme im Flash trotzdem jederzeit überschreiben. Der Download in den RAM geht aber deutlich schneller und wird daher für Testzwecke empfohlen.



In den zwei Flashbereichen kannst du zwei verschiedene Programme, zum Beispiel zwei verschiedene Verhaltensweisen für einen mobilen Roboter, ablegen. Die zwei Programme kannst Du dann mit der **Prog**-Taste am Interface auswählen, starten und stoppen. Wenn die Option ***Programm nach download starten*** aktiviert ist, wird das Programm nach dem Download sofort gestartet. Während das Programm läuft, blinkt am ROBO Interface die grüne LED **Prog 1** (Programm in Flash 1 geladen) oder **Prog 2** (Programm in Flash 2 geladen) neben der **Prog**-Taste. Wurde ein Programm in den RAM geladen, blinken beide LEDs. Zum Stoppen des Programms drückst du die **Prog**-Taste. Dann leuchtet die LED dauernd. Zum Wechsel von Programm 1 zu Programm 2 hältst du die Prog-Taste so lange gedrückt, bis die LED für das gewünschte Programm 1 oder 2 aufleuchtet. Zum Starten des Programms drückst du die Taste erneut.

Bei mobilen Robotern ist die Option **Programme über Taster am Interface starten** sinnvoller. Falls du keinen RF Data Link besitzt, musst du nämlich noch das Kabel abstecken, bevor dein Programm den Roboter in Bewegung setzt. In diesem Fall musst du zuerst über die Prog-Taste am Interface das gewünschte Programmauswählen und durch erneutes Drücken das Programm starten.

Wenn die letzte Option ***Beim Einschalten automatisch starten*** aktiviert ist, wird das Programm 1 im Flash automatisch gestartet, sobald das Interface mit Strom versorgt wird. Damit kannst du z. B. dein Interface über ein Netzteil mit Zeitschaltuhr mit Strom versorgen und das Programm jeden Tag um die gleiche Zeit starten. Es muss dann weder das Interface dauernd eingeschaltet bleiben noch das Programm jedes Mal nach dem Einschalten über die Prog-Taste gestartet werden.

Hinweise:

Wenn ein Programm in den Flash geladen oder aus dem Flash ausgeführt wird, gehen in den RAM geladene Programme verloren, weil Flash Programme den RAM Speicher ebenfalls verwenden.

Eine ausführliche Beschreibung der Funktionen des ROBO Interfaces findest du auch in der Bedienungsanleitung zum Interface.

3.8 Tipps und Tricks

Verbindungslinien verändern

Wenn du Elemente verschiebst bemüht sich ROBO Pro die Verbindungslinien vernünftig anzupassen. Falls dir eine angepasste Linie einmal nicht gefällt, kannst du die Verbindungslinien leicht verändern, indem du mit der linken Maustaste auf die Linie klickst und die Linie mit gedrückter Maustaste verschiebst. Je nachdem wo sich die Maus auf der Linie befindet, wird ein Eckpunkt oder eine Kante der Linie verschoben. Das wird durch unterschiedliche Mauscursors angezeigt:

Wenn sich die Maus über einer senkrechten Verbindungslinie befindet, kannst du mit gedrückter linker Maustaste die ganze senkrechte Linie verschieben.

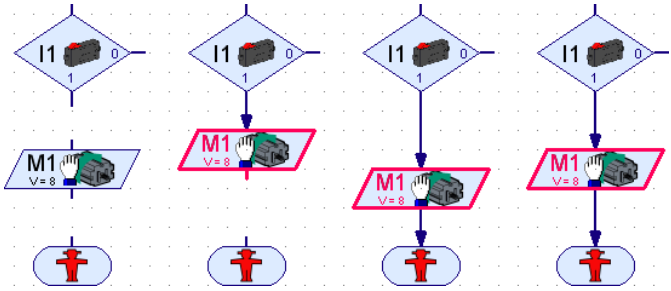
Wenn sich die Maus über einer waagrechten Verbindungslinie befindet, kannst du mit gedrückter linker Maustaste die ganze waagrechte Linie verschieben.

Wenn sich die Maus über einer schrägen Verbindungslinie befindet, wird ein neuer Punkt in die Verbindungslinie eingefügt, wenn du die linke Maustaste drückst. Du musst die linke Maustaste gedrückt halten, und die Maustaste erst loslassen, wenn sich die Maus da befindet, wo der neue Punkt seinen Platz haben soll.

Wenn sich die Maus in der Nähe eines Eckpunktes oder Endpunktes einer Verbindungslinie befindet, kannst du den Punkt mit gedrückter linker Maustaste verschieben. Einen verbundenen Linienendpunkt kannst du nur auf einen anderen passenden Anschluss eines Programmelements ziehen. In diesem Fall wird der Endpunkt der Verbindungslinie mit diesem Anschlusspunkt verbunden. Anderenfalls wird der Punkt nicht verschoben.

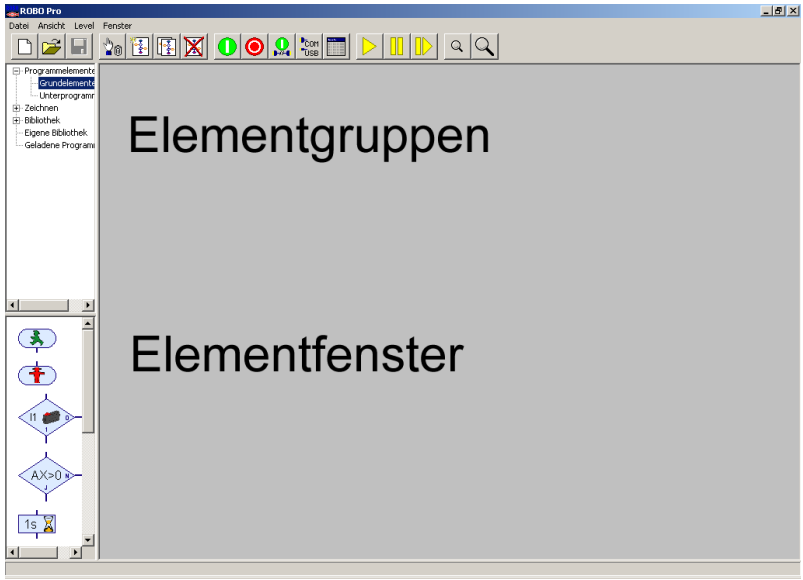
Verbindungslinien einmal anders

Verbindungslinien lassen sich auch durch Verschieben der Programmelemente erzeugen. Wenn du ein Programmelement so verschiebst, dass sein Eingang ein oder zwei Rasterpunkte unter dem Ausgang eines anderen liegt, wird zwischen beiden Elementen eine Verbindungslinie erzeugt. Das gilt auch bei einem Ausgang, der über einen Eingang geschoben wird. Danach kannst du das Programmelement in seine Endposition schieben oder weitere Verbindungen für die verbleibenden Ein und Ausgänge zeichnen:



4 Level 2: Arbeiten mit Unterprogrammen

Nachdem du erfolgreich dein erstes Steuerprogramm erstellt und getestet hast, bist du bereit für ROBO Pro Level 2. Wähle nun im Menü **Level** den Eintrag **Level 2: Unterprogramme** aus. Sicher bemerkst du gleich den Unterschied: Das Elementfenster ist verschwunden und stattdessen hast du nun am linken Rand zwei Fenster übereinander:



Aber keine Angst! Das Elementfenster ist noch da, nur ist es jetzt leer. Im Level 2 gibt es mehr Programmelemente, so dass es zu unübersichtlich wäre alles in ein Fenster zu packen. Daher sind die Elemente ab Level 2 zu Elementgruppen zusammengefasst. Die Elemente sind in Gruppen so ähnlich organisiert wie Dateien in Ordnern auf deiner Computerfestplatte. Wenn du im oberen Fenster am linken Rand eine Gruppe auswählst, erscheinen alle Elemente dieser Gruppe im unteren Fenster. Die Elemente aus dem Level 1 findest du in der Gruppe **Programmelemente** / **Grundelemente**. Da das Elementfenster nun nur noch halb so groß ist, musst du den Rollbalken rechts am Elementfenster verwenden um die unteren Elemente anzuzeigen.

So, nun aber zum eigentlichen Thema: den Unterprogrammen! Zwar sind unsere bisher entworfenen Ablaufpläne noch nicht so umfangreich, dass wir bereits die Übersicht verlieren, aber sicherlich kannst du dir vorstellen, dass dies bei größeren Projekten mit umfangreicheren Ablaufplänen sehr leicht der Fall sein kann. Plötzlich ist das Arbeitsblatt voller Bausteine, überall sind Verbindungslinien und auf dem Bildschirm muss man mit den Scroll-Balken ständig hin- und herfahren. "Wo war jetzt dieser oder jener Ausgang?" Kurz – es droht ein kleines Chaos! Was tun? Gibt es nicht eine Möglichkeit, hier wieder etwas mehr Ordnung zu schaffen? Doch – und sie nennt sich ***Unterprogramm***!

4.1 Dein erstes Unterprogramm

Ein Unterprogramm ist den Programmen die du bisher kennen gelernt hast sehr ähnlich. Damit du das genauer untersuchen kannst, musst du zunächst ein neues Programm und in dem Programm ein neues leeres Unterprogramm erzeugen. Drücke dazu auf Programm **Neu** und dann den **UP Neu** Knopf in der Werkzeugleiste. Ein Fenster erscheint, in dem du den Namen und eine Beschreibung für das Unterprogramm eingeben kannst.

Der Name sollte nicht zu lang sein (ca. 8-10 Buchstaben), da sonst das Unterprogrammsymbol sehr groß wird. Du kannst alle Eingaben, die du hier machst, natürlich später jederzeit ändern.

Sobald du das Fenster **Neues Unterprogramm** mit **OK** schließt, erscheint in der Unterprogrammleiste das neue Unterprogramm:

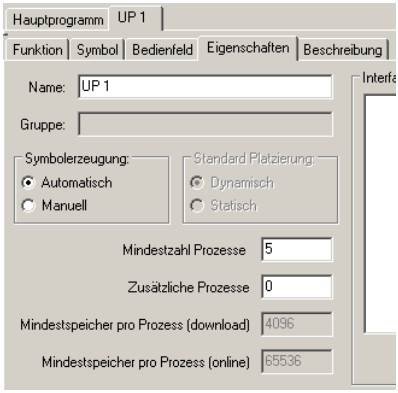
Du kannst jederzeit zwischen Hauptprogramm und Unterprogramm wechseln, indem Du in der Unterprogrammleiste auf den Programmnamen klickst. Da beide Programme noch leer sind, sieht man allerdings noch keinen Unterschied.

Wir wollen nun die Garagentorsteuerung aus dem vorigen Kapitel (siehe Abschnitt 3.6 *Weitere Programmelemente* auf Seite 22) in Unterprogramme gliedern. Das Programm besteht aus vier Funktionseinheiten:

- Warten bis Taster I1 gedrückt
- Tor öffnen
- 10 Sekunden warten
- Tor schließen

Das Öffnen und Schließen wollen wir nun auf zwei Unterprogramme verteilen. Die beiden Unterprogramme kannst du dann aus dem Hauptprogramm mit nur einem Symbol aufrufen. Das Warten auf Taster I1 und die Wartezeit von 10 Sekunden bleiben im Hauptprogramm, da beide ohnehin nur aus einem Element bestehen. Du hast gerade ein neues Programm mit einem Unterprogramm namens **Unterprogramm 1** angelegt. **Öffnen** und **Schließen** wären aber bessere Namen für die zwei Unterprogramme. Du kannst das bereits angelegte Unterprogramm umbenennen, indem Du zunächst das Unterprogramm 1 über die Unterprogrammleiste auswählst, sofern es nicht schon ausgewählt ist.

Schalte dann über die Funktionsleiste auf das Eigenschaftsfenster für das Unterprogramm um, indem du auf **Eigenschaften** klickst. Hier kannst du den Namen von **UP 1** in **Auf** ändern. Die meisten anderen Felder sind nur in der Fortgeschrittenenstufe oder gar in der Expertenstufe veränderbar. Der Punkt **Symbolerzeugung** wird etwas später erklärt.

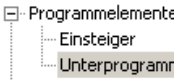


Wenn du in der Funktionsleiste auf Beschreibung klickst, kannst du die zuvor eingegebene Beschreibung ändern, obwohl „Mein erstes Unterprogramm“ ja nach wie vor ganz zutreffend ist.

Klicke in der Funktionsleiste nun auf **Funktion**, damit du die Funktion des Unterprogramms programmieren kannst. Nun siehst du wieder das Programmfenster, in dem du schon im vorherigen Kapitel die Programmelemente für dein erstes ROBO Pro Programm eingefügt hast. Achte darauf, dass in der Unterprogrammleiste das Unterprogramm **Auf** ausgewählt ist:

Bist du bereit dein erstes Unterprogramm zu programmieren? Na dann los! Aber womit fängt ein Unterprogramm eigentlich an? Gute Frage! Ein Hauptprogramm hast du immer mit dem Startelement begonnen. Ein Unterprogramm beginnt mit einem ähnlichen Element, dem Unterprogrammeingang. Das Element heißt so, weil durch dieses Element die Programmführung vom Hauptprogramm in das Unterprogramm hineingeht. Du kannst hier kein Startelement verwenden, weil ja kein neuer Prozess gestartet werden soll.

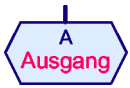
	Startelement	Startet einen neuen, eigenständigen Prozess
	Unterprogramm-eingang	Hier wird die Programmführung vom Hauptprogramm an das Unterprogramm übergeben



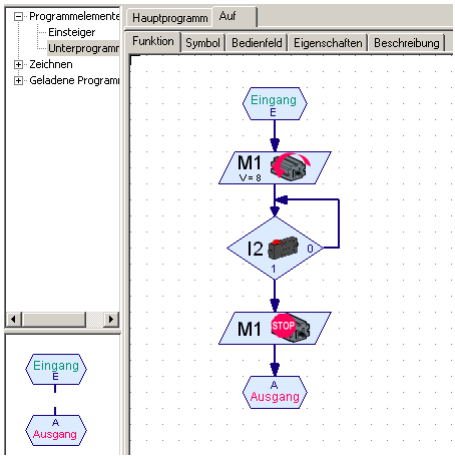
Das Unterprogrammeingang findest du im Elementgruppenfenster unter **Unterprogramm I/O**. Platziere nun das Unterprogrammeingang oben im Programmfenster für das Unterprogramm **Öffnen**. Einem Unterprogrammeinganselement kannst du auch einen anderen Namen als **Eingang** geben, aber das ist nur nötig wenn du später einmal ein Unterprogramm mit mehreren Eingängen schreibst.

Der weitere Programmablauf im Unterprogramm ist identisch mit dem Öffnen-Teil des bisherigen Hauptprogramms: Du schaltest den Motor M1 mit Drehrichtung links ein, wartest bis der Taster am Eingang I2 geschlossen ist und schaltest dann den Motor wieder aus.

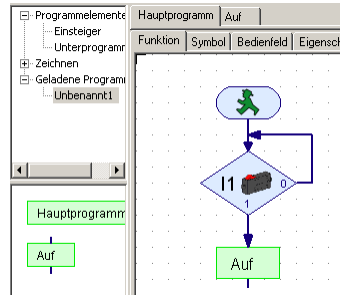
Um das Unterprogramm abzuschließen verwendest du ein Unterprogrammausgang. Der Unterschied zwischen dem Unterprogrammausgang und dem Stoppelement ist der gleiche wie zwischen Unterprogrammeingang und Prozessstart.

	Stoppelement	Beendet die Programmausführung eines unabhängigen Prozesses
	Unterprogrammausgang	Hier wird die Programmführung vom Unterprogramm an das Hauptprogramm zurückgegeben.

Dein fertiges Unterprogramm sollte nun etwa so aussehen:



Achte darauf, dass du das Unterprogramm wirklich unter **Auf**, und nicht unter **Hauptprogramm** eingetragen hast. Schalte nun in der Unterprogrammleiste wieder von **Auf** auf **Hauptprogramm**. Du siehst nun das nach wie vor leere Programmfenster des Hauptprogramms. Füge in das Hauptprogramm wie gewohnt ein Startelement (keinen Unterprogrammeingang!) ein. Auch die Abfrage des Tasters an **I1**, der das Garagentor öffnen soll, führst du wie gewohnt im Hauptprogramm aus.



Dein neues Unterprogramm kannst du nun wie ein gewöhnliches Programmelement in dein Hauptprogramm (oder ein anderes Unterprogramm) einfügen. Du findest es im Elementgruppenfenster unter **Geladene Programme** und dem Dateinamen deines Programms. Wenn du die Datei noch nicht gespeichert hast ist der Name **unbenannt1**. Wenn du noch mehr Programmdateien geladen hast, kannst du im Auswahlfenster auch die Unterprogramme, die zu anderen Dateien gehören, auswählen. Auf diese Weise kannst du sehr einfach Unterprogramme aus einer anderen Datei verwenden.

In der Elementgruppe **Geladen Programme / unbenannt1** findest du zwei grüne Unterprogrammsymbole. Das erste mit dem Namen **Hauptprogramm** ist das Symbol für das Hauptprogramm. Das wird eher selten als Unterprogramm verwendet, aber möglich ist das schon, zum Beispiel wenn du einen ganzen Maschinenpark steuerst, und du die Steuerungen für die einzelnen Maschinen zuvor einzeln als Hauptprogramme entwickelt hast. Das zweite Symbol mit dem Namen **Auf** ist das Symbol deines neuen Unterprogramms. **Auf** ist der Name, den du unter Eigenschaften

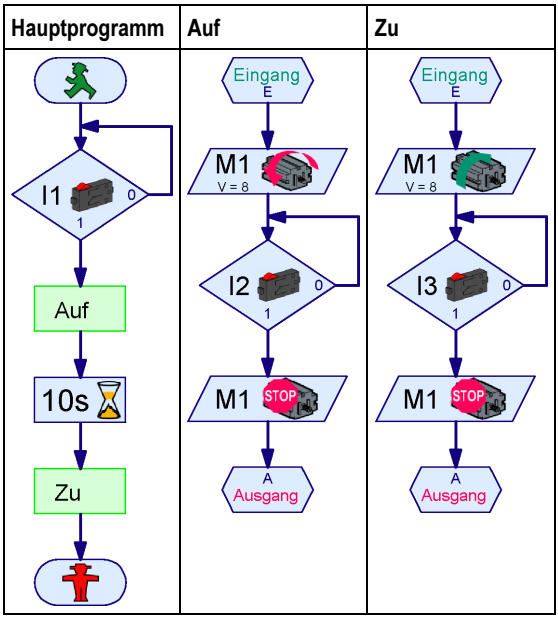
eingetragen hast. Füge nun das Unterprogrammsymbol, so wie du es von gewöhnlichen Programmelementen gewohnt bist, in dein Hauptprogramm ein. So einfach ist das!


UP Neu

Du kannst dein Hauptprogramm schon einmal mit einem Stoppelement abschließen, und es ausprobieren. Das Tor lässt sich durch drücken des Tasters I1 öffnen, aber den Schließen-Teil haben wir noch nicht programmiert. Dazu legst du ein weiteres Unterprogramm an. Drücke den **UP Neu** Knopf in der Werkzeugleiste und gib im Fenster **Neues Unterprogramm** als Name **Zu** ein. Eine Beschreibung brauchst du nicht eingeben, aber schaden tut es nichts, damit du später noch weißt wozu das Unterprogramm gedacht ist.

Gib nun im Programmfenster für das Unterprogramm **Zu** das Programm zum Schließen des Garagentors ein. Du beginnst wieder mit einem Unterprogrammeingang. Der Motor **M1** soll sich zunächst rechts drehen. Sobald der Endtaster an **I3** geschlossen wird, soll der Motor **M1** stoppen. Abgeschlossen wird das Unterprogramm wieder mit einem Unterprogrammausgang.

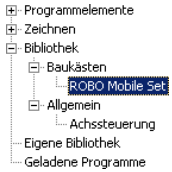
Wechsle nun wieder über die Unterprogrammleiste zum Hauptprogramm. Wenn du vorhin dein Hauptprogramm mit einem Stoppelement angeschlossen hast um es auszuprobieren, musst du das Stoppelement jetzt wieder löschen. Nachdem das Garagentor geöffnet ist, soll es 10 Sekunden offen bleiben, bevor es wieder geschlossen wird. Nach einer Wartezeit von 10 Sekunden fügst du aus der Elementgruppe **Geladene Programme / unbenannt1** das Symbol **Zu** für das Unterprogramm ein. Das Hauptprogramm und die beiden Unterprogramme sollten in etwa so aussehen:



Das Programm startet am Startelement im **Hauptprogramm**. Dann wartet das Programm bis der Taster **I1** gedrückt ist. Dafür könntest du übrigens auch das ***Warten auf Eingang*** Element verwenden (siehe Abschnitt 7.1.8 *Warten auf Eingang* auf Seite 61). Nachdem der Taster I1 gedrückt ist trifft das Hauptprogramm auf den Aufruf des Unterprogramms **Auf**. Die Programmausführung wechselt dadurch zum Unterprogrammeingang des Unterprogramms **Auf**. Das Unterprogramm **Auf** öffnet das Garagentor und kommt dann zu seinem Unterprogrammausgang. An dieser Stelle verzweigt das Programm wieder zum Hauptprogramm. Nach dem Ende des Unterprogramms **Auf** wird im Hauptprogramm 10 Sekunden gewartet. Anschließend wechselt die

Programmausführung zum Unterprogramm **Zu**, das das Garagentor wieder schließt. Nach der Rückkehr aus dem Unterprogramm **Zu** trifft das Hauptprogramm auf ein Stoppelement, wodurch das Programm beendet ist.

4.2 Die Unterprogrammbibliothek

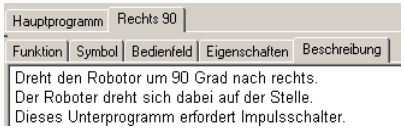


Du kannst recht einfach Unterprogramme von einer Datei in eine andere kopieren, indem du beide Dateien lädst und dann ein Unterprogramm aus einer Datei über die Elementgruppe **Geladene Programme** in die andere Datei einfügst. Für häufig verwendete Unterprogramme geht es aber noch einfacher, und zwar mit der **Bibliothek**. ROBO Pro enthält eine Bibliothek von fertigen Unterprogrammen, die du einfach wieder verwenden kannst. Außerdem kannst du eine eigene Bibliothek anlegen, in der du deine häufig verwendeten Unterprogramme ablegen kannst.

4.2.1 Verwenden der Bibliothek

Die **Bibliothek** ist zunächst in zwei Hauptgruppen unterteilt. Unter der Gruppe **Baukästen** findest du Unterprogramme, die du für Modelle aus bestimmten Baukästen verwenden kannst. Unter der Gruppe **Allgemein** findest du Unterprogramme, die du für alle möglichen Modelle verwenden kannst. Die meisten dieser Unterprogramme aus der Gruppe **Allgemein** erfordern aber Techniken aus dem Level 3, die erst im nächsten Kapitel erklärt werden.

Für jeden Computing Baukasten, wie zum Beispiel das ROBO Mobile Set, gibt es in der Gruppe **Baukästen** eine eigene Untergruppe. Diese ist manchmal weiter nach den Modellen unterteilt, die du in der Bauanleitung zu dem Baukasten findest. Wenn du den Baukasten oder eines der Modelle auswählst, werden im Elementfenster die fertigen Unterprogramme für dieses Modell angezeigt.



Wenn du mit der Maus auf eines der Unterprogrammsymbole zeigst, wird eine kurze Beschreibung angezeigt. Wenn du ein Unterprogramm in dein Programm einfügst, kannst du eine genaue Beschreibung anzeigen, indem du das Unterprogramm in der Unterprogrammleiste auswählst und dann in der Funktionsleiste auf **Beschreibung** klickst:

Achtung: Wenn du ein Unterprogramm aus der Bibliothek einfügst, werden zum Teil weitere Unterprogramme eingefügt, die von diesem Unterprogramm verwendet werden. Du kannst alle Unterprogramme wieder entfernen, indem du aus dem Menü **Bearbeiten** die Funktion **Undo** auswählst.

4.2.2 Verwenden der eigenen Bibliothek

Nachdem du dich eine Weile mit ROBO Pro beschäftigt hast, wirst du sicherlich eigene Unterprogramme haben, die du häufiger verwendest. Damit du nicht jedes Mal die entsprechende Datei suchen und laden musst, kannst du auch eine eigene Unterprogrammbibliothek anlegen, die genauso funktioniert wie die vordefinierte Bibliothek. Die eigene Bibliothek besteht aus einer oder mehreren ROBO Pro Dateien, die alle in einem Ordner gespeichert sind. Für jede Datei in diesem Ordner wird in der Gruppenauswahl eine eigene Gruppe angezeigt.

In welchem Ordner du deine eigne Bibliothek speichern möchtest, kannst du im Menü **Datei** unter **Eigenes Bibliotheksverzeichnis** angeben. Das Standardverzeichnis für die eigene Bibliothek ist C:\Programme\ROBOPro\Eigene Bibliothek. Wenn du ein eigenes Benutzerverzeichnis auf deinem Computer hast, empfiehlt es sich dort einen eigenen Ordner dafür anzulegen und diesen zu verwenden.

Tipp: Am Anfang kannst du den Ordner, in dem du auch deine ROBO Pro Programme speicherst, bei **Eigenes Bibliotheksverzeichnis** angeben. Dann hast du schnellen Zugriff auf alle Unterprogramme in allen Dateien in deinem Arbeitsordner.

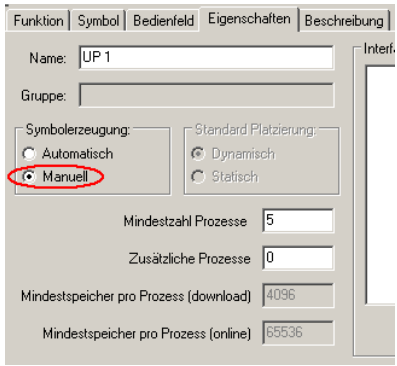
Organisieren deiner eigenen Bibliothek

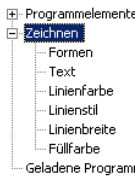
Es gibt in ROBO Pro keine speziellen Funktionen um eine Bibliothek zu ändern. Das geht aber trotzdem ganz einfach. Wenn du zu einer Bibliotheksgruppe Unterprogramme hinzufügen oder daraus entfernen möchtest, musst du zunächst einmal die entsprechende Datei laden. Du findest diese Datei in dem Verzeichnis, das du unter **Eigenes Bibliotheksverzeichnis** eingestellt hast. Nun kannst du zum Beispiel eine zweite Datei laden und aus dieser ein Unterprogramm aus der Gruppe **Geladene Programme** in das Hauptprogramm der Bibliothek ziehen. Bei einer Bibliothek ist das Hauptprogramm kein richtiges Programm, sondern lediglich eine Sammlung aller Unterprogramme der Bibliothek. Das Hauptprogramm selbst wird bei Bibliotheken im Elementfenster nicht angezeigt. Du kannst natürlich auch Unterprogramme aus einer Bibliothek löschen oder Unterprogramme verändern.

Wenn du eine Bibliotheksdatei verändert und gespeichert hast, musst du im Menü **Datei** den Menüpunkt **Eigene Bibliothek aktualisieren** auswählen. Dadurch wird die Dateiliste im Gruppenfenster aktualisiert.

4.3 Bearbeiten von Unterprogrammsymbolen

Wie du im vorigen Abschnitt gesehen hast, erzeugt ROBO Pro für deine Unterprogramme automatisch grüne Unterprogrammsymbole. Du kannst aber auch eigene Symbole zeichnen, die besser verdeutlichen, was deine Unterprogramme machen. Dazu musst du im Eigenschaftsfenster des Unterprogramms von automatischer auf manuelle Symbolerzeugung umschalten. Anschließend kannst du in der Funktionsleiste von **Eigenschaften** auf **Symbol** umschalten und dort das Unterprogrammsymbol bearbeiten. Zeichenfunktionen findest du im Elementgruppenfenster unter **Zeichnen**.

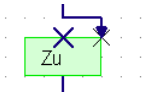




Unter **Zeichen / Formen** findest alle üblichen grafischen Grundelemente wie Rechteck, Kreis, Ellipse, Polygone und ähnliches. Unter **Zeichen / Text** findest du Textobjekte in verschiedenen Schriftgrößen. In den anderen Gruppen unter Zeichnen findest du Funktionen zum Ändern der Farbe und ähnlicher Eigenschaften ausgewählter Elemente. Die genaue Anwendung der Zeichenfunktionen ist in Kapitel 9 *Zeichenfunktionen* auf Seite 87 erklärt. Beachte auch die Funktionen im

Hauptmenü unter **Zeichnen**.

Du kannst auch die Anschlüsse des Unterprogramms verschieben, aber du kannst die Anschlüsse nicht löschen oder neue hinzufügen. Im Unterprogrammsymbol gibt es immer für jeden Unterprogrammeingang oder Unterprogrammausgang der Unterprogrammfunktion einen Anschluss. Die Anschlusselemente werden auch dann automatisch erzeugt, wenn du auf manuelle Symbolerzeugung umgeschaltet hast.



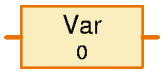
Sobald du das Symbolbearbeitungsfenster verlässt, werden alle Aufrufe des Unterprogramms im Hauptprogramm oder in anderen Unterprogrammen entsprechend angepasst. Bitte beachte, dass wenn du Anschlüsse eines Unterprogramms verschoben hast, bei den Aufrufen des Unterprogramms ein kleines Durcheinander entstehen kann, wenn die Anschlüsse schon angeschlossen waren. Die Endpunkte der Verbindungslinien enden dann unter Umständen nicht mehr auf dem richtigen Anschluss, was durch ein Kreuz am Linienendpunkt und am Anschluss angezeigt wird (siehe Bild). In der Regel reicht es aus, wenn du einmal mit der linken Maustaste irgendwo auf die Verbindungslinie klickst. Die Linie wird dann automatisch neu verlegt. Bei Unterprogrammen mit vielen Verbindungen kann es aber sein, dass du die Linie noch bearbeiten musst.

5 Level 3: Variablen, Bedienfelder & Co

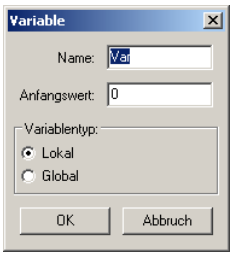
Denk daran ROBO Pro im Menü **Level** auf **Level 3** (oder höher) umzustellen!

Stell dir einmal vor, dass du in einem Museum in einem bisher unerforschten Seitengang eine faszinierende Maschine entdeckst, die du unbedingt aus fischertechnik nachbauen möchtest. Beim ergründen der Maschine vergisst du aber die Zeit und merkst nicht dass alle anderen Besucher das Museum verlassen. Erst als das Museum bereits geschlossen ist, hast du die Maschine genau genug studiert um sie nachbauen zu können. Aber leider musst du erst eine unangenehme Nacht alleine im Museum verbringen, bevor du loslegen kannst. Damit das nicht wieder vorkommt, bietest du dem Museumsdirektor an einen Besucherzähler zu programmieren, der alle Besucher beim hineingehen und hinausgehen zählt und ein rote Warnlampe einschaltet solange noch Besucher im Museum sind. Nur wie machst du das? Wie kannst du mit ROBO Pro etwas zählen? Die Antwort lautet: mit **Variablen**.

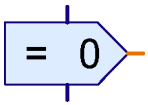
5.1 Variablen und Befehle



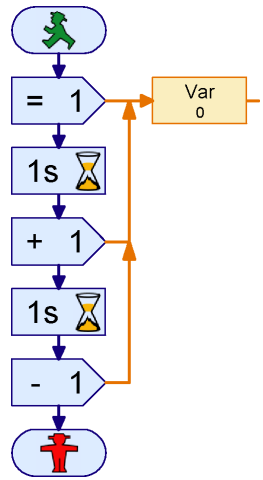
Eine Variable ist ein Element, das sich eine Zahl merken kann. Im Eigenschaftsfenster der Variablen gibst du einen **Namen** ein, der einen Hinweis darauf geben soll, was für eine Zahl in der Variablen gespeichert ist. Unter **Anfangswert** kannst du angeben, welche Zahl am Anfang des Programms in der Variablen gespeichert werden soll. Die Einstellung **Variablentyp** wird im Abschnitt 7.3.2 *Lokale Variable* auf Seite 65) erklärt.



Die gespeicherte Zahl kannst du ändern, indem du Befehle an die Variable schickst. Eine Variable versteht 3 verschiedene Befehle: **=**, **+** und **–**. Der **=** Befehl ersetzt die gespeicherte Zahl durch eine neue Zahl. Der **+** und **–** Befehl zählen zu der gespeicherten Zahl etwas hinzu oder ziehen etwas davon ab.



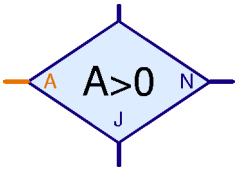
Die Befehle schickst du mit einem ***Befehlselement*** an die Variable. Das Befehlselement hat wie die meisten anderen Programmelemente oben einen blauen Programmeingang und unten einen blauen Programmausgang. Rechts hat es aber etwas ganz neues, einen ***orangen*** Anschluss. Das ist ein Befehlsausgang. Immer wenn das Befehlselement ausgeführt wird, schickt es über diesen Ausgang an alle angeschlossenen Elemente einen Befehl. Die Variable hat auf der linken Seite einen passenden Befehlseingang. Wenn du den Befehlsausgang mit dem Befehlseingang verbindest, zeichnet ROBO Pro statt der üblichen blauen Verbindungen eine orange Linie. Über die orangen Linien können Programmelemente Befehle oder Nachrichten senden und so Informationen austauschen.



Das Programm rechts schickt der Variablen **Var** zunächst einen **= 1** Befehl. Ein Befehl besteht in der Regel aus einem eigentlichen Befehl wie **=** und einem Wert wie **1**. Der =1 Befehl setzt die Variable auf 1. Nach einer Sekunde schickt das Programm der

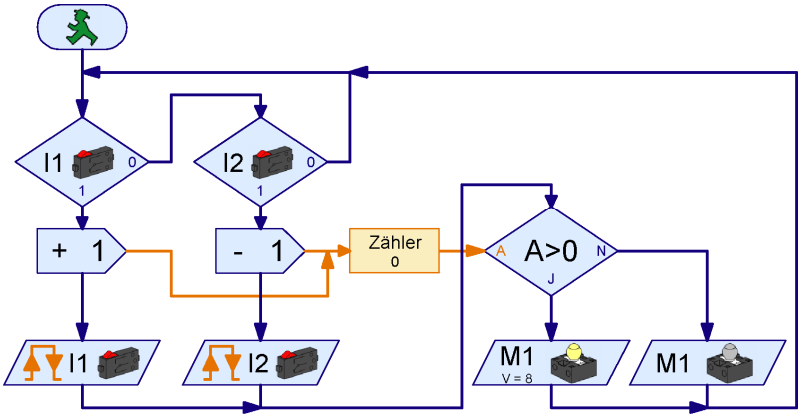
Variablen einen **+1** Befehl. Die Variable zählt daraufhin zu ihrem bisherigen Wert 1 hinzu und hat dann der Wert 2. Nach einer weiteren Sekunde schickt das Programm einen **-1** Befehl. Daraufhin hat die Variable wieder den Wert 1.

Versuche einmal dieses einfache Programm in ROBO Pro zu zeichnen. Die Befehlselemente findest du in der Gruppe **Befehle**, die Variable in der Gruppe **Variable, Timer, ….** Wenn du das Programm im Online-Modus ausführst, siehst du wie sich der Wert der Variablen ändert.



Schön und gut, wirst du nun vielleicht sagen: Ich kann mir den Wert der Variablen ansehen, aber was mache ich denn damit? Ganz einfach: Die Variable hat rechts einen orangen Anschluss, über den sie Nachrichten mit ihrem aktuellen Wert an alle angeschlossenen Elemente schickt. Es gibt einige Elemente in ROBO Pro, die links einen orangen Eingang haben, den du mit dem Ausgang der Variablen verbinden kannst. So findest du zum Beispiel in der Gruppe **Verzweigung, Warten, ...** ein Ja / Nein Verzweigungselement, das nicht direkt einen Eingang abfragt, sondern einen beliebigen Wert abfragen kann, unter anderem den Wert einer Variablen.

Damit lässt sich der Besucherzähler für das Museum wie folgt programmieren:

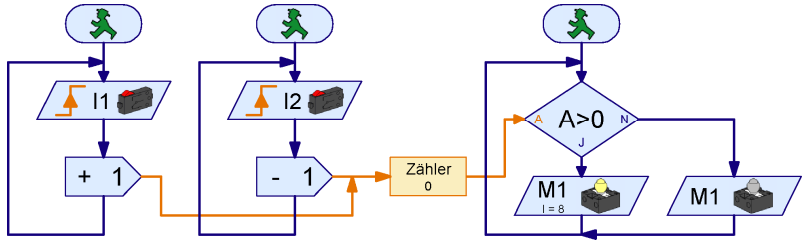


Das Drehkreuz am Eingang betätigt den Taster an **I1**, das Drehkreuz am Ausgang den Taster an **I2**. Sobald **I1** gedrückt ist, schickt das Programm der Variablen **Zähler** einen **+ 1** Befehl. Anschließend wartet das Programm bis der Taster an **I1** wieder losgelassen wird. Mit dem Taster für den Ausgang an **I2** verhält es sich genauso, nur dass hier der Variablen **Zähler** ein **- 1** Befehl geschickt wird. Jedes mal wenn sich der Zähler verändert hat, wird der Zählerstand kontrolliert. Wenn die Variable **Zähler** einen Wert **> 0** hat wird die rote Warnlampe an **M1** eingeschaltet, sonst ausgeschaltet.

Zeichne nun das obige Programm nach und probiere es aus. Sobald du den Taster an I1 drückst und wieder los lässt, leuchtet die Warnlampe an M1 auf. Wenn du den Taster an I2 betätigst, geht sie wieder aus. Wenn du I1 mehrfach betätigst musst du I2 genauso oft betätigen, damit die Warnlampe wieder ausgeht. Versuche auch einmal was passiert wenn erst 5 Besucher kommen, dann 2 gehen, dann noch mal 3 kommen. Wie oft musst du jetzt den Taster an I2 betätigen, damit die Warnlampe wieder aus geht?

5.2 Variablen und mehrere Prozesse

Vielleicht ist dir beim Testen des Besucherzählers aufgefallen, dass es Probleme macht, wenn die Taster an I1 und I2 gleichzeitig gedrückt werden. Solange einer der Taster gedrückt ist, kann das Programm nicht mehr auf den anderen Taster reagieren. Nachdem aber die Besucher am Eingang und am Ausgang durchaus gleichzeitig durch das jeweilige Drehkreuz gehen können, führt das zu Zählfehlern. Du kannst diesen Fehler beheben, indem du mehrere parallele Prozesse verwendest. Bisher hatten alle Programme immer nur ein Startelement. Du kannst aber durchaus mehrere Startelemente verwenden. Alle Abläufe mit einem eigenen Startelement werden dann nebeneinander abgearbeitet. Der Fachmann spricht daher von ***nebenläufigen Prozessen***. Mit dieser Technik kannst du das Besucherzählerprogramm wie folgt abändern:



Für I1 und I2 werden nun unabhängige Prozesse verwendet. Wenn der Taster an I1 gedrückt ist, bleibt der Prozess für I2 davon unabhängig und kann weiter den Taster an I2 überwachen. Zum Abfragen der Zählwerte und zum Ein- und Ausschalten der Warnlampe wird ebenfalls ein eigener Prozess verwendet.

Wie du siehst macht es keine Probleme von mehreren Prozessen aus auf eine Variable zu zugreifen. Du kannst einer Variablen von mehreren Prozessen aus Befehle schicken und du kannst den Wert einer Variablen in mehreren Prozessen verwenden. Daher eignen sich Variablen auch sehr gut um Informationen zwischen Prozessen auszutauschen

Der Museumsdirektor ist von deinem genialen Besucherzähler so begeistert, dass er dich gleich um die Lösung eines anderen Problems bittet: Das Museum hat eine neue Ausstellung eingerichtet. Da aber alle Besucher die neue Ausstellung sehen wollen, herrscht dort so ein Gedränge, dass nun gar keiner mehr etwas sehen kann. Der Direktor möchte daher die Zahl der Besucher in der Ausstellung auf 10 begrenzen. Am Eingang und am Ausgang zur Ausstellung hat der Direktor jeweils ein Drehkreuz aufgestellt. Das Drehkreuz am Eingang lässt sich elektronisch verriegeln. Nun braucht er nur noch einen fähigen Programmentwickler, und zwar dich!

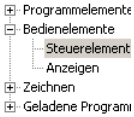
Versuche das beschriebene Programm mit ROBO Pro zu entwickeln. Es funktioniert im Wesentlichen wie der Besucherzähler. Die elektronische Verriegelung des Eingangs simulierst du durch eine rote Lampe an M1, die eingeschaltet sein soll, wenn 10 Besucher in der Ausstellung sind.

5.3 Bedienfelder

Nachdem du das Problem mit der Ausstellung gelöst hast, hat der Museumsdirektor schon wieder eine neue Aufgabe: Er möchte wissen wie viele Besucher an einem Tag sein Museum besucht haben. Ein Programm das zählen kann ist ja nun kein Problem mehr für dich, aber wie kannst du einen Wert anzeigen? Natürlich könntest du das Programm im Online-Modus ausführen und dem Museumsdirektor zeigen, bei welcher Variablen er den Wert nachsehen kann. Aber für einen Computer-Laien wie den Museumsdirektor ist das doch recht kompliziert. Etwas Einfacheres muss her!

Für solche Fälle gibt es in ROBO Pro Bedienfelder. Ein Bedienfeld ist eine eigene Seite, auf der du Anzeigen und Bedienknöpfe zeichnen kannst. Lade dein Besucherzählerprogramm und schalte in der Funktionsleiste auf **Bedienfeld** um.

Das Bedienfeld ist zunächst einmal eine leere graue Fläche. Auf diese Fläche platzierst du Anzeigen und Steuerelemente, die du im Elementgruppenfenster unter **Bedienelemente** findest. Unter den Steuerelementen findest du Druckknöpfe, Schieberegler und ähnliches. Unter Anzeigen findest du Textanzeigen, Anzeigelampen und Anzeigen mit Drehzeiger.

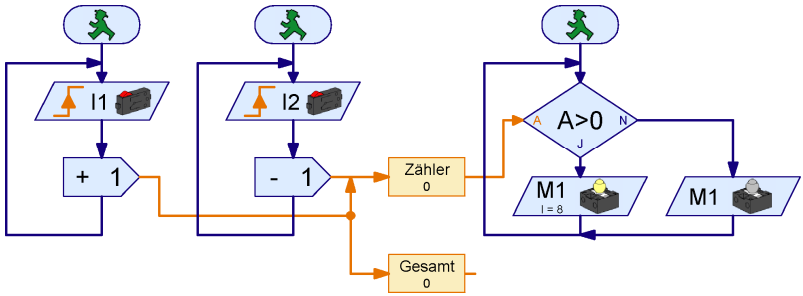


Achtung: Ein Bedienfeld ist Teil eines Unterprogramms. Wenn du Unterprogramme hast, achte darauf, dass du das Bedienfeld unter **Hauptprogramm** und nicht unter einem Unterprogramm anlegst! Später als Profi kannst du mehrere Bedienfelder anlegen.

Falls du ein Bedienfeld gezeichnet hast und das Bedienfeld später plötzlich verschwunden ist, hast du vermutlich in der Unterprogrammleiste ein Unterprogramm ausgewählt. Schalte wieder auf **Hauptprogramm** und dein Bedienfeld ist sicher wieder da.

Für den Besucherzähler nimmst du eine **Textanzeige** (die Farbe ist egal) aus dem Elementfenster **Bedienelemente / Anzeigen** und platzierst sie im Bedienfeld. In dieser Anzeige soll nun die Anzahl der Museumsbesucher angezeigt werden.

Zunächst musst du aber zu deinem Programm eine zweite Variable hinzufügen, die die Zahl der Besucher am Eingang zählt ohne die Besucher am Ausgang wieder abzuziehen. Dazu schaltest du in der Funktionsleiste wieder um auf **Funktion** und fügst die Variable **Gesamt** wie folgt ein:



Wie du siehst kann ein Befehlselement auch verwendet werden, um an zwei Variablen gleichzeitig einen Befehl zu schicken. Die **-1** Befehle erhält die Variable **Gesamt** nicht, weil Befehle entlang der orangen Linien nur in Pfeilrichtung übertragen werden. Die **+1** Befehle werden dagegen an beide Variablen übermittelt. Das soll hier aber nur ein Beispiel sein. In der Regel ist es einfacher und übersichtlicher ein zweites Befehlselement zu verwenden.

Tipp: Wenn sich orange Linien verzweigen ist es oft praktischer die Linien vom Ziel zum Anfang zu zeichnen. Wenn du im obigen Beispiel die Linie zur Variablen **Gesamt** zeichnen möchtest, klicke zuerst auf den Eingang der Variable **Gesamt** und verlege die Linie dann rückwärts bis zum Verzweigungspunkt. Wenn du dagegen eine orange Linie auf einer beste-

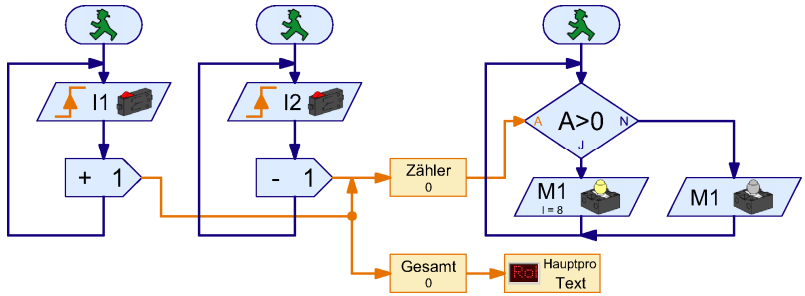
henden orangen Linie beginnen möchtest, musst du mit der linken Maustaste einen Doppelklick an der Stelle machen, wo die neue Linie anfangen soll.

So, nun hast du eine Textanzeige im Bedienfeld und eine Variable, die du in der Anzeige darstellen möchtest. Wie verbindet man nun die beiden? Da die Textanzeige und die Variable auf verschiedenen Seiten sind, kannst du die beiden ja schlecht mit einer Linie verbinden. Deswegen gibt es ein spezielles Element, das den Wert, der im Bedienfeld dargestellt werden soll, an die entsprechende Anzeige weiterleitet. Das oben abgebildete Element **Bedienfeldausgang** findest du am Ende der Gruppe **Eingänge, Ausgänge**. Füge so einen Bedienfeldausgang in dein Programm neben der Variablen **Gesamt** ein und verbinde den rechten Anschluss der Variablen mit dem Anschluss des **Bedienfeldausgangs**.

Da du in der Regel mehr als eine Anzeige in einem Bedienfeld haben wirst, musst du dem Bedienfeldausgang noch mitteilen, an welche Anzeige es die Variablenwerte schicken soll. Das geht ganz einfach über das Eigenschaftsfenster des Elements. Wenn du mit der rechten Maustaste auf den Bedienfeldausgang klickst, siehst du eine Auswahlliste, in der alle Anzeigen aufgelistet sind, die bereits im Bedienfeld eingefügt wurden. Da jedes Unterprogramm ein eigenes Bedienfeld haben kann, sind die Bedienfelder nach Unterprogrammen geordnet. In unserem Beispiel gibt es kein Unterprogramm, nur das Hauptprogramm. Darunter gibt es eine Anzeige mit dem Namen **Text**. Wähle diese Anzeige aus und klicke auf OK.

Sobald du den Bedienfeldausgang mit einer Anzeige verbunden hast, ändert sich das Symbol und die Beschriftung entsprechend. Der von uns verwendete Bedienfeldausgang stellt eine Verbindung mit der Textanzeige mit Namen **Text** im (Unter-)Programm **HAUPT** her.

Nachdem du den Bedienfeldausgang eingefügt und mit der Textanzeige verbunden hast, sieht dein Programm so aus:



Probiere es gleich einmal aus. Sobald du das Programm im Online-Modus gestartet hast, zeigt die Anzeige im Bedienfeld die Zahl der Besucher an, die das Drehkreuz am Eingang passiert haben.

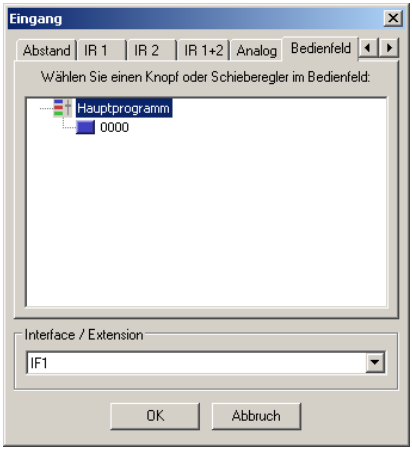
Hinweis: Falls du mehr als eine Anzeige in einem Bedienfeld verwenden möchtest, ist es wichtig, dass du jeder Anzeige einen anderen Namen gibst, damit du sie bei der Verknüpfung mit dem Programm unterscheiden kannst. Dazu drückst du mit der rechten Maustaste auf die Anzeige im Bedienfeld. Dort kannst du unter **ID / Name** einen Namen eingeben. Wenn du dann einen Bedienfeldausgang mit der Anzeige verknüpfst, erscheint dieser Name im Auswahlfenster des Bedienfeldausgangs. Da wir im Moment nur eine Anzeige haben, ist der Name aber unwichtig und wir behalten den Namen **Text** bei.

Das Programm ist noch nicht ganz perfekt. Was noch fehlt ist ein Schalter um den Zähler zurückzusetzen. Dazu wollen wir aber keinen normalen Taster, sondern einen Knopf verwenden, auf dem wir im Bedienfeld drücken können.

Den Bedienknopf findest du im Elementfenster unter der Gruppe **Bedienelemente** / **Steuerelemente**. Schalte in der Funktionsleiste auf **Bedienfeld** um und füge in dein Bedienfeld neben der Textanzeige einen Knopf ein. Die Beschriftung ***Knopf*** ist natürlich nicht ganz passend, lässt sich aber leicht über das Eigenschaftsfenster für den Knopf ändern. Klicke mit der rechten Maustaste auf den Knopf, gib bei **Beschriftungstext** zum Beispiel 0000 ein und bestätige mit **OK**.

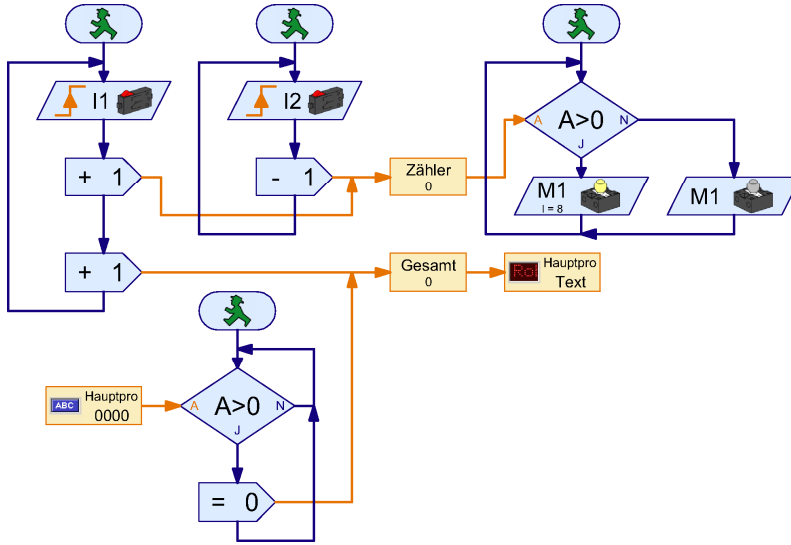
Genauso wie bei der Textanzeige benötigen wir auch ein Programmelement, das den Knopf mit dem Programmablauf verbindet. Schalte deshalb in der Funktionsleiste zunächst wieder auf **Funktion** um. Du findest im Elementfenster in der Gruppe **Eingänge, Ausgänge**. das abgebildete Element **Bedienfeldeingang.** Platziere es im Programmablauf unter dem bisherigen Ablauf.

Jetzt musst du noch den Bedienfeldeingang mit dem Knopf im Bedienfeld verknüpfen. Dazu klickst du mit der rechten Maustaste auf das Element Bedienfeldeingang. Die Steuerelemente sind wie bei den Anzeigen nach Unterprogrammen geordnet, da jedes Unterprogramm ein eigenes Bedienfeld haben kann. Wähle nun den Knopf **0000** aus und bestätige mit OK.



Vielleicht ist dir aufgefallen, dass man dieses Element über die Reiterleiste des Eigenschaftsfensters auf alle möglichen Arten von Eingängen einstellen kann. Das wird jedoch erst im übernächsten Abschnitt *Befehlseingänge für Unterprogramme* erklärt.

Den Wert, den der Bedienfeldeingang liefert, fragst du mit einem Verzweigungselement ab. Dieses Element hast du bereits verwendet, um die Variable abzufragen. Das fertige Programm mit Nullstellung sieht nun so aus:



Solange der Knopf ***0000*** gedrückt ist, wird dem Gesamtzähler ein **= 0** Befehl geschickt, der den Zähler auf 0 setzt.

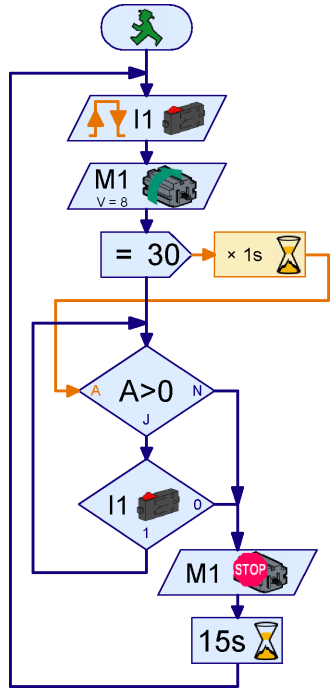
5.4 Timer

Der Museumsdirektor weiß nach deinen Erfolgen gar nicht mehr, was er ohne dich anfangen soll, und ernennt dich daher zum Computerberater des Museums. So ein Posten bringt natürlich viel Ruhm und Ehre mit sich, aber auch viel Arbeit, und zwar folgende: In dem Museum gibt es viele Modelle, die sich auf Knopfdruck bewegen. Nun drücken aber manche Besucher ziemlich lange auf die Knöpfe, so dass die Modelle heiß laufen und so immer wieder zur Reparatur müssen. Der Direktor möchte nun, dass die Modelle solange laufen, wie der Knopf gedrückt wird, höchstens aber 30 Sekunden am Stück. Wenn das Modell einmal gelaufen ist, soll es eine Pause von 15 Sekunden einlegen, bis es wieder eingeschaltet werden kann.

Hmm, gar kein Problem denkst du nun vielleicht: Ein paar Wartezeiten, ein paar Programmverzweigungen und fertig. Versuche es ruhig einmal! Nach einer Weile wirst du feststellen, dass das gar nicht so einfach ist, und zwar aus zwei Gründen:

- Während der Zeit von 30 Sekunden muss das Programm den Knopf abfragen um festzustellen, ob der Knopf schon vor Ablauf der 30 Sekunden losgelassen wird. Na gut, zugegeben, das kann man durch zwei Prozesse lösen, die gleichzeitig ablaufen, siehe Abschnitt 5.2 *Variablen und mehrere Prozesse* auf Seite 38.
- Wenn ein Besucher nach 5 Sekunden den Knopf los lässt, und dann nach 15 Sekunden wieder drückt, muss die Wartezeit von 30 Sekunden wieder neu gestartet werden. Die Wartezeit ist aber erst 5 + 15 = 20 Sekunden gelaufen und damit noch aktiv. Auch mit parallel ablaufenden Prozessen kann man eine Wartezeit nicht neu starten. Eventuell geht es mit zwei Wartezeiten in drei Prozessen, die du abwechselnd startest, aber da bekommt man ja schon langsam Kopfweh vom Nachdenken.

Gibt es da nicht etwas Einfacheres? Doch, und zwar ***Timervariablen***, oder kurz ***Timer***. Ein Timer funktioniert zunächst mal wie eine ganz normale Variable. Der Timer merkt sich eine Zahl und du kannst die Zahl mit **=**, **+** und **–** Befehlen verändern. Das Besondere an einem Timer ist nun, dass er die Zahl von selber regelmäßig bis auf 0 herunterzählt. Die Zeit für einen Zählschritt kannst du in Schritten zwischen einer tausendstel Sekunde und einer Minute einstellen. Mit Timern kann man viele Zeitsteuerungsaufgaben viel eleganter lösen, als mit Wartezeiten. Siehst du schon, wie du die Aufgabe mit einem Timer lösen kannst?



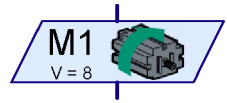
Richtig: Sobald der Besucher den Taster an **I1** drückt. startest du das Modell und setzt dann den Timer mit einem **=** Befehl auf 30 x 1 Sekunde = 30 Sekunden. Dann fragst du in einer Schleife ab, ob die Zeit von 30 Sekunden abgelaufen ist oder ob der Taster an **I1** losgelassen wurde. Wenn eine der beiden Abbruchkriterien erfüllt ist, stoppst du das Modell und wartest **15** Sekunden. Danach geht wieder alles von vorne los.

Zugegeben, langsam werden die Programme etwas anspruchsvoller. Aber versuche einmal diese Aufgabe zu lösen: Entwickle ein Programm gleicher Funktion mit Wartezeiten anstatt mit Timern! ***Achtung: Dies ist eine sehr schwierige Aufgabe und nur für diejenigen gedacht, die gerne auch mal etwas länger an einem Rätsel herumknobeln! Alle Anderen gehen einfach weiter zum nächsten Abschnitt.*** Es gibt für diese Aufgabe zwei Lösungsansätze: Du kannst zwei Wartezeiten verwenden, die du abwechselnd in eigenen Prozessen startest. Da es eine Auszeit von 15 Sekunden gibt, ist eine der beiden Wartezeiten spätestens nach dem zweiten Durchlauf abgelaufen, so dass sie dann neu gestartet werden kann. Eine andere Alternative wäre, einen Timer mit einer normalen Variable und einem Element **Wartezeit** mit einer kurzen Wartezeit von zum Beispiel einer Sekunde nachzubauen.

5.5 Befehlseingänge für Unterprogramme

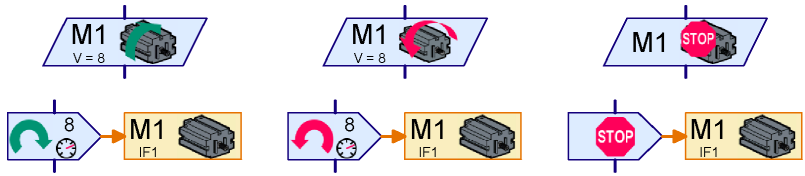
Wie immer funktioniert dein Programm ausgezeichnet und fischertechnik freut sich, weil im Museum alle Modelle mit dem ROBO Interface ausgerüstet werden. Nur leider sind wie überall in öffentlichen Einrichtungen auch im Museum die Kassen leer. Daher möchte der Direktor mit so wenig Interfaces wie möglich auskommen. Immerhin hat ein ROBO Interface vier Motorausgänge und auch genügend Eingänge um vier Modelle anzusteuern. Da sich die meisten Modelle nur in einer Richtung drehen können, kannst du sogar 8 Modelle über die einpoligen Ausgänge O1 bis O8 ansteuern.

Das spart dem Museumsdirektor natürlich eine Menge Geld. Dafür musst du jetzt das Programm 7 Mal kopieren und überall die Ein- und Ausgänge anpassen. Oder doch nicht? Könnte man das nicht auch mit Unterprogrammen machen?



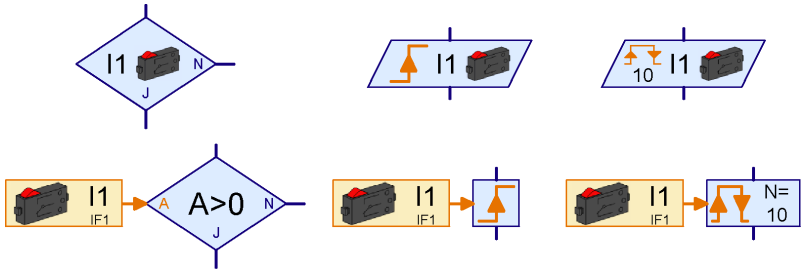
Könnte man schon, aber dabei taucht ein Problem auf: Wenn du in einem Unterprogramm die üblichen Tasterabfragen und Motorelemente aus der Gruppe **Grundelemente** verwendest, fragt jeder Aufruf des Unterprogramms den gleichen Taster ab und steuert die gleichen Motoren an. Das liegt daran, dass zum Beispiel in einem Motorausgangselement der Steuerbefehl für den Motor (rechts, links oder stopp) und die Motorausgangsnummer (M1…M8) eine Einheit bilden. Da es ein Unterprogramm aber nur einmal gibt, steht da auch immer der gleiche Motor drin. Änderst du in einem Unterprogrammaufruf die Nummer des Motorausgangs, wird sie in allen vorhandenen Aufrufen des Unterprogramms ebenfalls geändert. Du müsstest also wieder das Unterprogramm 7 Mal kopieren, jedem Unterprogramm einen anderen Namen geben und überall die Ein- und Ausgänge von Hand anpassen.

Aber man kann dieses Problem viel eleganter lösen. Der Trick ist, die Steuerbefehle von den Motorsymbolen zu trennen. Dann kann man die Steuerbefehle (links, rechts, stopp) ins Unterprogramm und die Motorelemente ins Hauptprogramm setzen. Im Unterprogrammablauf schickst du dann über ein Befehlselement, das du schon bei den Variablen kennen gelernt hast, die Befehle links, rechts oder stopp an das Hauptprogramm, wo du sie dann zu verschiedenen Motoren weiterleiten kannst. Für den Motor gibt es ein Motorelement, das nur einen Motor repräsentiert ohne festzulegen, was der Motor machen soll. Dieses Element hat einen Befehlseingang, an den du Befehle schicken kannst. Die Elemente aus der Gruppe Grundelemente kannst du wie folgt durch ein Befehlselement und ein Motorelement ersetzen:



In der oberen Zeile siehst du jeweils ein Motorelement aus der Gruppe **Grundelemente**. In der zweiten Zeile ist die Kombination aus einem Befehlselement der Gruppe **Befehle** und ein Motorelement der Gruppe **Eingänge, Ausgänge** abgebildet, die genau den gleichen Effekt hat. Tatsächlich sind die oberen Elemente nur Abkürzungen oder Vereinfachungen für die Kombinationen in der unteren Zeile. Beide senden an den Motor **M1** einen Befehl rechts, links oder stopp.

Das Gleiche geht auch für die Abfrage von Tastern:



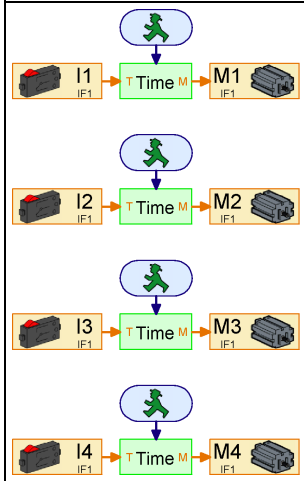
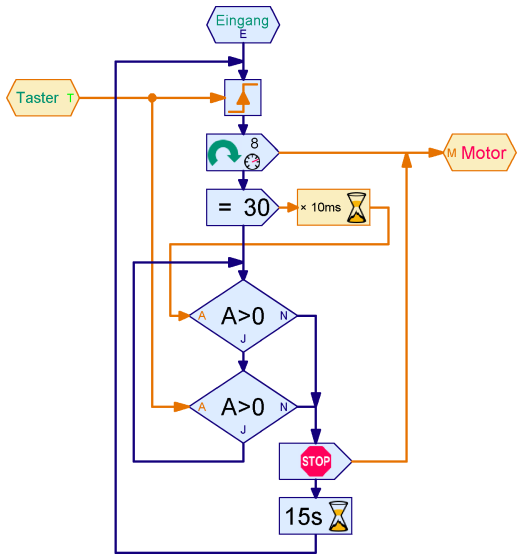
In der oberen Zeile siehst du wieder Elemente aus der Gruppe **Grundelemente**. In der unteren Reihe findest du jeweils eine Kombination aus einem Digitaleingang und einem Element aus der Gruppe **Verzweigung, Warten, ...**. Das orange Element **Digitaleingang** findest du wie das Motorelement in der Gruppe **Eingänge, Ausgänge**.

Mit diesem Trick kannst du die Logik eines Programmablaufs von den Ein- und Ausgängen trennen. Etwas fehlt aber noch. Wenn die Motor- und Tasterelemente im Hauptprogramm stehen sollen und die Befehle in einem Unterprogramm, muss es ja einen Weg geben die Taster und Motorelemente mit dem Unterprogramm zu verbinden. Die dafür nötigen Anschlusselemente findest du in der Gruppe **Unterprogramm I/O**.

Über einen Unterprogramm-Befehlseingang kannst du Befehle von außen in ein Unterprogramm hinein schicken. Das Digitaleingangselement (Taster) schickt über die orange Linie seinen neuen Wert wenn sich der Zustand des Eingangs ändert (mit einem so genannten „= Befehl“). Im Dialogfeld des Elements kannst du dem Eingang einen Namen geben.

Über einen Unterprogramm-Befehlsausgang kannst du Befehle aus einem Unterprogramm hinaus schicken. So kannst du zum Beispiel die Befehle links, rechts oder stopp aus einem Unterprogramm heraus an einen Motor schicken. Auch bei diesem Element kannst du im Dialogfeld einen Namen eingeben.

Nun hast du alles beisammen für deinen ***Mehrfachmodelltimer mit Unterprogrammen***:

Hauptprogramm	Unterprogramm **Time**
	

Das **Unterprogramm Time** ist fast genau gleich wie das Programm aus dem vorigen Abschnitt. Die Elemente **Warten auf Digitaleingang I1** am Anfang und in der Schleife sind aber durch **Warten-auf-**Elemente mit Datenanschlüssen für orange Linien aus der Gruppe **Verzweigung, Warten, ...** ersetzt worden. Beide sind mit dem Unterprogramm-Befehlseingang **Taster** verbunden. Die zwei Motorsteuerelemente am Anfang und am Ende des Programms sind durch Befehlselemente ersetzt worden. Beide senden ihre Befehle an den Unterprogramm-Befehlsausgang **Motor**.

Im **Hauptprogramm** wird das Unterprogramm **Time** vier Mal aufgerufen. Der Unterprogramm-Befehlseingang **Taster** hat an dem grünen Unterprogrammsymbol auf der linken Seite automatisch den orangen Anschluss **T** erzeugt. Durch den Unterprogramm-Befehlsausgang **Motor** ist auf der rechten Seite der Anschluss **M** entstanden. Der Anschluss **T** des Unterprogrammsymbols wird

jeweils mit einem der Taster **I1** bis **I4** verbunden. An den Anschluss **M** wird jeweils einer der Motoren **M1** bis **M4** angeschlossen. Auf diese Weise fragt jeder Aufruf des Unterprogramms **Time** einen anderen Taster ab und steuert einen anderen Motor!

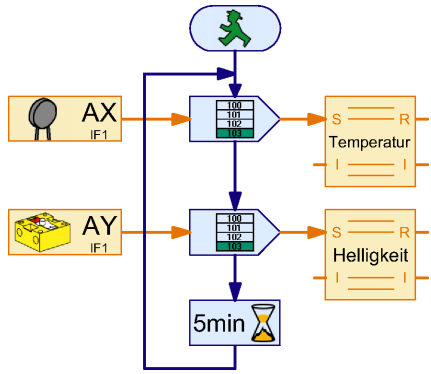
Versuche das obige Unterprogramm und Hauptprogramm nachzuzeichnen und probiere es aus. Du musst zuerst das Unterprogramm zeichnen, weil du das Unterprogramm sonst nicht in das Hauptprogramm einfügen kannst. Falls du Schwierigkeiten mit dem Unterprogramm hast, lies noch einmal im Kapitel 4 *Level 2: Arbeiten mit Unterprogrammen* auf Seite 28 nach.

5.6 Listen (Arrays)

Nachdem alle Versuchsaufbauten im Museum mit deiner Kosten sparenden Steuerung ausgerüstet sind, lässt das nächste Problem des Museumsdirektors nicht lange auf sich warten: In einem Raum mit sehr wertvollen antiken Ausstellungsstücken ist es in letzter Zeit zu schädlichen Temperaturschwankungen gekommen. Du vermutest, dass dies mit der Sonneneinstrahlung zusammenhängt. Um diesen Zusammenhang zu beweisen möchtest du ein Gerät bauen, das die Helligkeit und die Temperatur aufzeichnet. Das ROBO Interface hat ja mehrere Analogeingänge und du weißt bereits auch wie man mit Hilfe von Variablen Werte speichern kann. Das Ganze sollte also kein Problem sein, oder doch? Um über 12 Stunden alle 5 Minuten zwei Werte aufzunehmen braucht man 288 Variablen! Das wird aber ein riesiges und unübersichtliches Programm werden. Kann man das vielleicht wieder mit Unterprogrammen vereinfachen? Schon, aber es gibt einen viel besseren Weg: Das Element **Liste** (Programmierer nennen es „Array“).

In einer Liste kann man nicht nur einen Wert sondern eine ganze Liste von Werten speichern. Am Anfang ist die Liste in der Regel leer. Wenn du an den oberen linken Dateneingang mit der Bezeichnung **S** einen Befehl **Anhängen** schickst, wird der Wert, der in diesem Befehlselement angegeben ist, am Ende der Liste angehängt. Die maximale Länge der Liste kannst du zwischen 1 und 32767 über das Eigenschaftsfenster des Elements **Liste** einstellen. Damit wird das Programm zur Aufzeichnung von Temperatur und Helligkeit ganz einfach:

Am Analogeingang **AX** ist der Temperatursensor und am Analogeingang **AY** der Helligkeitssensor angeschlossen. Das Programm liest in einer Schleife alle 5 Minuten beide Werte ein und fügt sie über den Befehl **Anhängen** jeweils einer Liste hinzu.

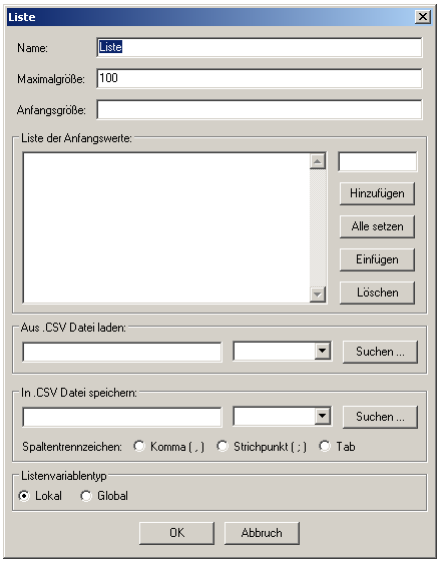


> ***Hinweis:*** Beim Einfügen des Befehlselements musst du im Eigenschaftsfenster die Option **Dateneingang für Befehlswert** aktivieren. Dann erscheint links am Befehlselement ein Dateneingang, an den du den Analogeingang anschließen kannst.

Zum Testen des Programms ist es hilfreich die Schleifenzeit von 5 Minuten auf einige Sekunden zu verringern.

Nun fragst du dich sicherlich, wie du die gespeicherten Werte wieder aus der Liste auslesen kannst. Dazu gibt es zwei Möglichkeiten: Du kannst die Werte wie bei einer gewöhnlichen Variable auslesen und in deinem Programm weiter verarbeiten. Da die Liste mehrere Elemente enthält, wählst du zuerst am linken Dateneingang mit der Bezeichnung **I** die Nummer des Elements aus,

das du auslesen willst. Dann wird der Wert, den dieses Element besitzt, am Datenausgang **R** auf der rechten Seite ausgegeben.



ROBO Pro kann aber auch alle Werte der Liste in eine Datei auf deinem Computer speichern, die du dann z. B. in Excel weiter verarbeiten kannst. Da du im vorliegenden Fall die aufgezeichneten Helligkeiten und Temperaturen nur anschauen und vergleichen möchtest, ist das sicherlich praktischer. ROBO Pro speichert die Werte in einer so genannten ***CSV-Datei*** (comma separated values = kommagetrennte Werte). CSV-Dateien sind Textdateien, die eine oder mehrere Spalten mit je einer Datenreihe enthalten. Du kannst also auch mehrere Messreihen wie Temperatur und Helligkeit in verschiedenen Spalten einer CSV-Datei speichern. Die Spalten sind durch Komma getrennt. In Ländern, in denen man 0,5 mit Komma und nicht 0.5 mit Punkt schreibt (z. B. Deutschland), wird als Trennzeichen für die Spalten häufig ein **Strichpunkt (;)** verwendet. Falls du Probleme beim Austausch von CSV-Dateien zwischen ROBO Pro und zum Beispiel Microsoft Excel hast, kannst du im Eigenschaftsfenster der Liste das **Spaltentrennzeichen** ändern.

Den Namen der CSV-Datei und die Spalte, in welcher der Inhalt einer Liste gespeichert werden soll, kannst du im Eigenschaftsfenster der Liste unter **CSV-Datei speichern** einstellen. Die Daten werden gespeichert, wenn das Programm im Online-Modus beendet wird, oder wenn du im Menü **Datei** den Punkt **CSV-Dateien speichern** auswählst, so lange das Programm noch läuft (Online oder Download). Im Downloadmodus kannst du das ROBO Interface zum Aufzeichnen der Daten vom PC trennen und dann zum Speichern wieder anschließen.

Nachdem du das obige Programm im Online-Modus ausgeführt hast, kannst du die von ROBO Pro erzeugte .CSV Datei mit den Daten in Microsoft Excel oder einem anderen Tabellenkalkulationsprogramm öffnen. Falls du kein Tabellenkalkulationsprogramm hast, kannst du auch den Windows Editor (Notepad.exe) verwenden, den du meistens im Windows Startmenü unter Zubehör findest.

Solange das Programm im Online-Modus läuft, kannst du die Daten in einer Liste auch ansehen, indem du mit der rechten Maustaste auf das Listenelement klickst.

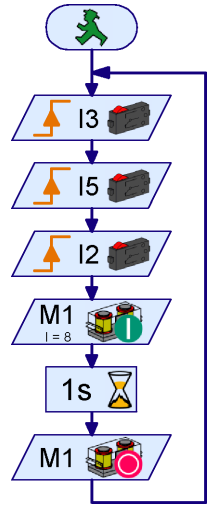
5.7 Operatoren

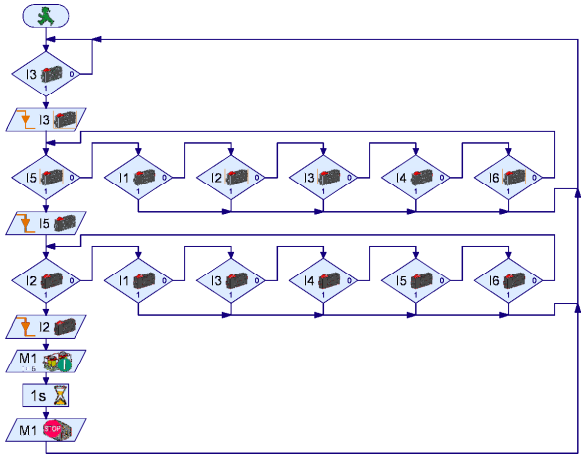
Dein Programm zur Aufzeichnung von Helligkeit und Temperatur hat gut funktioniert, aber aus der Aufzeichnung ergibt sich, dass die Temperatur in dem Ausstellungsraum des Museums nichts mit der Sonne zu tun hat. Wie sich herausstellt, haben einige der Besucher die Klimasteuerung in dem Ausstellungsraum mit einer Modellsteuerung verwechselt und fleißig daran herumgedrückt. Kein Wunder, dass die Temperatur im Ausstellungsraum verrückt spielt!

Aber dieses Problem lässt sich leicht beheben, und zwar mit einem elektronischen Zahlenschloss. Das Zahlenschloss soll ein Tastenfeld mit Tasten von 1 bis 6 haben. Wenn 3 Ziffern hintereinander

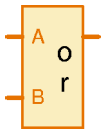
richtig eingegeben sind, soll das Tastenschloss über einen Magneten die Abdeckung der Klimasteuerung frei geben.

Auf den ersten Blick ist so ein Tastenschloss ganz einfach: das Programm wartet einfach ab, bis die richtigen Tasten in der richtigen Reihenfolge gedrückt werden. Rechts ist so ein Programm für die Kombination 3-5-2 zu sehen. Auf den zweiten Blick hat dieses Programm aber ein Problem: Man kann das Schloss ganz leicht knacken, indem man 3 Mal hintereinander auf alle Tasten von 1 bis 6 drückt. Damit hat man dann auf jeden Fall immer auch die richtige Taste gedrückt. Wie sagte Albert Einstein so schön: „Man sollte die Dinge so einfach wie möglich machen –- aber nicht einfacher“. Das Programm muss also nicht nur abfragen, ob die richtigen Tasten kommen, sondern auch ob eine falsche Taste gedrückt wird. Das Programm sieht dann so aus:

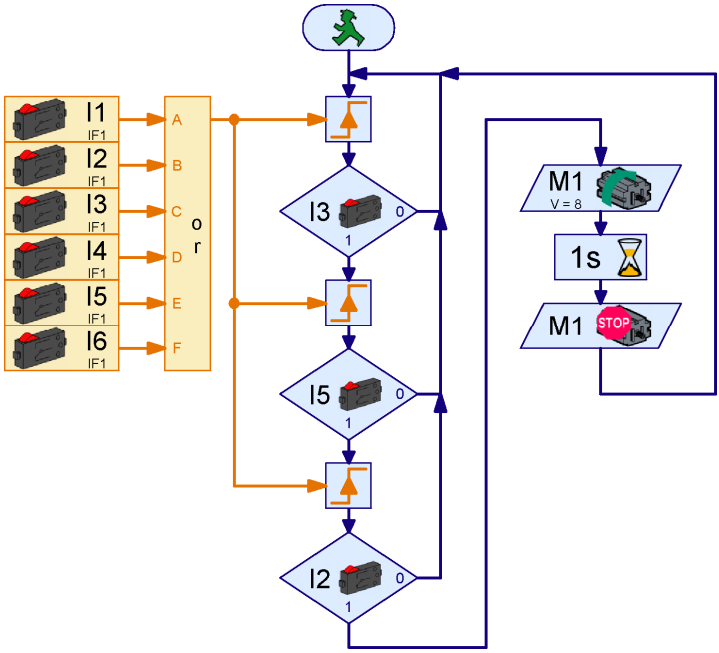




Dieses Programm öffnet das Schloss nur, wenn die Tasten 3 5 2 gedrückt werden, ohne dass dazwischen irgendeine andere Taste gedrückt wird. Wenn zum Beispiel die Taste 3 gedrückt wird, wartet das Programm zunächst bis die Taste wieder losgelassen wird. Wird anschließend irgendeine andere Taste als die Taste 5 gedrückt, beginnt das Programm wieder von vorne. Das Programm funktioniert also richtig, aber einfach und übersichtlich ist es nicht. Außerdem ist es recht schwierig den Code zu ändern. Aber keine Angst, es geht auch einfach und richtig, und zwar mit ***Operatoren***. Es gibt verschiedene Arten von Operatoren. Du findest sie unter **Programmelemente** in der Gruppe **Operatoren**. Für das Zahlenschloss benötigen wir zunächst einen **Oder Operator**.



An die Eingänge des **Oder Operators** (English **or**) kann man mehrere Signale anschließen. Am Ausgang liefert der Operator immer dann 1, wenn mindestens einer der Eingänge 1 (oder größer 0) ist. Wenn man an die Eingänge des **Oder Operators** mehrere Taster anschließt, ist der Ausgang des Operators immer 1, wenn mindestens einer der Taster gedrückt ist. Die Zahl der Eingänge kann über das Eigenschaftsfenster des Operators bis 26 eingestellt werden. Man kann also auch alle 6 Taster an einen Operator anschließen. Du fragst dich nun vielleicht, wie man damit das Zahlenschloss vereinfachen kann? Ganz einfach: mit dem Operator kannst du in jedem Schritt zunächst warten bis irgendeine Taste gedrückt ist. Dann kannst du nachprüfen, ob es die richtige Taste ist. Pro Ziffer brauchst du dann statt 7 nur noch 2 Programmelemente.

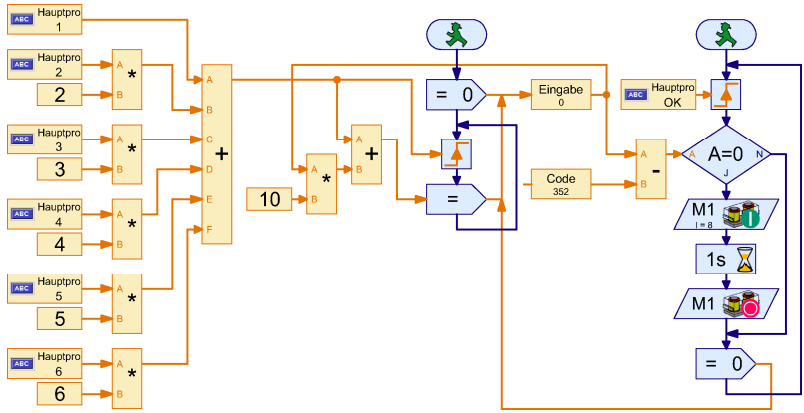


Die Taster an den Eingängen I1 bis I6 werden über einen Oder Operator mit 6 Eingängen zusammengefasst. Wenn mindestens einer der Taster gedrückt ist, liefert der Oder Operator einen Ausgangswert von 1, sonst 0. Mit einem **Warten auf** Element wartet das Programm so lange, bis einer der Taster gedrückt wird. Im Anschluss daran wird sofort getestet, ob es der richtige Taster war. Falls ja, wird wieder auf einen Tastendruck gewartet. Falls eine falsche Taste gedrückt wurde, beginnt das Programm wieder von vorne.

Ändere das obige Programm so, dass es statt der Taster Bedienelemente in einem Bedienfeld verwendet. Zeichne dazu zunächst ein Bedienfeld mit 6 Knöpfen mit der Beschriftung 1 bis 6. Ändere dann die Digitaleingänge über das Eigenschaftsfenster. Die Verzweigungen musst du durch Verzweigungen mit Dateneingang und Bedienfeldeingänge ersetzen.

Das Zahlenschloss funktioniert nun einwandfrei, aber den Code (3 5 2) zu ändern ist nach wie vor nicht so ganz einfach. Man muss dazu die Eingänge in drei Verzweigungselementen ändern. Für die Klimasteuerung des Museums ist es nicht erforderlich den Code regelmäßig zu ändern, aber wenn du das Codeschloss zum Beispiel für eine Alarmanlage verwendest, möchtest du den Code vermutlich regelmäßig ändern. Einfacher wäre es natürlich, wenn man den Code in einer Variablen speichern könnte. Dann könnte man den Code sogar automatisch ändern. Wenn zum Beispiel in der Alarmanlage ein stiller Alarm ausgelöst wird, könnte man den normalen Code durch einen besonderen Alarmcode ersetzen.

Damit du die Eingaben mit der Code-Variablen vergleichen kannst, musst du die Eingaben selbst ebenfalls in einer Variablen speichern. Am Anfang soll die Variable mit dem Eingabewert den Wert 0 haben. Wenn du nun die Taste 3 drückst, soll die Variable den Wert 3 haben. Nach dem nächsten Tastendruck auf die Taste 5 den Wert 35 und schließlich nach einem Druck auf die Taste 2 den Wert 352.



Das Zahlenschloss mit Codevariable hat zwei Prozesse. Im linken Prozess wird mit einigen Mal-Operatoren und einem Plus-Operator jeder Taste eine Zahl zugeordnet. Der Taste 1 die Zahl 1, der Taste 2 die Zahl 2 und so weiter. Die Tasten liefern einen Wert von 1 oder 0 und wenn man den Wert mit einer festen Zahl X multipliziert, ergibt sich ein Wert von 0 oder X. Da die Werte 0 sind, wenn die Tasten nicht gedrückt sind, kann man alle Werte aufaddieren und bekommt so den Tastenwert als Zahl. Sobald eine Taste gedrückt wird, wird der Eingabevariablen 10 mal der vorherige Wert plus der Wert der gedrückten Taste zugewiesen. Die Multiplikation mit 10 schiebt den bisherigen Wert der Eingabevariablen eine Zehnerstelle weiter (aus 35 wird zum Beispiel 350).

Der rechte Prozess wartet so lange, bis nach der Eingabe des Codes die OK Taste im Bedienfeld gedrückt wird. Dann wird die Codevariable Code, die bei richtig eingegebenem Code den Wert 352 hat, mit der Eingabevariablen verglichen. Wenn beide den gleichen Wert haben, wird der Öffnungsmagnet ausgelöst, sonst nicht. Schließlich wird die Eingabevariable wieder auf 0 gesetzt. Die Variablen Eingabe und Code werden miteinander verglichen, indem ihre Differenz mit 0 verglichen wird. Du hättest auch ein Vergleichselement verwenden können.

Wenn du zwei Tasten gleichzeitig drückst, werden die Werte der Tasten addiert. Wenn du zum Beispiel 3 und 6 gleichzeitig drückst, ergibt sich der Wert 9. Damit kannst du ein Supergeheimschloss bauen, bei dem man zuweilen mehrere Tasten gleichzeitig drücken muss. Überlege dir, welche Tasten du in welcher Reihenfolge drücken musst, damit sich das Schloss bei einem Code von 495 öffnet. Beachte dabei, dass das Element **Warten auf…** das Programm fortsetzt, wenn sich der Wert erhöht, nicht nur wenn er sich von 0 nach 1 ändert.

Funktioniert das Zahlenschloss auch für 2 oder 4 stellige Codes? Wenn ja, bis zu welcher Stellenzahl funktioniert es und warum? Wie ist das mit den anderen Zahlenschlossprogrammen?

6 Erweiterungsmodule und mehrere Interfaces ansteuern

Mit einem ROBO Interface oder einem Intelligent Interface kann man bereits ziemlich aufwändige Modelle steuern. Aber der eine oder andere von euch mag es vielleicht noch ein bisschen komplizierter. Falls du mit 8 Tastern, 4 Motoren und 4 Analogeingängen nicht auskommst, kannst du dein ROBO Interface mit bis zu 3 ***ROBO I/O-Extensions*** oder dein Intelligent Interface mit einem ***Extensionmodul*** erweitern. Und für die, denen selbst das nicht ausreicht, gibt es noch die Möglichkeit im Onlinemodus mehrere Interfaces (jedes möglicherweise mit Erweiterungsmodulen), von einem Programm aus anzusteuern. Die Zahl der Ein- und Ausgänge ist also zumindest im Onlinemodus nur durch die Leistungsfähigkeit deines PCs begrenzt.

6.1 Erweiterungsmodule

Vielleicht ist dir die Auswahlliste unter **Interface / Extension** in den Eigenschaftsfenstern für Ein- und Ausgangselemente schon aufgefallen. Dort kannst du auswählen, an welchem Interface oder Extension Modul sich der Ein- oder Ausgang befindet. Sofern du nichts anderes eingestellt hast (siehe nächster Abschnitt) hat die Liste die folgenden Einträge:

- **IF1:** Das ist das eigentliche Interface (also kein Extensionmodul)
- **EM1..EM3:** Das sind die Erweiterungsmodule 1 bis 3. Wenn du ein Intelligent Interface verwendest kannst du nur ein Erweiterungsmodul verwenden (**EM1**). Die Erweiterungsmodule **EM2** und **EM3** werden aber auch angezeigt, wenn du mit einem Intelligent Interface arbeitest, weil die Programme unabhängig davon sein sollen, ob du ein ROBO Interface oder ein Intelligent Interface verwendest.

Es ist also ganz leicht Erweiterungsmodule wie ***ROBO I/O Extensions*** zu verwenden. Du brauchst dazu bei den Ein- und Ausgängen lediglich das gewünschte Modul (Interface oder Extension) auswählen.

6.2 Mehrere Interfaces

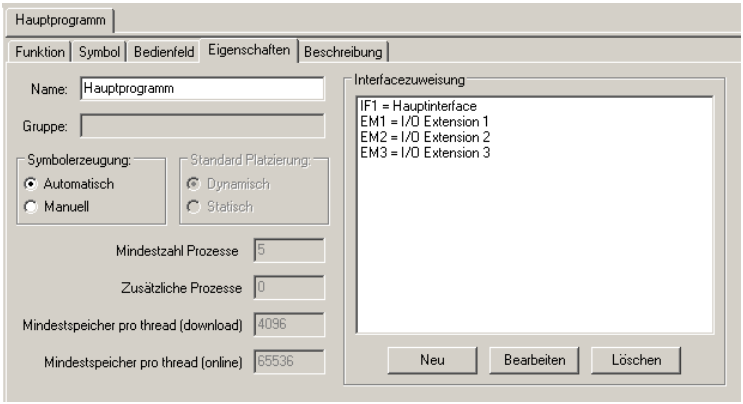
Wenn du mehrere Interfaces aus einem Programm heraus ansteuern willst, funktioniert das nur im Onlinemodus. Du kannst z. B. ein ROBO Interface an COM1 anschließen, ein zweites an USB. Beide können mit bis zu 3 ROBO I/O-Extensions ausgestattet sein. Eine andere Kombinationsmöglichkeit wäre z. B. 2 ROBO Interfaces an USB (beide mit I/O-Extensions) und ein Intelligent Interface an COM1 (gerne auch mit Extensionmodul). Damit du im Eigenschaftsfenster der Ein- und Ausgänge definieren kannst, welches Interface angesprochen werden soll, musst du die Interfacezuweisung anpassen.

Solange du nichts anderes einstellst, findest du in den **Interface / Extension** Auswahllisten der Ein- und Ausgänge die Einträge **IF1**, **EM1**, **EM2** und **EM3**. Du kannst die Liste aber auch ändern oder erweitern. Dafür kann es mehrere Gründe geben:

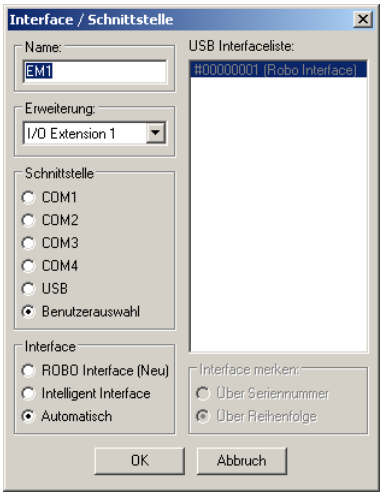
- Zu besseren Übersicht möchtest du den Modulen statt IF1 oder EM1 einen Namen geben, der angibt welchen Teil deiner Maschine oder deines Roboters das Modul steuert.

- Du möchtest zwei Erweiterungsmodule (zum Beispiel EM1 und EM2) vertauschen, weil das mit den Kabeln besser geht, aber dein Programm nicht ändern.
- Du möchtest ein Programm, das für ein ROBO Interface mit mehr als einem Erweiterungsmodul geschrieben wurde, mit mehreren Intelligent Interfaces laufen lassen.
- Du möchtest in deinem Programm mehr als ein ROBO Interface oder mehr als ein Intelligent Interface verwenden.

All das kannst du ganz einfach machen, indem du die **Interfacezuweisung** im Eigenschaftsfenster des ***Hauptprogramms*** änderst:



Hier kannst du sehen, welche Module (Interface oder Extension) den Namen **IF1** bis **EM3** zugeordnet sind. Mit dem Button **Neu** kannst du ein neues Interface hinzufügen. Willst du einen Eintrag in der Liste ändern, wählst ihn aus und klickst auf **Bearbeiten.** In beiden Fällen wird das folgende Fenster angezeigt:



- Unter **Name** kannst du den Namen für das Modul ändern. Der Name sollte nicht zu lang sein, weil der Platz für den Interfacenamen in den Grafiksymbolen sehr klein ist. Wenn du den Namen änderst, musst du meistens auch den Modulamen in allen Ein- und Ausgangselementen ändern, die diesen Namen verwenden.
- Unter **Erweiterung** kannst du einstellen, ob der Name sich auf ein Interface oder auf eines der Erweiterungsmodule 1 bis 3 bezieht.

COM/USB

- Unter **Schnittstelle** kannst du auswählen, an welcher Schnittstelle das Interface angeschlossen ist. Wenn du hier **Benutzerauswahl** angibst, wird das Interface verwendet, das du in der Werkzeugleiste unter **COM/USB** ausgewählt hast. Solange du nur ein Interface mit mehreren Erweiterungsmodulen aber nicht mehrere Interfaces verwenden möchtest, ist das am einfachsten, weil so auch jemand anders dein Programm ohne Änderung benutzen kann.

Verwendest du mehrere Interfaces, stellst du hier die Schnittstelle ein, an der das jeweilige Interface angeschlossen ist.

- Unter **Interface** kannst du angeben, ob du ein ROBO Interface oder ein Intelligent Interface verwenden möchtest. Ist das Interface mit der seriellen Schnittstelle verbunden, kann das Programm automatisch erkennen ob es sich um welches Interface es sich handelt (Auswahl **Automatisch**).
- Der rechte Teil des Fensters ist nur wichtig, wenn du mehrere ROBO Interfaces am USB Bus angeschlossen hast. Wenn du unter **Schnittstelle** auf **USB** klickst, kannst du unter **USB Interfaceliste** eines der Interfaces auswählen.

Achtung: Wenn du mehrere Interfaces am USB-Bus betreiben möchtest, musst du zunächst jedem Interface eine eigene Seriennummer zuweisen. Standardmäßig werden alle Interfaces mit der gleichen Seriennummer ausgeliefert, um Probleme beim Austausch von Interfaces zu vermeiden. Das Windows Betriebssystem erkennt jedoch nur Interfaces mit verschiedenen Seriennummern. Mehr dazu erfährst du in Abschnitt 6.5 *Ändern der Interface-Seriennummer oder Firmwareversion* auf Seite 54.

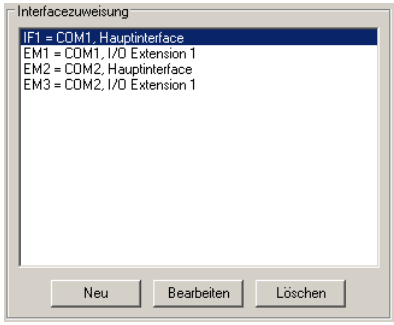
- Unter **Interface merken** kannst du einstellen, wie sich das Programm das ausgewählte Interface merkt. Hier gibt es zwei Möglichkeiten: Wenn du **Über Seriennummer** auswählst, speichert das Programm die Seriennummer des Interfaces. Auch wenn du weitere Interfaces an den USB-Bus anschließt und entfernst, kann das Programm das ausgewählte Interface immer wieder anhand der Seriennummer finden. Allerdings hat das den Nachteil, dass das Programm nur noch mit einem Interface mit gleicher Seriennummer funktioniert. Wenn du ein Interface mit einer anderen Seriennummer verwenden möchtest, musst du die Interfacezuweisung oder die Seriennummer des Interfaces ändern. Um Probleme mit den Seriennummern zu umgehen gibt es die zweite Möglichkeit **Über Reihenfolge**. Wenn du diesen Punkt auswählst, speichert das Programm statt der Seriennummer die Reihenfolge in der Liste. Das kann zwar zu einem Durcheinander führen, wenn du Interfaces am USB-Bus hinzufügst oder entfernst, aber das Programm läuft unverändert mit jedem beliebigen Interface.

6.3 Interfacezuweisungen in Unterprogrammen

Normalerweise machst du alle Interfacezuweisungen für dein Programm im Eigenschaftsfenster des Hauptprogramms. Du kannst aber auch in Unterprogrammen Interfacezuweisungen eingeben. In dem Unterprogramm kannst du dann die Interfacezuweisungen des Unterprogramms und des Hauptprogramms verwenden. Wenn zwei Zuweisungen den gleichen Namen haben, hat die Zuweisung im Unterprogramm Vorrang. So kannst du zum Beispiel definieren, dass im Hauptprogramm IF1 auf das Hauptinterface zugreift, in einem bestimmten Unterprogramm IF1 aber für ein Erweiterungsmodul steht. Das ist sehr praktisch wenn du einen ganzen Maschinenpark steuern möchtest, wobei jede Maschine von einem eigenen Interface gesteuert wird. Du kannst dann die Steuerungen für die einzelnen Maschinen erst als unabhängige Programme entwickeln, wobei jedes Hauptprogramm auf IF1 zugreift. Später kannst du alle Maschinenhauptprogramme in einem Gesamtprogramm als Unterprogramm einfügen. Im Gesamtprogramm brauchst du dann nur die Interfacezuweisungen zu ändern, aber nicht den Namen in jedem einzelnen Ein- und Ausgang.

6.4 Tipps & Tricks

Wenn du ein Programm, das für ein ROBO Interface mit 3 Erweiterungsmodulen entwickelt wurde, auf 2 Intelligent Interfaces mit je einem Erweiterungsmodul laufen lassen möchtest, kannst du die abgebildete Interfacezuordnung verwenden. Die Erweiterungsmodule 2 und 3 werden dabei durch ein weiteres Intelligent Interface mit Erweiterungsmodul an COM2 ersetzt.



6.5 Ändern der Interface-Seriennummer oder Firmwareversion

Standardmäßig werden alle ROBO Interfaces und ROBO I/O-Extensions mit der gleichen Seriennummer ausgeliefert. Solange du nur ein Interface an einem Computer verwenden möchtest, ist das praktischer, weil auf diese Weise alle Interfaces für den Computer gleich aussehen und es keine Probleme beim Austauschen von Interfaces gibt. Wenn du aber mehr als ein Interface an einem Computer über USB betreiben willst, musst du vorher die Seriennummer des Interfaces ändern, damit der Computer die Interfaces unterscheiden und ansprechen kann. Wenn du die Interfaces über mehrere serielle Schnittstellen ansprichst, ist das dagegen nicht nötig.

Um die Seriennummer eines Interfaces zu ändern, gehst du wie folgt vor:

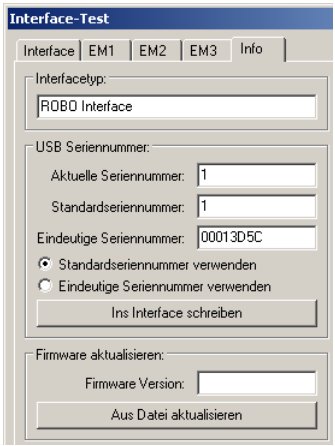
- Schließe das Interface ***einzeln*** an den USB Bus des Computers an.
- Drücke in der Werkzeugleiste auf den **COM/USB** Knopf und wähle die USB Schnittstelle aus.

COM/USB

- Öffne das Interfacetest Fenster über den **Test** Knopf in der Werkzeugleiste und wechsle zum **Info** Reiter:


Test

- Unter **Interfacetyp** wird die Art der Interfaces, also zum Beispiel **ROBO Interface** oder **ROBO I/O Extension** angezeigt.



- Unter **USB Seriennummer** kannst du die Seriennummer einstellen, die das Interface beim Starten verwendet. Jedes Interface hat zwei eingebaute Seriennummern, eine **Standardseriennummer**, die 1 ist solange du nichts anderes einstellst, und eine **eindeutige Seriennummer**, die du nicht verstellen kannst, und die bei jedem Interface anders ist. Der einfachste Weg um mehrere Interfaces am USB-Bus zu verwenden, besteht darin bei jedem Interface den Auswahlknopf auf **Eindeutige Seriennummer verwenden** umzustellen. Dann ist garantiert, dass jedes Interface eine eigene, unverwechselbare Seriennummer hat. Falls du viele Interfaces für ein Modell verwendest, kann es aber sehr unpraktisch sein, sich die ganzen Seriennummern zu merken. In diesem Fall ist es einfacher, wenn du die Standardseriennummern deiner Interfaces zum Beispiel auf 1, 2, 3 usw. einstellst und diese verwendest. Nachdem du die Seriennummer verstellt oder ausgewählt

hast, musst du noch den Knopf **Ins Interface schreiben** drücken. Nachdem du die Seriennummer geändert hast, musst du das Interface von der Stromversorgung trennen und wieder anschließen.

Achtung: Wenn die Seriennummer geändert wird, muss eventuell der Treiber neu installiert werden, wozu unter Windows NT, 2000 und XP Administratorrechte erforderlich sind. Wenn du die Seriennummer änderst, aber den Treiber nicht neu installieren kannst, weil du keine Administratorrechte hast, kannst du auf das Interface nicht mehr über USB zugreifen. In diesem Fall kannst du das Interface von der Stromversorgung trennen und beim Wiederanlegen der Stromversorgung den **Port-**Taster gedrückt halten. Das Interface startet dann mit der Seriennummer 1 und wird wieder von dem bereits installierten Treiber erkannt. Dabei wird die Seriennummer aber nicht dauerhaft umgestellt, d.h. beim nächsten Start ohne Port-Taster ist wieder die vorherige Seriennummer eingestellt. Um die Seriennummer dauerhaft umzustellen gehst du vor wie oben beschrieben.

- Unter **Firmware aktualisieren** kannst du schließlich das interne Steuerprogramm deines ROBO Interfaces aktualisieren, falls fischertechnik einmal eine neue Version der Interface-Firmware anbieten sollte.

7 Übersicht Programmelemente

Alle Programmelemente, die in ROBO Pro zur Verfügung stehen, sind im Folgenden nach Elementgruppen geordnet in der Reihenfolge beschrieben, in der sie im Elementfenster abgebildet sind.

7.1 Grundelemente (Level 1)

7.1.1 Start

Ein Prozess in einem Programm beginnt immer mit einem Startelement. Fehlt dieses Programmelement am Anfang, wird der Prozess nicht abgearbeitet. Wenn ein Programm mehrere Prozesse enthält, muss jeder dieser Prozesse mit einem Startbaustein beginnen. Die verschiedenen Prozesse werden dann gleichzeitig gestartet.

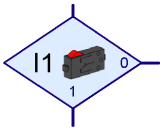
Ein Startelement hat keine Eigenschaften, die du verändern kannst. Deswegen wird im Gegensatz zu den meisten anderen Elementen ***kein*** Eigenschaftsfenster geöffnet, wenn du mit der rechten Maustaste auf das Element klickst.

7.1.2 Ende

Soll ein Prozess beendet werden, verbindet man den Ausgang des letzten Elements mit dem Ende-Element. Ein Prozess kann auch an verschiedenen Stellen mit diesem Element beendet werden. Es besteht auch die Möglichkeit Ausgänge von verschiedenen Elementen mit einem einzigen Endebaustein zu verbinden. Es kann aber auch durchaus vorkommen, dass ein Prozess als Endlosschleife ausgeführt wird und kein Ende-Element enthält.

Ein Ende-Element hat keine Eigenschaften, die du verändern kannst. Deswegen wird im Gegensatz zu den meisten anderen Elementen ***kein*** Eigenschaftsfenster geöffnet, wenn du mit der rechten Maustaste auf den Baustein klickst.

7.1.3 Verzweigung Digital

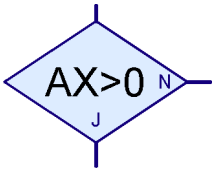


Mit dieser Verzweigung kannst du den Programmablauf abhängig vom Zustand eines der Digitaleingänge **I1** bis **I8** in zwei Richtungen lenken. Wenn z. B. ein Taster am Digitaleingang geschlossen (=1) ist, verzweigt das Programm zum **1**-Ausgang. Wenn der Eingang dagegen offen (=0) ist, verzweigt das Programm zum **0**-Ausgang.

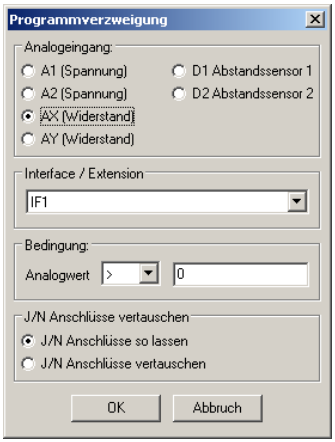
Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

- Mit den Knöpfen **I1** bis **I8** kannst du eingeben, welcher Eingang des Interface abgefragt werden soll.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interface oder einen Eingang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild für den am Eingang angeschlossenen Sensor auswählen. An Digitaleingänge werden meistens Taster angeschlossen, häufig aber auch Fototransistoren oder Reed-Kontakte.
- Unter **1/0 Anschlüsse vertauschen** kannst du die Position der 1- und 0-Ausgänge der Verzweigung vertauschen. Normalerweise ist der 1-Ausgang unten und der 0-Ausgang rechts. Oft ist es aber praktischer, wenn der 1 Ausgang rechts ist. Drücke auf **1/0 Anschlüsse vertauschen**, dann werden die beiden Anschlüsse vertauscht, sobald du das Fenster mit OK schließt.

7.1.4 Verzweigung Analog



Zusätzlich zu den Digitaleingängen hat das ROBO Interface 6 Analogeingänge, 2 Widerstandseingänge AX und AY, 2 Spannungseingänge A1 und A2 sowie 2 Eingänge für Abstandssensoren D1 und D2. Mit dieser Verzweigung kannst du den Wert eines Analogeingangs mit einer festen Zahl vergleichen und, je nachdem ob der Vergleich zutrifft oder nicht, zum Ja (J) oder Nein (N) Ausgang verzweigen.

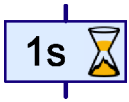


Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

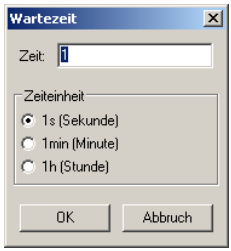
- Unter **Analogeingang** kannst du auswählen, welcher Eingang des Interface abgefragt werden soll. Alle Analogeingänge liefern einen Wert zwischen 0 und 1023. Weitere Informationen zu den verschiedenen Analogeingängen findest du im Abschnitt 7.6.2 *Analogeingang* auf Seite 75.

- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interface oder einen Eingang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bedingung** kannst du einen Vergleichsoperator wie kleiner (<) oder größer (>) auswählen und den Vergleichswert eingeben. Der Vergleichswert sollte im Bereich zwischen 0 und 1023 liegen. Wenn du ein Programm mit einer Verzweigung für Analogeingänge im Online-Modus startest, wird der aktuelle Analogwert angezeigt.
- Unter **J/N Anschlüsse vertauschen** kannst du die Position der 1- und 0-Ausgänge der Verzweigung vertauschen. Normalerweise ist der Ja (J) Ausgang unten und der Nein (N) Ausgang rechts. Oft ist es aber praktischer wenn der Ja-Ausgang rechts ist. Drücke auf **J/N Anschlüsse vertauschen**, dann werden die J- und N-Anschlüsse vertauscht, sobald du das Fenster mit OK schließt.

7.1.5 Wartezeit



Mit dem Element **Wartezeit** kannst du die weitere Ausführung eines Prozesses um eine einstellbare Zeit verzögern.

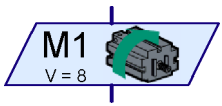


Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt. Hier kannst Du die Wartezeit in Sekunden, Minuten oder Stunden eingeben. Der Einstellbereich für die Wartezeit reicht von einer Millisekunde (das ist eine tausendstel Sekunde) bis 500 Stunden (das sind knapp 3 Wochen). Allerdings wird die Zeitmessung ungenauer, je länger die Wartezeit ist.

Die folgende Liste gibt zu verschiedenen Wartezeiten die Genauigkeit an:

Wartezeit	Genauigkeit
Bis 30 Sekunden	1/1000 Sekunde
Bis 5 Minuten	1/100 Sekunde
Bis 50 Minuten	1/10 Sekunde
Bis 8,3 Stunden	1 Sekunde
Bis 83 Stunden	10 Sekunden
Bis 500 Stunden	1 Minute

7.1.6 Motorausgang

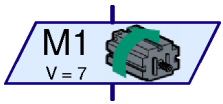


Mit dem Programmelement ***Motorausgang*** schaltet man einen der zweipoligen Ausgänge M1-M4 des Interface. Die Ausgänge des Interface können sowohl für Motoren als auch für Lampen oder Elektromagnete genutzt werden. Bei einem Motor möchte man außer der Geschwindigkeit auch die Drehrichtung einstellen können.

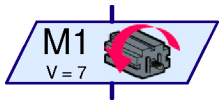
Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

- Unter **Motorausgang** kannst du einstellen welcher der vier Motorausgänge **M1** bis **M4** verwendet werden soll.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Ausgang des Interface oder einen Ausgang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild auswählen, das den am Ausgang angeschlossenen fischertechnik Baustein darstellt.
- Unter **Aktion** stellst du ein, wie der Ausgang beeinflusst werden soll. Einen Motor kannst du links oder rechts laufen lassen oder stoppen. Falls du eine Lampe an einen Motorausgang anschließt (siehe Tipp unter Lampenausgang) kannst du diese ein- oder ausschalten.
- Schließlich kannst du noch eine **Geschwindigkeit** oder **Intensität** zwischen 1 und 8 angeben. 8 ist die größte Geschwindigkeit, Helligkeit oder Magnetkraft, 1 die kleinste. Beim Stoppen oder Ausschalten brauchst du natürlich keine Geschwindigkeit angeben.

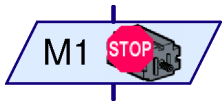
Hier die Symbole für einige Aktionen und Bilder aufgelistet:



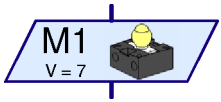
Motor rechts



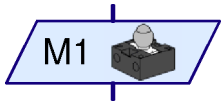
Motor links



Motor stopp



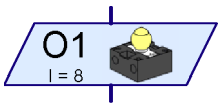
Lampe ein



Lampe aus

Tipp: Manchmal wird auch ein Motor nur in einer Richtung betrieben, z.B. bei einem Förderband. In diesem Fall kannst du für den Motor einen Lampenausgang verwenden, so dass ein Anschluss weniger verbraucht wird.

7.1.7 Lampenausgang (Level2)

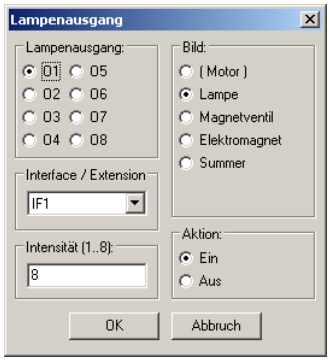


Mit dem Programmelement ***Lampenausgang*** schaltet man einen der einpoligen Ausgänge O1-O8 des Interface. Die Ausgänge des Interface können entweder paarweise als Motorausgänge (siehe oben) oder einzeln als Lampenausgänge O1-O8 genutzt werden. Die Lampenausgänge belegen im Gegensatz zum Motorausgang nur einen

Anschlusspin. Dadurch kannst du 8 Lampen oder Magnetventile getrennt ansteuern. Den anderen Anschluss der Lampe verbindest du mit der Massebuchse (⊥) des Interface.

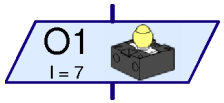
Tipp: Wenn du nur vier Lampen oder Motoren anschließen möchtest, kannst Du auch für Lampen einen Motorausgang verwenden. Das ist praktischer, weil du so beide Anschlüsse der Lampe am Interfaceausgang direkt anschließen kannst und nicht alle Minuspole mit der Massebuchse verbinden musst.

Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

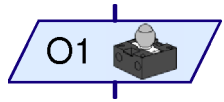


- Unter **Lampenausgang** kannst du einstellen welcher der acht Ausgänge **O1** bis **O8** verwendet werden soll.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Ausgang des Interface oder einen Ausgang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild auswählen, das den am Ausgang angeschlossenen fischertechnik Baustein darstellt.
- Unter **Aktion** stellst du ein, wie der Ausgang beeinflusst werden soll. Eine Lampe kannst du ein- oder ausschalten.
- Schließlich kannst du noch eine **Intensität** zwischen 1 und 8 angeben. 8 ist die größte Helligkeit oder Magnetkraft, 1 die kleinste. Beim ausschalten brauchst du natürlich keine Intensität angeben.

Hier sind die Symbole für die verschiedenen Aktionen für das Bild **Lampe** aufgelistet:

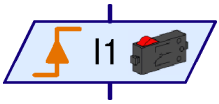


Lampe ein

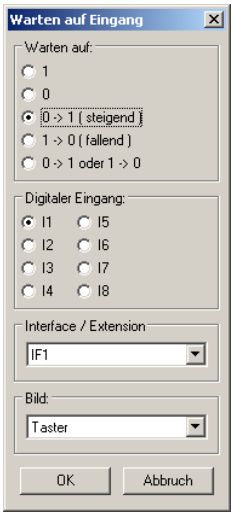


Lampe aus

7.1.8 Warten auf Eingang



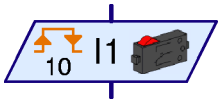
Das Element **Warten auf Eingang** wartet, bis ein Eingang des Interfaces einen bestimmten Zustand hat oder sich auf eine bestimmte Art ändert.



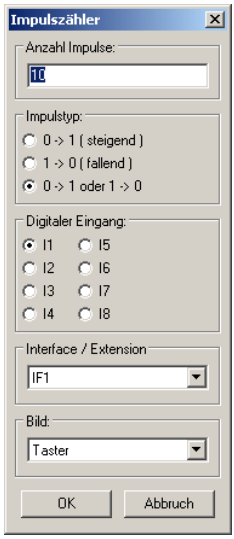
Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

- Unter **Warten auf** kannst du auswählen, auf welche Art von Änderung oder Zustand gewartet werden soll. Wenn du **1** oder **0** auswählst, wartet das Element so lange bis der Eingang geschlossen (1) oder offen (0) ist. Wenn Du **0 -> 1** oder **1 -> 0** auswählst wartet das Element, bis sich der Eingangszustand von offen in geschlossen (0->1) oder von geschlossen in offen (1->0) ***verändert***. Bei der letzten Möglichkeit wartet das Element, bis sich der Eingang verändert, egal ob von offen nach geschlossen oder umgekehrt. Im Abschnitt 3.6 *Weitere Programmelemente* auf Seite 22 ist zum weiteren Verständnis erklärt, wie du dieses Element mit dem Element Verzweigung nachbauen kannst.
- Unter **Digitaler Eingang** kannst du einstellen, welcher der 8 digitalen Eingänge von **I1** bis **I8** abgefragt werden soll.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interface oder einen Eingang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild für den am Eingang angeschlossenen Sensor auswählen. An Digitaleingänge werden meistens Taster angeschlossen, häufig aber auch Fototransistoren oder Reed-Kontakte.

7.1.9 Impulszähler



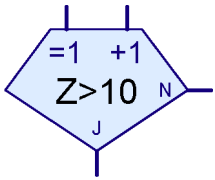
Viele Fischertechnik Robotermodelle verwenden auch so genannte Impulszahnräder. Diese Zahnräder betätigen einen Taster bei jeder Umdrehung 4 Mal. Mit solchen Impulszahnrädern kann man einen Motor statt einer bestimmten Zeit eine genau definierte Zahl von Umdrehungen einschalten. Dazu muss man die Zahl der Impulse an einem Eingang des Interface zählen. Zu diesem Zweck gibt es das Element ***Impulszähler***, das auf eine einstellbare Zahl von Impulsen wartet.



Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

- Unter **Impulstyp** kannst du auswählen, auf welche Art von Impuls gezählt werden soll. Wenn du **0 -> 1** (steigend) auswählst, wartet das Element, bis sich der Eingangszustand so oft von offen in geschlossen (0->1) verändert hat. wie du unter **Anzahl Impulse** angegeben hast. Bei **1 -> 0** (fallend) wartet das Element, bis sich der Eingangszustand so oft von geschlossen in offen verändert hat, wie angegeben. Mit Impulszahnrädern wird aber häufiger die dritte Möglichkeit verwendet. Hier zählt das Element sowohl die Änderungen **0 -> 1** als auch die Änderungen **1 -> 0**, so dass pro Umdrehung eines Impulszahnrades 8 Impulse gezählt werden.
- Unter **Digitaler Eingang** kannst du einstellen welcher der 8 digitalen Eingänge von **I1** bis **I8** abgefragt werden soll.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interface oder einen Eingang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild für den am Eingang angeschlossenen Sensor auswählen. Für Digitaleingänge werden meistens Taster verwendet, häufig aber auch Fototransistoren oder Reed-Kontakte.

7.1.10 Zählschleife



Mit dem Element **Zählschleife** kannst du ganz einfach ein bestimmtes Programmstück mehrfach ausführen lassen. Das Zählschleifenelement hat einen eingebauten Zähler. Wenn die Zählschleife über den **=1** Eingang betreten wird, wird der Zähler auf 1 gesetzt. Wenn die Zählschleife dagegen über den **+1** Eingang betreten wird, wird zum Zähler 1 dazu gezählt. Je nach dem ob der Zähler größer einem von dir vorgegeben Wert ist oder nicht, verzweigt die Zählschleife zum Ja (**J**) oder zum Nein (**N**) Ausgang. Ein Beispiel dazu findest du im Abschnitt 3.6.4 *Zählschleife* auf Seite 24.

Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt:

- Unter **Anzahl der Schleifendurchläufe** gibst du ein, wie oft die Zählschleife über Nein (**N**) Ausgang verlassen werden soll, bevor der Ja (**J**) Ausgang aktiviert wird. Der eingegebene Wert sollte positiv sein.
- Wenn du **J/N Anschlüsse vertauschen** anklickst, werden der **J**- und der **N-**Anschluss vertauscht, sobald du das Fenster mit OK schließt. Je nachdem wo der **J**- und wo der **N**-Anschluss sind, steht der Programmteil, der wiederholt wird, rechts oder unter der Zählschleife.

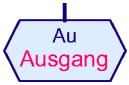
7.2 Unterprogramm I/O (Level2-3)

In dieser Elementgruppe findest du Programmelemente, die du nur in Unterprogrammen benötigst.

7.2.1 Unterprogramm-Eingang (Level 2)

Ein Unterprogramm kann einen oder mehrere Unterprogramm-Eingänge haben. Über diese Eingänge übergibt das Hauptprogramm oder übergeordnete Unterprogramm die Ausführung an das Unterprogramm. Im grünen Symbol des Unterprogramms, das in das übergeordnete Programm eingefügt wird, wird für jeden Unterprogramm-Eingang an der Oberseite ein Anschlusspin eingefügt. Die Anschlüsse im Symbol haben die gleiche Reihenfolge (von links nach rechts) wie die Unterprogramm-Eingänge im Funktionsplan des Unterprogramms. Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt. Dort kannst du dem Eingang einen Namen geben, der dann im Symbol angezeigt wird. Mehr zum Thema Unterprogramme findest du im Kapitel 4: *Level 2: Arbeiten mit Unterprogrammen* auf Seite 28.

7.2.2 Unterprogramm-Ausgang (Level 2)



Ein Unterprogramm kann einen oder mehrere Unterprogramm-Ausgänge haben. Über diese Ausgänge übergibt das Unterprogramm die Programmausführung zurück an das Hauptprogramm oder übergeordnete Unterprogramm. Im grünen Symbol des Unterprogramms, das in das übergeordnete Programm eingefügt wird, wird für jeden Unterprogrammausgang an der Unterseite ein Anschlusspin eingefügt. Die Anschlüsse im Symbol haben die gleiche Reihenfolge (von links nach rechts) wie die Unterprogramm-Ausgänge im Ablaufplan des Unterprogramms. Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt. Dort kannst du dem Ausgang einen Namen geben, der dann im Element angezeigt wird. Mehr zum Thema Unterprogramme findest du im Kapitel 4: *Level 2: Arbeiten mit Unterprogrammen* auf Seite 28.

7.2.3 Unterprogramm-Befehlseingang (Level 3)

Über dieses Element können Unterprogramme mit Eingangselementen wie z.B. Schaltern im Hauptprogramm oder im übergeordneten Unterprogramm verbunden werden oder von dort mit Werten aus Variablenelementen, z.B. Koordinaten, versorgt werden. Im grünen Symbol des Unterprogramms, das in das übergeordnete Programm eingefügt wird, wird für jeden Unterprogramm-Befehlseingang an der linken Seite ein Anschlusspin eingefügt. Die Anschlüsse im Symbol haben die gleiche Reihenfolge (von oben nach unten) wie die Unterprogramm-Befehlseingänge im Unterprogramm. Wenn

du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt. Dort kannst du dem Ausgang einen Namen geben, der dann im Element angezeigt wird. Die Anwendung dieses Elements wird ausführlich in Abschnitt 5.5 *Befehlseingänge für Unterprogramme* auf Seite 43 erklärt.

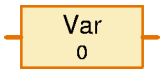
7.2.4 Unterprogramm-Befehlsausgang (Level 3)

Über dieses Programmelement können Befehle wie z. B. links, rechts, stopp, an Motoren oder andere Ausgangselemente gesendet werden, die sich im Hauptprogramm oder im übergeordneten Unterprogramm befinden. Im grünen Symbol des Unterprogramms, das in das übergeordnete Programm eingefügt wird, wird für jeden Unterprogramm-Befehlsausgang an der rechten Seite ein Anschlusspin eingefügt. Die Anschlüsse im Symbol haben die gleiche Reihenfolge (von oben nach unten) wie die Unterprogramm-Befehlsausgänge im Unterprogramm. Wenn du mit der rechten Maustaste auf das Element klickst, wird das Eigenschaftsfenster angezeigt. Dort kannst du dem Ausgang einen Namen geben, der dann im Element angezeigt wird. Die Anwendung dieses Elements wird ausführlich in Abschnitt 5.5 *Befehlseingänge für Unterprogramme* auf Seite 43 erklärt.

7.3 Variable, Liste, ... (Level 3)

Die Programmelemente in dieser Gruppe können einen oder mehrere Zahlenwerte speichern. Damit kann man Programme entwickeln, die ein Gedächtnis haben.

7.3.1 Variable (global)



Eine Variable kann einen einzelnen Zahlenwert zwischen -32767 und 32767 speichern. Der Wert der Variablen wird gesetzt, indem man am Befehlseingang auf der linken Seite ein **=** Befehlselement anschließt (siehe Abschnitt 7.4.1 = *(Zuweisen)* auf Seite 69). Man kann einer Variablen über das Eigenschaftsfenster auch einen Anfangswert geben, den die Variable behält bis sie den ersten Befehl zur Änderung des Wertes erhält.

ROBO Pro legt für alle Variablenelemente mit ***gleichem Namen*** und **Variablentyp** = **Global** nur eine einzige Variable an. Alle globalen Variablen mit gleichem Namen sind identisch und haben immer den gleichen Wert, auch wenn sie in verschiedenen Unterprogrammen vorkommen. Wenn eines dieser Variablenelemente über einen Befehl verändert wird, ändern sich auch alle anderen globalen Variablenelemente mit gleichem Namen. Es gibt auch lokale Variablen (siehe nächster Abschnitt) bei denen das nicht so ist.

Zusätzlich zum **=** Befehl versteht eine Variable auch den **+** und den **–** Befehl. Erhält eine Variable z.B. den Befehl **+ 5**, addiert sie den Wert 5 zu ihrem aktuellen Wert hinzu. Beim **–** Befehl wird der mit dem Befehl übermittelte Wert vom aktuellen Wert der Variablen abgezogen.

Achtung:
Wenn der Wert der Variablen bei einem **+** oder **–** Befehl über den Wertebereich der Variablen hinausgeht, wird zum Wert der Variablen 65536 hinzugezählt oder davon abgezogen, so dass der Wert wieder gültig ist. Da dieses Verhalten in der Regel nicht erwünscht ist, solltest Du darauf achten, dass das nicht passiert.

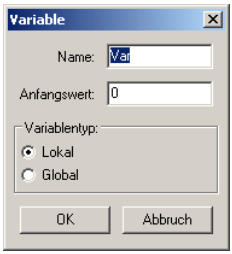
Jedes Mal, wenn sich der Wert der Variablen ändert, schickt sie einen **=** Befehl mit dem neuen Wert an alle Elemente, die am Befehlsausgang der Variablen angeschlossen sind. Wenn Du den Wert einer Variablen beobachten willst, kannst Du an den Ausgang der Variablen eine Bedienfeldanzeige anschließen (siehe Abschnitt 7.6.6 *Bedienfeldanzeige* auf Seite 79).

Hier noch einmal zur Übersicht alle Befehle, die das Variablenelement verarbeiten kann:

Befehl	Wert	Aktion
=	-32767 bis 32767	Setzt den Wert der Variablen auf den mit dem Befehl übermittelten Wert.
+	-32767 bis 32767	Addiert zum aktuellen Wert der Variablen den mit dem Befehl übermittelten Wert.
-	-32767 bis 32767	Subtrahiert vom aktuellen Wert der Variablen den mit dem Befehl übermittelten Wert.

Der krumme Wertebereich von -32767 bis 32767 kommt übrigens daher, dass Computer im Zweiersystem rechnen, und nicht im Zehnersystem wie wir. Im Zweiersystem ist 32767 eine glatte Zahl, etwa wie 9999 im Zehnersystem. Aber darum brauchen wir uns nicht zu kümmern, da der Computer alle Zahlen vom Zweier- ins Zehnersystem umrechnet. Nur bei den Maximalwerten von Variablen merkt man noch etwas davon und wenn es beim Rechnen einen Überlauf gibt.

Eigenschaftsfenster für Variablen



- Unter **Name** kannst du einen Namen für die Variable eingeben.
- Unter **Anfangswert** kannst du den Anfangswert für die Variable eingeben. Die Variable behält diesen Wert solange, bis sie über einen =, + oder – Befehl einen neuen Wert bekommt.
- Der Punkt **Variablentyp** ist nur für Variablen in Unterprogrammen interessant und wird im folgenden Abschnitt „Lokale Variable“ genauer erklärt. Bei Variablen im Hauptprogramm haben beide Einstellungen den gleichen Effekt.

7.3.2 Lokale Variable

Alle **globalen** Variablenelemente mit gleichem Namen verwenden ein und dieselbe Variable und haben immer den gleichen Wert. Das ist vermutlich das was du erwartest und was in der Regel auch praktisch ist. Wenn du aber Variablen in Unterprogrammen verwendest, kann das zu großen Problemen führen. Wenn dein Programm mehrere parallele Prozesse hat, kann ein Unterprogramm zu einer Zeit mehrfach ausgeführt werden. In so einer Situation führt es in der Regel zu Chaos, wenn das Unterprogramm in allen Prozessen die gleichen Variablen verwendet. Aus diesem Grund gibt es ***lokale Variablen***. Eine lokale Variable verhält sich fast genauso wie eine globale Variable, mit einem Unterschied: die lokale Variable gilt nur in dem Unterprogramm, in dem sie definiert ist. Selbst wenn zwei lokale Variablen in verschiedenen Unterprogrammen den gleichen Namen haben, sind es verschiedene, unabhängige Variablen. Auch wenn ein Unterprogramm von mehreren Prozessen mehrfach parallel ausgeführt wird, hat das Unterprogramm in jedem Prozess einen unabhängigen Satz von lokalen Variablen. Lokale Variablen existieren nur so lange, wie das Unterprogramm in dem sie definiert sind, ausgeführt wird. Der Anfangswert wird lokalen Variablen daher nicht beim Programmstart, sondern bei jedem Start des jeweiligen Unterprogramms zugewiesen. Da ein Unterprogramm bei mehreren Aufrufen in der Regel immer wieder das gleiche machen soll, ist es viel praktischer, wenn die Variablen bei jedem Aufruf auf den Anfangswert gesetzt werden. Lokale Variablen haben sozusagen keine Erinnerung an vorherige Aufrufe des gleichen Unterprogramms.

Im Hauptprogramm verhalten sich lokale Variablen und globale Variablen gleich, da das Gesamtprogramm und das Hauptprogramm gleichzeitig gestartet werden. Lokale Variablen sind aber etwas effizienter in der Programmausführung. Listenelemente sollte man dagegen eher global definieren, weil der Speicherbereich für globale Variablen größer ist als für lokale Variablen.

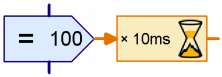
7.3.3 Konstante

Eine Konstante hat wie eine Variable einen Wert, aber der Wert kann nicht durch das Programm verändert werden. Eine Konstante kannst du mit einem Dateneingang eines Unterprogrammsymbols verbinden, wenn das Unterprogramm immer den gleichen Wert verwenden soll. Auch bei Berechnungen mit Operatoren sind Konstanten sehr praktisch. Ein Beispiel dafür findest du am Ende des Abschnitts 5.7 *Operatoren* auf Seite 47.

7.3.4 Timer-Variable

Eine Timervariable verhält sich im Wesentlichen genau so wie eine Variable. Auch die Unterscheidung zwischen normalen und statischen Variablen gibt es bei Timer-Variablen. Der einzige Unterschied ist, dass eine Timervariable den gespeichert Wert in einem festen Zeittakt bis auf 0 herunterzählt. Sobald der Timerwert 0 ist, bleibt die Timer-Variable stehen. Wenn der Timerwert z.B. durch einen Minus-Befehl negativ wird, wird der Wert beim nächsten Zeitschritt wieder 0.

Die Geschwindigkeit mit der die Timer-Variable herunterzählt kannst du zwischen 1/1000 Sekunde pro Schritt und 1 Minute pro Schritt im Eigenschaftsfenster einstellen. Du solltest dabei beachten, dass die Genauigkeit des Timers von den eingestellten Zeitschritten abhängt. Wenn du zum Beispiel einen Timer auf 1 x 10 Sekunden setzt, kann der nächste 10 Sekunden Timerschritt kurze Zeit später , (z.B. bereits nach einer Sekunde) oder erst nach 10 Sekunden erfolgen. Timer sind also immer nur so genau, wie die eingestellten Zeitschritte. Daher solltest du eher kleine Zeitschritte wählen, zum Beispiel 10 x 1 Sekunde oder 100 x 0,1 Sekunden statt 1 x 10 Sekunden. Einen Zeitschritt von einer Minute solltest du nur wählen, wenn das Programm mindestens eine Stunde warten soll. Dann wird es auf eine Minute mehr oder weniger nicht ankommen.

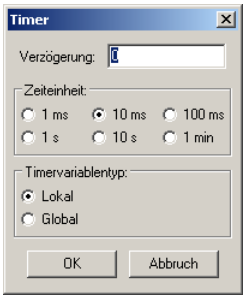


Die Anzahl der Schritte, die heruntergezählt werden soll, wird der Timer-Variablen in der Regel über einen **=**-Befehl von einem Befehlselement zugewiesen. Im abgebildeten Beispiel werden 100 Schritte von je 10ms heruntergezählt. Das entspricht einer Zeitdauer von 1000 ms=1 Sekunde. Die Genauigkeit beträgt dabei 10ms.

Mit Timer-Variablen kann man sehr einfach auch schwierige Zeitmessungs- und Verzögerungsaufgaben lösen. Wenn ein Roboter zum Beispiel eine Suche nach 20 Sekunden abbrechen soll, kannst du am Anfang der Suche eine Timervariable auf 20 x 1 Sekunde (oder 200 x 0,1 Sekunde) setzen und dann im Suchprogramm regelmäßig abfragen, ob der Timer-Wert noch größer 0 ist. Du kannst auch bei Teilerfolgen der Suche den Timer wieder auf den Anfangswert setzen.

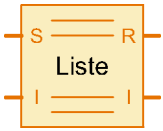
Wenn du eine Zeit messen willst, muss du die Timer-Variable am Anfang auf einen möglichst großen positiven Wert (30000 oder 32767) setzen, damit viel Zeit verbleibt, bis der Timerwert 0 ist. Wenn du wissen willst, wie viel Zeit seitdem vergangen ist, ziehst du den aktuellen Timerwert vom Anfangswert ab.

Eigenschaftsfenster für Timer-Variablen



- Unter **Verzögerung** kannst du den Anfangswert für die Timervariable festlegen. In der Regel gibt man hier 0 ein und setzt den Wert der Timer-Variable mit einem **=** Befehl zum passenden Zeitpunkt. Wenn der Timer aber bei Start des Programms oder eines Unterprogramms loslaufen soll, kann hier der entsprechende Wert eingegeben werden.
- Unter **Zeiteinheit** kannst du die Größe der Zeitschritte auswählen, in denen die Timer-Variable heruntergezählt wird.
- Unter **Timervariablentyp** kannst du einstellen, ob der Timer eine globale oder eine lokale Variable ist (siehe Abschnitt 7.3.2 *Lokale Variable* auf Seite 65).

7.3.5 Liste



Das Element **Liste** entspricht einer Variablen, mit der man nicht nur einen Wert, sondern mehrere Werte speichern kann. Die maximale Zahl von Werten, die in einer Liste gespeichert werden können, wird im Eigenschaftsfenster des Elements festgelegt.

Du kannst an die Liste hinten Werte anhängen und Werte am Ende der Liste entfernen. Zudem kannst du einen beliebigen Wert in der Liste verändern oder auslesen und einen beliebigen Wert in der Liste mit dem ersten Wert in der Liste vertauschen. Einen Wert in die Mitte oder am Anfang der Liste einfügen oder löschen kann man nicht direkt. Man kann aber ein entsprechendes Unterprogramm schreiben, das diese Funktionen ausführt.

Folgende Funktionen einer Liste werden genutzt, indem man dem Element an den Eingang **S** (für Schreiben) Befehle schickt. Dem **S**-Eingang kann man folgende Befehle schicken:

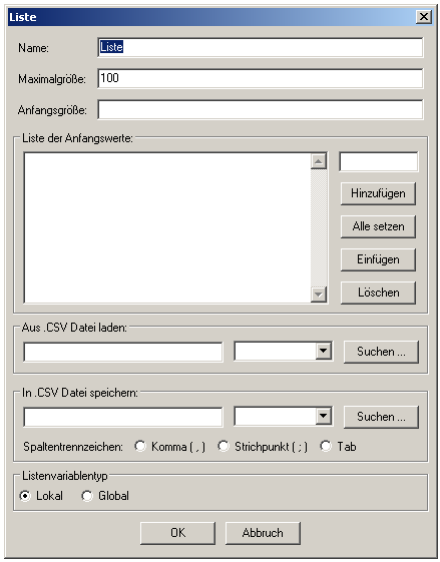
Befehl	Wert	Aktion
Anhängen 	-32767 bis 32767	Hängt den mit dem Befehl übermittelten Wert am Ende der Liste an. Die Liste wird um ein Element vergrößert. Wenn die Liste schon die maximale Größe hat, wird der Befehl ignoriert.
Entfernen 	0 bis 32767	Löscht die angegebene Zahl von Elementen am Ende der Liste. Der mit dem Befehl übermittelte Wert ist die Anzahl der zu löschenden Elemente. Wenn die Anzahl größer ist als die Zahl der Elemente in der Liste, werden alle Elemente in der Liste gelöscht. Ist die Anzahl 0 oder negativ, wird der Befehl ignoriert.
Vertauschen 	0 bis 32767	Vertauscht das angegebene Element mit dem ersten Element in der Liste. Der mit dem Befehl übermittelte Wert ist die Nummer des Elements, das vertauscht werden soll.

Über den Eingang **I** (für Index) kann ein bestimmtes Element der Liste ausgewählt werden. Dazu schickt man dem **I**-Eingang einen **=**-Befehl mit der gewünschten Elementnummer. Das erste Element hat dabei die Nummer 0. Dem Element, das über den **I**-Eingang ausgewählt wurde, kann man einen neuen Wert zuweisen, indem man an den **S**-Eingang mit einem **=**-Befehl den gewünschten neuen Wert schickt.

Das über den **I**-Eingang ausgewählte Element kann über den **R**-Ausgang (für Rücklesen) abgefragt werden. Ändert sich der **I**-Eingang oder der Wert des Eintrags, der über den **I**-Eingang ausgewählt wurde, schickt die Liste den aktuellen Wert des ausgewählten Eintrags an die Elemente, die am **R**-Ausgang angeschlossen sind.

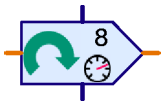
Über den **I**-Ausgang lässt sich abfragen, ob der am **I**-Eingang angelegte Index gültig ist. Wenn N die Anzahl der Elemente ist, muss am **I**-Eingang ein Wert zwischen 0 und N-1 angelegt werden. Ist das der Fall, sendet der **I**-Ausgang einen **=**-Befehl mit Wert N, anderenfalls mit Wert 0 an alle angeschlossenen Elemente.

Eigenschaftsfenster für Listen



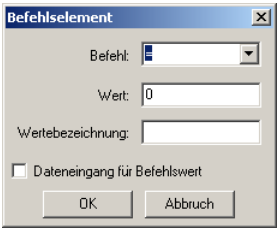
- Unter **Maximalgröße** kannst du die maximale Zahl von Elementen der Liste eingeben. Diese Größe kann von **Anhängen**-Befehlen nicht überschritten werden.
- Unter **Anfangsgröße** gibst du die Anzahl der Elemente ein, mit der die Liste zum Start vorbelegt werden soll.
- Unter **Liste der Anfangswerte** kannst du die Anfangswerte eingeben, mit denen die Liste vorbelegt werden soll. Mit den Knöpfen rechts neben der Liste kannst du die Liste bearbeiten.
- Unter **Aus .CSV Datei laden** kannst du eine Excel-kompatible .CSV Datei auswählen, aus der die Liste ihre Werte übernimmt. Im Auswahlfeld in der Mitte kannst du die Spalte der .CSV Datei auswählen, die dazu verwendet werden soll. Die Datei wird sofort geladen und unter **Liste der Anfangswerte** angezeigt. Wenn du das Programm startest oder einen Download durchführst, versucht ROBO Pro nochmals die aktuellen Werte aus der Datei zu laden. Gelingt das nicht, werden die unter Liste der Anfangswerte gespeicherten Werte verwendet.
- Unter **In .CSV Datei speichern** kannst du eine Datei angeben, in die der Inhalt der Liste nach Programmende gespeichert werden soll. Das funktioniert aber nur im Online- oder Online-Modus und nur für statische Listen (siehe nächster Punkt). Der Inhalt der Liste wird in die ausgewählte Spalte der Datei geschrieben. Unter **Spaltentrennzeichen** kannst du auswählen, ob die einzelnen Spalten in der Liste mit Kommata oder Strichpunkten getrennt werden sollen. In Ländern in denen man 0.5 mit Punkt schreibt wird in der Regel ein Komma als Spaltentrennzeichen verwendet. Da man in Deutschland 0,5 mit Komma schreibt, wird in Deutschland auch häufig ein Strichpunkt als Spaltentrennzeichen verwendet. Wenn du beim Import einer ROBO Pro CSV Datei zum Beispiel in Microsoft Excel Probleme hast, versuche ein anderes Spaltentrennzeichen.
- Unter **Listenvariablentyp** kannst du einstellen, ob die Liste Timer eine globale oder eine lokale Variable ist (siehe Abschnitt 7.3.2 *Lokale Variable* auf Seite 65). Für große Listen (Maximalgröße mehr als 100 Elemente) ist der Typ **Global** empfehlendwert, weil für globale Variablen mehr Speicher zur Verfügung steht, als für lokale Variablen.

7.4 Befehle (Level 3)



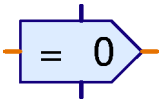
Alle Programmelemente in dieser Gruppe sind Befehlselemente. Je nach Anwendung werden sie auch als Nachrichtenelemente bezeichnet. Wenn ein Befehlselement ausgeführt wird (d.h. wenn der Programmfluss oben in den blauen Eingang des Elements hineingeht), schickt das Befehlselement einen Befehl oder eine Nachricht an das Element, das rechts an seinem Ausgang angeschlossen ist. Es gibt verschiedene Befehle wie rechts, links oder stopp, die unterschiedliche Wirkungen auf das angeschlossene Element haben. In der Regel verstehen die angeschlossenen Elemente nur wenige Befehle. Bei den verschiedenen Programmelementen ist aufgelistet, welche Befehle sie verstehen und welche Wirkung die Befehle haben. Die meisten Befehle werden noch von einem Wert begleitet. Bei einem **Rechts**-Befehl gibt man zum Beispiel noch eine Geschwindigkeit zwischen 1 und 8 an. Ein **Stopp**-Befehl hat dagegen keinen zusätzlichen Wert.

Eigenschaftsfenster für Befehlselemente



- Unter **Befehl** kannst du aus einer Liste aller möglichen Befehle den gewünschten Befehl auswählen.
- Unter **Wert** gibst du den Zahlenwert ein, der mit dem Befehl übermittelt werden soll. Falls kein Wert übermittelt werden soll, bleibt dieses Feld leer.
- Unter **Wertebezeichnung** kannst du einen kurzen Hinweistext eingeben, der zusammen mit dem Wert im Befehlselement angezeigt wird (z.B. X= oder T=). Der Hinweis soll verdeutlichen um was für eine Art von Wert es sich handelt. Dies dient aber nur als Kommentar und hat keinerlei Funktion.
- Unter **Dateneingang für Befehlswert** kannst du angeben, ob das Befehlselement an seiner linken Seite einen orangen Dateneingang für den zu übermittelnden Wert haben soll. Der Wert kann bei allen Befehlselementen entweder in dem Befehlselement direkt eingegeben werden oder über einen Dateneingang auf der linken Seite des Befehlselements eingelesen werden. Dadurch kann man zum Beispiel einen Motor in einem Regelkreis mit einer veränderlichen Geschwindigkeit ansteuern.

7.4.1 = (Zuweisen)

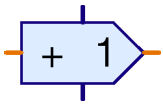


Der **=**-Befehl weist dem Empfänger einen Wert zu. In der Regel wird er verwendet um Variablen, Timer-Variablen, Listenelementen oder Bedienfeld-Ausgängen einen Wert zuzuweisen.

Der **=**-Befehl wird aber nicht nur von Befehlselementen gesendet, sondern von allen Programmelementen mit Datenausgängen. Alle Elemente versenden **=**-Befehle, wenn sich der Wert eines Ausgangs ändert. Ein Digitaleingangselement versendet beispielsweise einen = 1-Befehl wenn ein Taster am Eingang geschlossen wird und einen = 0 Befehl, wenn der Taster geöffnet wird. Dabei wird aber kein Befehlselement verwendet. In Programmelementen mit Datenausgängen sind sozusagen **=**-Befehlselemente eingebaut.

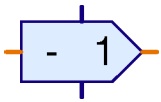
Alle Dateneingänge von ROBO Pro-Programmelementen können zumindest den **=**-Befehl verarbeiten. Der **=**-Befehl ist damit der am häufigsten verwendete Befehl in ROBO Pro.

7.4.2 + (Plus)



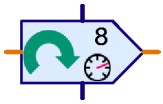
Der **+**-Befehl wird an Variablen oder Timer-Variablen geschickt, um den Wert der Variablen zu erhöhen. Mit dem **+**-Befehl kann ein beliebiger Wert übermittelt werden, der zu der Variablen hinzu addiert wird. Da der mit dem Befehl übermittelte Wert auch negativ sein kann, kann sich der Wert der Variable dadurch auch vermindern. Siehe Abschnitt 7.3.1 *Variable* auf Seite 64 und Abschnitt 7.3.4 *Timer-Variable* auf Seite 66.

7.4.3 – (Minus)



Der **–** Befehl wird ähnlich verwendet wie der zuvor beschriebene **+** Befehl. Der einzige Unterschied ist, dass der mit dem Befehl übermittelte Wert vom Wert der Variable abgezogen wird.

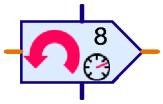
7.4.4 Rechts



Der Befehl **Rechts** wird an Motorausgangselemente geschickt um den Motor mit Drehrichtung rechts einzuschalten. Siehe Abschnitt 7.6.4 *Motorausgang* auf Seite 77.

Der Wert ist eine Geschwindigkeit von 1 bis 8.

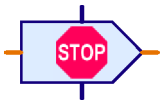
7.4.5 Links



Der Befehl **Links** wird an Motorausgangselemente geschickt um den Motor mit Drehrichtung links einzuschalten. Siehe Abschnitt 7.6.4 *Motorausgang* auf Seite 77.

Der Wert ist eine Geschwindigkeit von 1 bis 8.

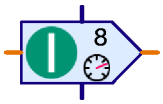
7.4.6 Stopp



Der Befehl **Stopp** wird an Motorausgangselemente geschickt um den Motor anzuhalten. Siehe Abschnitt 7.6.4 *Motorausgang* auf Seite 77.

Mit dem **Stopp** Befehl wird kein Wert übermittelt.

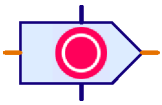
7.4.7 Ein



Der Befehl **Ein** wird an Lampenausgangselemente geschickt um die Lampe einzuschalten. Siehe Abschnitt 7.6.5 *Lampenausgang* auf Seite 78. Ein **Ein**-Befehl kann auch an Motorausgangselemente geschickt werden, er entspricht dem Befehl **Rechts.** Für Motoren sollte aber besser der Befehl **Rechts** verwendet werden, weil die Drehrichtung direkt erkennbar ist.

Der Wert ist die Helligkeit oder Intensität von 1 bis 8.

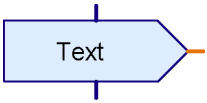
7.4.8 Aus



Der Befehl **Aus** wird an Lampenausgangselemente geschickt um die Lampe auszuschalten. Siehe Abschnitt 7.6.5 *Lampenausgang* auf Seite 78. Ein **Aus** Befehl kann auch an Motorausgangselemente geschickt werden, er entspricht dem Befehl **Stopp**.

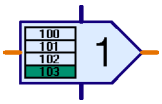
Mit dem **Aus** Befehl wird kein Wert übermittelt.

7.4.9 Text



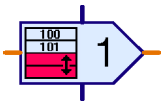
Der Befehl **Text** ist ein besonderer Befehl, da er nicht einen Befehl mit einer Zahl, sondern einen beliebigen Text an das angeschlossene Element schickt. Allerdings gibt es nur ein Programmelement, das den Text Befehl verarbeiten kann, und zwar eine Textanzeige in einem Bedienfeld. Weitere Informationen findest du im Abschnitt 8.1.2 *Textanzeige* auf Seite 83.

7.4.10 Wert Anhängen



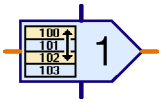
Der Befehl **Anhängen** ist ein Spezialbefehl für Listenelemente. Siehe Abschnitt 7.3.5 *Liste* auf Seite 67 . Mit dem Befehl wird ein Wert übermittelt, der an das Ende der Liste angehängt wird. Wenn die Liste bereits voll ist, wird der Befehl ignoriert.

7.4.11 Wert(e) entfernen



Der Befehl **Entfernen** ist ein Spezialbefehl für Listenelemente. Siehe Abschnitt 7.3.5 *Liste* auf Seite 67. Mit dem Befehl lässt sich eine beliebige Anzahl von Elementen am Ende einer Liste löschen. Die gewünschte Anzahl wird mit dem Befehl als Wert übermittelt. Ist der übermittelte Wert größer als die Anzahl der Elemente in der Liste, werden alle Elemente in der Liste gelöscht. Um eine Liste vollständig zu löschen kann man einen Befehl **Entfernen** mit dem maximal möglichen Wert von 32767 an ein Listenelement senden.

7.4.12 Werte vertauschen

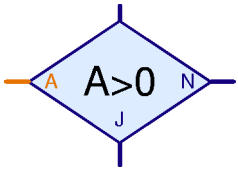


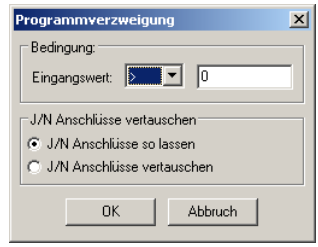
Der Befehl **Vertauschen** ist ein Spezialbefehl für Listenelemente. Siehe Abschnitt 7.3.5 *Liste* auf Seite 67. Mit dem Befehl lässt sich ein beliebiges Element einer Liste mit dem ersten Element der Liste vertauschen. Die Nummer des Elements, mit dem das erste Element vertauscht wird, wird mit dem Befehl als Wert übermittelt. ***Wichtig:*** das erste Element einer Liste hat die Nummer 0. Wenn der übermittelte Wert keine gültige Elementnummer ist, wird der Befehl vom Listenelement ignoriert.

7.5 Vergleiche, Warten auf, … (Level3)

Die Programmelemente in dieser Gruppe dienen alle zur Programmverzweigung oder zur Verzögerung des Programmablaufs.

7.5.1 Verzweigung (mit Dateneingang)





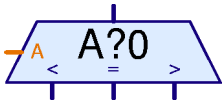
Diese Programmverzweigung besitzt einen orangen Dateneingang **A** links am Element. Darüber wird ein Wert eingelesen, der häufig von einem Eingangselement (siehe Abschnitt 7.6.1 bis 7.6.6 ab Seite 74) kommt. Der Dateneingang **A** kann aber auch mit den Datenausgängen von Variablen, Timer-Variablen oder Operatoren (siehe Abschnitt 7.7 *Operatoren* auf Seite 80) verbunden werden. Der Wert am Dateneingang **A** wird von dem Element mit einem festen, aber frei definierbaren Wert verglichen. Je nachdem ob der Vergleich zutrifft oder nicht, verzweigt das Element zum **J**- oder zum **N**-Ausgang.

Eigenschaftsfenster für die Verzweigung

- Unter **Bedingung** trägst du im rechten Feld den Wert ein, mit dem der Eingangswert **A** verglichen werden soll. Für den Vergleich stehen alle üblichen Vergleichsoperationen zur Verfügung.
- Wenn du J/N Anschlüsse vertauschen auswählst, werden der **J**- und **N**-Ausgang miteinander vertauscht, sobald du das Eigenschaftsfenster mit OK schließt. Um die J/N Anschlüsse wieder an ihre Ausgangsposition zu bringen, kannst du sie nochmals vertauschen.

Der am meisten verwendete Vergleich ist **A>0**. Das bedeutet, dass der Programmfluss zum **J**-Ausgang verzweigt, wenn der am Dateneingang **A** angelegte Wert größer als 0 ist. Damit lassen sich zum Beispiel Digitaleingänge auswerten, die eine 1 oder 0 liefern. Aber auch Timer-Variablen und viele andere Werte lassen sich mit dem Vergleich **A>0** sinnvoll auswerten.

7.5.2 Vergleich mit Festwert

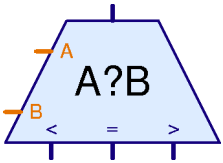


Mit dem Programmelement **Vergleich mit Festwert** lässt sich der Wert am Dateneingang **A** mit einem festen, aber frei definierbaren Wert vergleichen. Je nachdem ob der am Dateneingang **A** angelegte Wert größer, kleiner oder gleich dem Festwert ist, verzweigt dieses Vergleichselement zum rechten, linken oder mittleren Ausgang. In der Regel wird am Dateneingang **A** der Ausgang einer Variable oder einer Liste angeschlossen. Das Vergleichselement lässt sich durch zwei Verzweigungselemente ersetzen. Es ist jedoch in vielen Fällen übersichtlicher, wenn man nur ein Element benötigt.

Eigenschaftsfenster für Vergleich

- Unter Vergleichswert kannst du den konstanten Wert eingeben, mit dem der Wert am Eingang **A** verglichen werden soll.

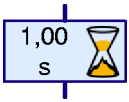
7.5.3 Vergleich



Mit dem Programmelement **Vergleich** lassen sich die an den beiden Dateneingängen A und B angelegten Werte miteinander vergleichen. Je nachdem ob **A** kleiner **B**, **A** größer **B** oder **A** gleich **B** ist, verzweigt das Element zum linken, rechten oder mittleren Ausgang. Die häufigste Anwendung dafür ist der Vergleich von einem Sollwert mit einem Istwert. Je nachdem wie der Sollwert zum Istwert liegt kann dann ein Motor zum Beispiel links oder rechts drehen oder angehalten werden.

Das **Vergleichen** Programmelement hat keinerlei Einstellmöglichkeiten und daher kein Eigenschaftsfenster.

7.5.4 Wartezeit



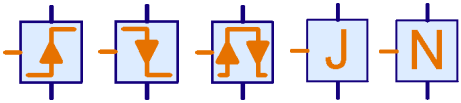
Mit diesem Element kann man eine **Wartezeit** in einem Ablauf programmieren. Die Wartezeit beginnt, wenn das Element im Ablauf an der Reihe ist. Sobald die eingegebene Wartezeit zu Ende ist, wird der Ablauf fortgesetzt. Siehe auch Abschnitt 3.6.1 *Wartezeit* auf Seite 22.

Eigenschaftsfenster für Wartezeit:

- Unter **Zeit** kannst du die Wartezeit eingeben. Du kannst auch Kommazahlen wie 1,23 verwenden.
- Unter **Zeiteinheit** kannst du als Zeiteinheit Sekunde, Minute oder Stunde auswählen. Die Zeiteinheit hat, anders als bei einer Timervariablen, keinen Einfluss auf die Genauigkeit der Wartezeit. Eine Wartezeit von 60 Sekunden und eine Wartezeit von 1 Minute verhalten sich genau gleich.

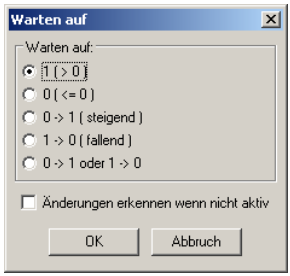
Im Expertenmodus (Level 5) wird ein erweitertes Eigenschaftsfenster angezeigt, das dem Eigenschaftsfenster für Timervariablen ähnlicher ist.

7.5.5 Warten auf…



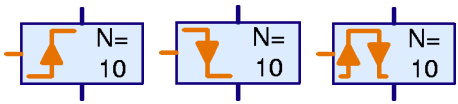
Das Programmelement **Warten auf…** verzögert die Programmausführung so lange, bis am Dateneingang des Elements eine Veränderung stattgefunden hat oder ein bestimmter Zustand eingetreten ist. Das Element gibt es in fünf Ausführungen: Das Element links wartet solange, bis sich der Wert am Eingang vergrößert hat. Dabei zählen nicht nur Änderungen von 0 nach 1, sondern jede Vergrößerung, also zum Beispiel auch von 2 nach 3. Das zweite Element wartet solange bis sich der Wert am Eingang verringert hat, und das Element in der Mitte wartet auf irgendeine Änderung, egal in welche Richtung. Das dritte Element wird häufig für Impulszahnräder verwendet. Das vierte und das fünfte Element warten nicht auf eine Änderung, sondern darauf, dass am Eingang der Zustand Ja (>0) oder Nein (<=0) anliegt. Wenn der betreffende Zustand bereits anliegt, wartet das Element nicht. Die ersten drei Elemente warten dagegen immer bis eine Änderung am Eingang erkennt wird.

Eigenschaftsfenster für Warten auf Änderung



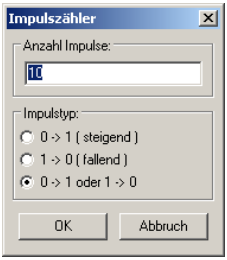
- Unter **Änderungsart** kannst du zwischen den fünf oben beschriebenen Funktionsweisen umschalten.
- Wenn der Knopf **Änderungen erkennen wenn nicht aktiv** gedrückt ist, erkennt das Element auch Änderungen die passiert sind, als das Element im Ablauf gar nicht an der Reihe war. In diesem Fall speichert das Element den zuletzt bekannten Wert. Wenn das Element dann erneut ausgeführt wird, setzt es die Programmausführung sofort fort, wenn sich der Wert in der Zwischenzeit auf die richtige Art geändert hat. Auf diese Weise ist die Wahrscheinlichkeit geringer, dass man eine Änderung verpasst, weil das Programm gerade etwas anderes gemacht hat.

7.5.6 Impulszähler



Dieses Programmelement wartet auf eine einstellbare Anzahl von Impulsen am Dateneingang auf der linken Seite, bevor es mit der Programmausführung fort fährt. Das ist für einfache Positionieraufgaben mit Impulszahnrädern recht praktisch. Für aufwändigere Positionierungen, z.B. mit einem veränderlichen Wert, müssen dann Unterprogramme mit Variablen verwendet werden.

Eigenschaftsfenster für Impulszähler



- Unter **Anzahl Impulse** gibst du die Anzahl der Impulse ein, auf die gewartet werden soll, bis die Programmausführung fortgesetzt wird.
- Unter **Impulstyp** kannst du zwischen den drei Impulsarten **0-1, 1-0** oder beliebiger Wechsel.

Die Möglichkeit Änderungen zu erkennen wenn das Element nicht aktiv ist wie beim einfachen **Warten auf…** Element, gibt es bei diesem Element nicht.

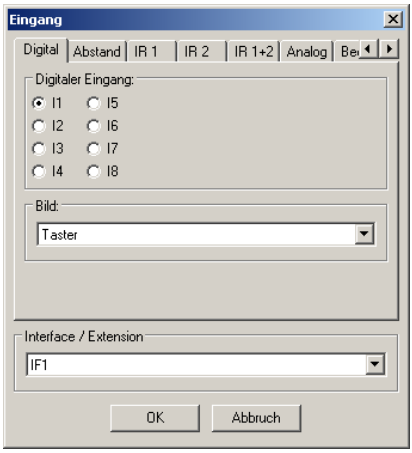
7.6 Interface-Eingänge / -Ausgänge, …

Diese Gruppe von Programmelementen enthält alle Ein- und Ausgangselemente. Wie diese Elemente verwendet werden ist in Kapitel 5 *Level 3: Variablen, Bedienfelder & Co* auf Seite 36.

7.6.1 Digitaleingang

Über das Element **Digitaleingang** lässt sich der Wert eines der Digitaleingänge I1 bis I8 des Interfaces abfragen. Sind die beiden zum Eingang gehörenden Buchsen am Interface miteinander verbunden, liefert das Digitaleingangselement an seinem orangen Anschluss einen Wert von 1, anderenfalls einen Wert von 0.

Eigenschaftsfenster für Digitaleingänge:



- Unter **Digitaler Eingang** kannst du auswählen, welcher Eingang des Interfaces verwendet werden soll. Eingänge von Erweiterungsmodulen wählst du unter **Interface / Extension** aus.
- Unter **Bild** kannst du ein Bild des Sensors auswählen, der an dem Eingang angeschlossen ist. In den meisten Fällen wird das ein **Taster** sein. Ein **Reed Kontakt** ist ein Schalter, der auf Magnetfelder reagiert. Auch ein **Fototransistor** lässt sich an einen Digitaleingang anschließen, obwohl er eigentlich ein analoger Sensor ist. An einem Digitaleingang angeschlossen kannst du den Fototransistor zusammen mit einer Linsenlampe als Lichtschranke verwenden, die entweder unterbrochen (= 0) oder geschlossen (= 1) ist. Wenn du den Fototransistor dagegen an einen *Analogeingang* anschließt, kannst du viele Abstufungen von hell und dunkel unterscheiden.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interface oder einen Eingang eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.

Genauer betrachtet gibt es für alle Arten von Eingängen nur ein einziges Programmelement. Über die Reiter oben im Eigenschaftsfenster kannst du die Eingangsart jederzeit umschalten. Das ist insbesondere zum Umschalten zwischen Schalter, IR- und Bedienfeld-Eingängen praktisch.

7.6.2 Analogeingang

Über das Element **Analogeingang** lässt sich der Wert eines der Analogeingänge abfragen. Im Gegensatz zu Digitaleingängen, die nur den Wert 0 oder 1 liefern können, können Analogeingänge feine Abstufungen unterscheiden. Alle Analogeingänge liefern einen Ausgangswert zwischen 0 und 1023. Beim ROBO Interface gibt es jedoch verschiedene Arten von Analogeingängen, die verschiedene physikalische Größen messen. Es gibt Analogeingänge für Widerstandmessungen, für Spannungsmessungen und für einen speziellen Sensor zur Abstandsmessung:

Eingang	Eingangstyp	Messbereich
A1, A2	Spannungseingänge	0-10,23V
AX, AY	Widerstandseingänge	0-5,5kΩ
D1, D2	Distanzsensoreingänge	ca. 0-50cm
AV	Versorgungsspannung	0-10V

Die üblichen fischertechnik Sensoren NTC-Widerstand, Foto-Transistor und Foto-Widerstand wandeln die Messgröße (Temperatur bzw. Lichtstärke) in einen Widerstand um. Daher musst du

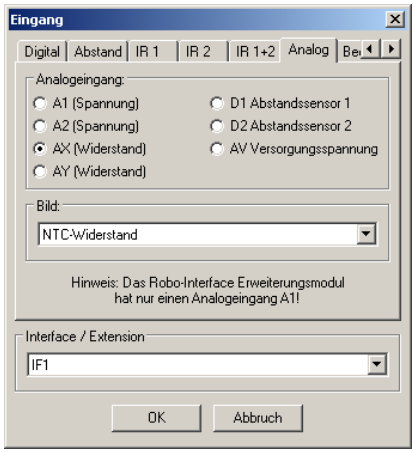
diese Sensoren an die **AX** oder **AY** Eingänge anschließen. Die Spannungseingänge **A1** und **A2** sind für alle Sensoren gedacht, die eine Spannung zwischen 0 und 10V ausgeben.

Für den **AV**-Eingang gibt es keine Buchse am ROBO Interface. Er ist immer mit der Versorgungsspannung des Interfaces verbunden. Auf diese Weise kannst du z.B. die Akkuspannung überwachen und dein Modell in die Ausgangsposition bringen, bevor der Akku leer ist.

An die Distanzsensoreingänge **D1** und **D2** können spezielle Sensoren von fischertechnik angeschlossen werden, die den Abstand z.B. zu einem Hindernis messen können.

Das Intelligent Interface hat nur zwei Analogeingänge, EX und EY. Diese entsprechen den AX und AY Eingängen des ROBO–Interface. Die anderen Analogeingänge können mit dem Intelligent Interface nicht verwendet werden!

Eigenschaftsfenster für Analogeingänge:



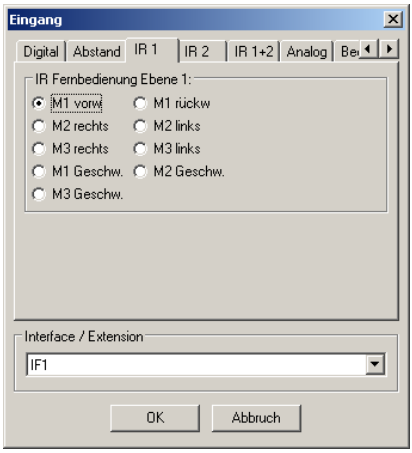
- Unter **Analogeingang** kannst Du gemäß oben stehender Tabelle den gewünschten Analogeingang auswählen.
- Unter **Bild** kannst du ein Bild des Sensors auswählen, der an dem Eingang angeschlossen ist.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Eingang des Interfaces, eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.

An dem Eigenschaftsfenster für Analogeingänge wird wieder deutlich, dass ROBO Pro für alle Eingänge nur ein einziges Element verwendet, das sich über die Reiter auf alle Eingangsarten umschalten lässt. Zur Vereinfachung stehen aber im Elementfenster bereits verschiedene Eingangselemente zur Auswahl bereit.

7.6.3 IR-Eingang

Im ROBO Interface ist ein Infrarot-Empfänger für den Handsender aus dem fischertechnik ***IR Control Set*** Art.-Nr. 30344 eingebaut. Der Infrarothandsender ist nicht nur zum Fernsteuern, sondern allgemein als Tastatur für deine Modelle sehr praktisch. Für das ***IR Control Set*** gibt es zwei Empfänger, zwischen denen man mit den Tasten **1** und **2** am Handsender umschalten kann. Im ROBO Interface kannst du daher jede Taste des Handsenders doppelt belegen. Zwischen den zwei Belegungen kannst du mit den Umschaltasten **1** und **2** umschalten. Alternativ können die Tasten **1** und **2** als ganz normale Tasten verwendet werden.

Im Eigenschaftsfenster eines IR-Eingangs kannst du in der Reiterleiste oben zwischen **IR 1**, **IR 2** und **IR 1+2** umschalten. Wenn du **IR 1** ausgewählt hast, liefert das IR-Eingangselement nur dann eine 1, wenn der entsprechende Taster am Sender gedrückt ist, und der Sender zuvor über die **1** Taste auf Belegung 1 eingestellt worden ist. Wenn du **IR 2** auswählst, muss der Sender dagegen über die Taste **2** auf Belegung 2 eingestellt sein.



Wenn du aber **IR 1+2** auswählst, ist es egal wie der Handsender eingestellt ist. Dann kannst du die Tasten 1))) und 2))) ebenfalls als Eingänge verwenden.

Im Programmelement wird die Auswahl durch eine weiße 1 oder 2 unten rechts im Handsendersymbol angezeigt. Bei **IR 1+2** wird im Programmelement keine Zahl angezeigt.

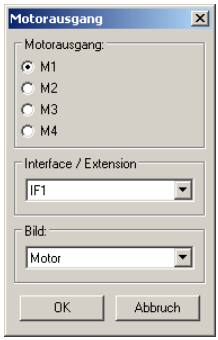
7.6.4 Motorausgang

Über das Element **Motorausgang** lässt sich einer der 4 zweipoligen Motorausgänge eines ROBO Interface oder Intelligent Interface ansteuern. Ein Motorausgang verwendet immer zwei Anschlüsse des Interface, während ein Lampenausgang nur einen Anschluss verwendet. Mehr zum Unterschied zwischen Motor- und Lampenausgang erfährst du in den Abschnitten 7.1.6 *Motorausgang* auf Seite 58 und 7.1.7 *Lampenausgang*.

An einen Motorausgang muss über ein Befehlselement ein Befehl geschickt werden, damit der Ausgang geschaltet wird. Ein Motorelement kann die folgenden Befehle verarbeiten:

Befehl	Wert	Aktion
Rechts	1 bis 8	Der Motor dreht sich nach rechts mit Geschwindigkeit 1 bis 8
Links	1 bis 8	Der Motor dreht sich nach links mit Geschwindigkeit 1 bis 8
Stopp	keiner	Der Motor stoppt
Ein	1 bis 8	Wie Rechts
Aus	keiner	Wie Stopp
=	-8 bis 8	Wert -1 bis -8: Der Motor dreht sich links Wert 1 bis 8: Der Motor dreht sich rechts Wert 0: Der Motor stoppt

Eigenschaftsfenster für Motorelemente:



- Unter **Motorausgang** kannst du auswählen, welche Ausgangsanschlüsse des Interface verwendet werden sollen. Ausgänge von Erweiterungsmodulen wählst du unter **Interface / Extension** aus.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Ausgang des Interface, eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild des Verbrauchers auswählen, der an dem Ausgang angeschlossen ist. In den meisten Fällen wird das ein **Motor** sein. Du kannst aber auch einen **Elektromagnet**, ein **Magnetventil** oder eine **Lampe** an einen Motorausgang anschließen.

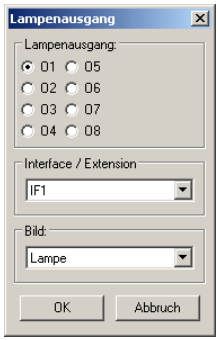
7.6.5 Lampenausgang

Über das Element **Lampenausgang** lässt sich einer der 8 einpoligen Lampenausgänge O1-O8 eines ROBO Interface oder Intelligent Interface ansteuern. Ein Lampenausgang verwendet immer nur einen Ausgangs-Anschluss des Interface. Der andere Anschluss des Verbrauchers wird mit Massebuchse verbunden. Einen so angeschlossenen Verbraucher kannst du nur ein- oder ausschalten, aber nicht umpolen. Mehr zum Unterschied zwischen Motor- und Lampenausgang erfährst du in den Abschnitten 7.1.6 *Motorausgang* auf Seite 58 und 7.1.7 *Lampenausgang*.

An ein Lampenelement muss über ein Befehlselement ein Befehl geschickt werden, der den Ausgang schaltet. Ein Lampenelement kann die folgenden Befehle verarbeiten:

Befehl	Wert	Aktion
Ein	1 bis 8	Die Lampe wird mit einer Helligkeit von 1 bis 8 eingeschaltet
Aus	keiner	Die Lampe wird ausgeschaltet
=	0 bis 8	Wert 1 bis 8: Die Lampe wird eingeschaltet Wert 0: Die Lampe wird ausgeschaltet

Eigenschaftsfenster für Lampenausgangselemente:



- Unter **Lampenausgang** kannst du auswählen, welcher Ausgangsanschluss des Interface verwendet werden soll. Ausgänge von Erweiterungsmodulen wählst du unter **Interface / Extension** aus.
- Unter **Interface / Extension** kannst du auswählen, ob du einen Ausgang des Interface, eines Erweiterungsmoduls oder eines zweiten Interface verwenden möchtest. Mehr dazu erfährst du im Kapitel 6 *Erweiterungsmodule und mehrere Interfaces ansteuern* auf Seite 51.
- Unter **Bild** kannst du ein Bild des Verbrauchers auswählen, der an dem Ausgang angeschlossen ist. In den meisten Fällen wird das eine **Lampe** sein. Du kannst aber auch einen **Elektromagnet**, ein **Magnetventil** und sogar einen **Motor** an einen

Lampenausgang anschließen. Ein an einen Lampenausgang angeschlossener Motor kann sich aber immer nur in eine Richtung drehen.

7.6.6 Bedienfeldeingang

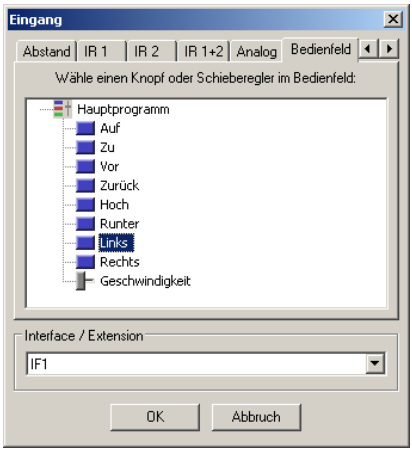
ROBO Pro bietet die Möglichkeit eigene Bedienfelder für deine Modelle zu zeichnen. Mehr dazu erfährst du im Kapitel 8 *Referenz Bedienelemente und Bedienfelder* auf Seite 82. Auf diese Weise kannst du deine Modelle komfortabel vom Computer aus steuern. Im Bedienfeld stehen Druckknöpfe, Schieberegler und Eingabeelemente zur Verfügung. Der Zustand dieser Elemente lässt sich im Programm über das Element **Bedienfeldeingang** abfragen. Druckknöpfe liefern einen Wert von 0 oder 1. Schieberegler liefern einen Wert in einem einstellbaren Bereich (standardmäßig 0 bis 100).

Bedienfelder lassen sich nur im Online -Modus verwenden. Mehr dazu erfährst du im Abschnitt 3.7 *Online- oder Download-Betrieb – Wo ist denn da der Unterschied?* auf Seite 24.

Eigenschaftsfenster für Bedienfeldeingänge:

Zu jedem Haupt- oder Unterprogramm gehört ein Bedienfeld. Unter dem Namen der Programme sind die Bedienelemente aufgelistet. Wenn du noch keine Bedienelemente angelegt hast, erscheinen auch keine Elemente in der Liste. Du musst also zuerst das Bedienfeld zeichnen, bevor du einen Bedienfeldeingang mit einem Bedienelement verbinden kannst.

Die Auswahl unter **Interface / Extension** wird bei Bedienfeldeingängen ignoriert, da es sich hierbei nicht um tatsächliche Eingänge an einem Interface-Modul handelt.



7.6.7 Bedienfeldausgang

ROBO Pro bietet die Möglichkeit eigene Bedienfelder für deine Modelle zu zeichnen. Mehr dazu erfährst du im Kapitel 8 *Referenz Bedienelemente und Bedienfelder* auf Seite 82. Neben Druckknöpfen und anderen Eingabeelementen zur Steuerung deines Modells, kannst du in einem Bedienfeld auch Anzeigelemente einfügen. In den Anzeigelementen kannst du zum Beispiel die Achs-Koordinaten eines Roboters oder den Zustand eines Endschalters anzeigen lassen. Den angezeigten Wert änderst du, indem du in deinem Programm ein Element **Bedienfeldausgang** einfügst und dem Element einen **=**-Befehl schickst, z. B. indem du eine Variable, einen Analogeingang oder ein Befehlselement daran anschließt.

Bedienfelder lassen sich nur im Online-Modus verwenden mehr dazu erfährst du im Abschnitt 3.7 *Online- oder Download-Betrieb – Wo ist denn da der Unterschied?* auf Seite 24.

Eigenschaftsfenster für Bedienfeldanzeigen:

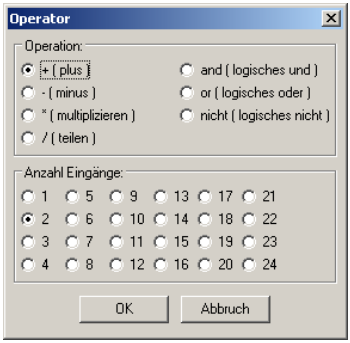
Zu jedem Haupt- oder Unterprogramm gehört ein Bedienfeld. Unter dem Namen der Programme sind die Bedienfeldanzeigen aufgelistet. Wenn du noch keine Bedienelemente angelegt hast, erscheinen auch keine Elemente in der Liste. Du musst also zuerst das Bedienfeld zeichnen, bevor du einen Bedienfeldeingang mit einem Bedienelement verbinden kannst.

7.7 *Operatoren*

Die Programmelemente dieser Gruppe sind so genannte Operatoren. Die Operatoren haben einen oder mehrere orange Dateneingänge. Die Werte an den Dateneingängen werden vom Operator zu einem Wert verknüpft und am Ausgang des Operators per **=**-Befehl ausgegeben.

Eigenschaftsfenster für Operatoren

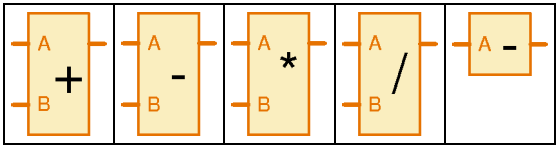
Alle Operatoren verwenden das gleiche Eigenschaftsfenster. Du kannst über das Eigenschaftsfenster auch einen Operator in einen anderen Operator umwandeln.



- Unter **Operation** kannst du einstellen, wie der Operator seine Eingänge verknüpfen soll. Die einzelnen Funktionen sind in den nächsten beiden Abschnitten erklärt.
- Unter **Anzahl Eingänge** kannst du einstellen, wie viele Eingänge der Operator habe soll.

7.7.1 Arithmetische Operatoren

ROBO Pro stellt dir die vier Grundrechenarten als Opertoren zur Verfügung. Die Symbole bei zwei Eingängen sehen so aus:



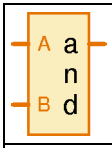
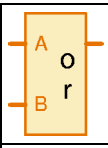
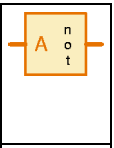
Plus	Minus	Mal	Geteilt	Minus
A + B	A - B	A * B	A / B	- A

Wenn der **Minus** Operator mehr als zwei Eingänge hat, werden alle weiteren Eingangswerte vom Wert am Eingang A abgezogen. Wenn der Minus Operator nur einen Eingang hat, kehrt der Operator das Vorzeichen des Eingangswertes um.

Wenn der **Geteilt** Operator mehr als zwei Eingänge hat, wird der Wert am Eingang A durch alle weiteren Eingangswerte geteilt.

7.7.2 Logische Operatoren

Zum Beispiel zur Verknüpfung von Digitaleingängen gibt es in ROBO Pro drei logische Operatoren:

		
Und	Oder	Nicht
A>0 und B>0	A>0 oder B>0	A <= 0

Die logischen Operatoren interpretieren einen Wert größer Null am Eingang als **ja** oder **wahr** und einen Wert kleiner oder gleich Null als **nein** oder **falsch**. Digitaleingänge liefern einen Wert von 0 oder 1, so dass 0 als **falsch** und 1 als **wahr** interpretiert wird.

Der **Und** Operator schickt an die am Ausgang angeschlossenen Elemente einen **=** Befehl mit Wert 1, wenn an allen Eingängen der Wert wahr, also ein Wert > 0 anliegt. Anderenfalls sendet das Element einen **=** Befehl mit Wert 0.

Der **Oder** Operator schickt an die am Ausgang angeschlossenen Elemente einen **=** Befehl mit Wert 1, wenn an mindestens einem der Eingänge der Wert wahr ist, also ein Wert > 0 anliegt. Anderenfalls sendet das Element einen **=** Befehl mit Wert 0.

Der **Nicht** Operator schickt an die am Ausgang angeschlossenen Elemente einen **=** Befehl mit Wert 1, wenn am Eingang der Wert falsch, also ein Wert <= 0 anliegt. Anderenfalls sendet das Element einen **=** Befehl mit Wert 0.

Die Funktion von logischen Operatoren kann man auch mit mehreren Verzweigungs-Elementen nachbilden. Aber oft ist es viel übersichtlicherer mehrere Eingänge mit Operatoren zu verknüpfen.

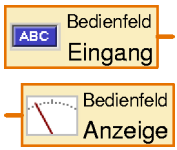
8 Übersicht Bedienelemente und Bedienfelder

In ROBO Pro lassen sich eigene Bedienfelder definieren. Über Bedienfelder kannst du komplexe Modelle komfortabel steuern. Da das Bedienfeld auf dem PC-Bildschirm angezeigt wird, funktionieren Bedienfelder nur im Online-Modus. Siehe dazu Abschnitt 3.7 *Online- oder Download-Betrieb – Wo ist denn da der Unterschied?* auf Seite 24.

Um ein Bedienfeld zu erstellen, wählst du in der Funktionsleiste **Bedienfeld** aus:

In dem leeren grauen Feld darunter kannst du dann Bedienelemente einfügen. Ein Bedienfeld gehört immer zu dem Haupt- oder Unterprogramm, in dem du dich bei der Erstellung des Bedienfeldes befunden hast. Daher ist es wichtig, dass du in der Unterprogrammleiste das richtige Unterprogramm auswählst, bevor du ein Bedienfeld anlegst. Meistens legt man Bedienfelder unter dem **Hauptprogramm** an.

In Bedienfeldern gibt es Anzeigen und Steuerelemente. Mit Anzeigen kannst du zum Beispiel Werte von Variablen oder Textnachrichten anzeigen. Steuerelemente funktionieren dagegen wie zusätzliche Taster oder Analogeingänge.



Zu jedem Bedienelementen, das du im Bedienfeld einfügst, gehört im Programm ein Element **Bedienfeldeingang** (für Steuerelemente) oder **Bedienfeldausgang** (für Anzeigen). Über diese Programmelemente stellst du die Verbindung zwischen deinem Programm und deinem Bedienfeld her. Du findest sie in der Elementgruppe **Eingänge, Ausgänge**. Je nachdem mit welcher Art von Bedienelement du diese Programmelemente verbindest, wird ein anderes Symbol angezeigt. In der Elementliste gibt es aber nur zwei Elemente: eines für Anzeigen und eines für Steuerelemente.

8.1 Anzeigen

Anzeigen werden so ähnlich verwendet wie Interfaceausgänge. Den Wert einer Anzeige kannst du mit einem **=**-Befehl setzen.

8.1.1 Messgerät

Das **Messgerät** ist einem analogen Zeigerinstrument nachempfunden. Es wird meistens verwendet um den Wert von Analogeingängen anzuzeigen, du kannst es aber auch für Variablen oder andere Programmelemente verwenden.

Das Messgerät wird vom Programm aus über einen Bedienfeldausgang gesteuert. Den **Bedienfeldausgang** findest du in der Elementgruppe **Eingänge, Ausgänge**.

Den Wert des Messgeräts setzt du, indem du dem zugehörigen Bedienfeldausgang im Programm einen **=**-Befehl schickst. Fast alle Programmelemente mit Datenausgängen schicken einen **=**-Befehl, wenn sich ihr Wert ändert. Analogeingänge oder Variablen kannst du zum Beispiel direkt mit dem Bedienfeldausgang verbinden.

Eigenschaftsfenster für Messgeräte

- Unter **ID / Name** solltest du zunächst einen Namen für das Messgerät eingeben. Der Name ist wichtig, damit du mehrere Messgeräte in deinem Programm voneinander unterscheiden kannst.
- Unter **Hintergrundfarbe** kannst du eine andere Farbe als weiß einstellen.
- Unter **Minimalwert** und **Maximalwert** gibst du die Werte an, die der Zeigerposition am linken und rechten Rand der Skala entsprechen. Wenn einer der Werte kleiner als 0 und der andere größer als 0 ist, wird ein besonders langer 0 Strich gezeichnet.
- Die Skala besteht aus langen und kurzen Strichen. Den Abstand der langen und kurzen Striche gibst du unter **Schreitweite kurze / lange Marken** ein. Wenn beide den gleichen Wert haben, sind nur lange Marken sichtbar.

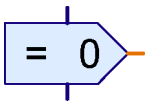
8.1.2 Textanzeige

In einer Textanzeige kannst du Zahlenwerte, Text oder beides gemischt anzeigen.

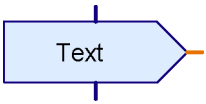
Die Textanzeige wird vom Programm aus über einen Bedienfeldausgang gesteuert. Den **Bedienfeldausgang** findest du in der Elementgruppe **Eingänge, Ausgänge**.

Sobald du den Bedienfeldausgang über sein Eigenschaftsfenster mit einer Textanzeige verbunden hast, ändert sich das Symbol und es erscheint der Name des Bedienfelds (z.B. Haupt) und der Anzeige (z.B. Text).

Du kannst den Text in der Anzeige Auf zwei Arten setzen:



- Du kannst den Inhalt der Anzeige setzen, indem du dem zugehörigen Bedienfeldausgang einen **=**-Befehl schickst. Das ist sehr praktisch, wenn du die Anzeige verwenden möchtest, um den Wert einer Variablen oder anderen Programmelements anzuzeigen, weil die meisten Programmelemente über ihre Datenausgänge automatisch **=**-Befehle schicken, wenn sich der Wert ändert. Der **=**-Befehl überschreibt nur die letzten 6 Zeichen der Anzeige. Den Rest der Anzeige kannst du mit einem vorgegebenen Text füllen. Auf diese Weise kannst du zum Wert einen Hinweistext in die Anzeige setzen. Wenn die Anzeige mehrzeilig ist, kannst du den Hinweistext auch in eine eigene Zeile setzen. Bei mehrzeiligen Anzeigen werden von einem **=**-Befehl nur die letzten 6 Zeichen der letzten Zeile überschrieben.



- Mit dem Text-Befehl kannst du den Inhalt der Anzeige ganz beliebig setzen. Der Text Befehl ist ein spezielles Befehlselement, das nicht nur eine Zahl, sondern einen ganzen Text über seinen Ausgang senden kann. Wie ein gewöhnliches Befehlselement kann das Befehlselement **Text** auch einen Dateneingang haben. Du

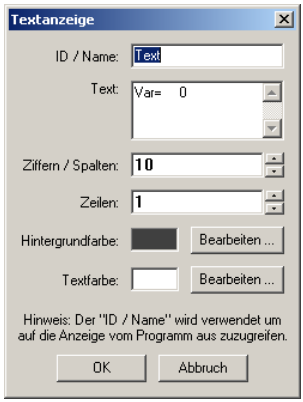
kannst dann den am Dateneingang anliegenden Zahlenwert in den Text einbauen. Wenn du einem Anzeigeelement mehrere **Text**-Befehle schickst, werden die Texte aneinander gehängt. Auf diese Weise kannst du Zahlen und Text beliebig mischen.

Steuerzeichen in Text-Befehlen

Im Befehlselement **Text** kannst du folgende Zeichen verwenden, um besondere Effekte zu erzielen:

Steuerzeichen	Effekt
######	Gibt den Wert am Dateneingang als Zahl mit 5 Stellen + Vorzeichen aus
##.##	Gibt den Wert am Dateneingang als Zahl mit 2 Nachkommastellen aus, die durch ***Punkt*** getrennt sind.
##,##	Gibt den Wert am Dateneingang als Zahl mit 2 Nachkommastellen aus, die durch ***Komma*** getrennt sind.
\c	Anzeige löschen und Einfügepunkt an den Anfang der Anzeige setzen

Eigenschaftsfenster für Textanzeigen



- Unter **ID / Name** solltest du zunächst einen Namen für die Anzeige eingeben. Der Name ist wichtig, damit du mehrere Anzeigen in deinem Programm voneinander unterscheiden kannst.
- Unter **Text** gibst du den Inhalt der Anzeige ein. Dieser Inhalt bleibt solange erhalten, bis du vom Programm aus einen Befehl an die Anzeige schickst. Wenn du einen **=**-Befehl an die Anzeige schickst, werden nur die letzten 6 Zeichen des Anzeigeninhalts überschrieben. Der Anfang des Textes bleibt erhalten, so dass du vor für eine Zahl noch einen Hinweis anzeigen kannst, um was für eine Zahl es sich handelt. Im abgebildeten Beispiel bleibt der Text **Var=** erhalten. Die Anzeige hat 10 Zeichen, also bleiben 10-6=4 Zeichen erhalten.
- Unter **Ziffern/Spalten** und unter **Zeilen** kannst du eingeben, für wie viel Zeichen die Anzeige Platz bieten soll. In einer mehrzeiligen Anzeige kannst du einen Hinweis wie ***Var=*** oder **Besucher** in einer eigenen Zeile anzeigen.
- Unter **Hintergrundfarbe** und **Textfarbe** kannst du das Farbdesign der Anzeigen verändern. Klicke auf **Bearbeiten ...** um eine Farbe auszuwählen oder eine eigene Farbe zu definieren.

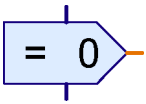
8.1.3 Anzeigelampe

Die **Anzeigelampe** ist die einfachste Art von Anzeige. Sie funktioniert so ähnlich wie ein fischertechnik Lampenbaustein, der an einem Ausgang des Interface angeschlossen ist.

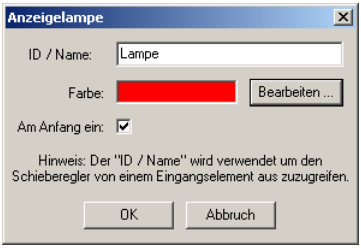
Die Anzeigelampe wird vom Programm aus über einen Bedienfeldausgang gesteuert. Den **Bedienfeldausgang** findest du in der Elementgruppe **Eingänge, Ausgänge**.

Sobald du den Bedienfeldausgang über sein Eigenschaftsfenster mit einer Anzeigelampe verbunden hast, ändert sich das Symbol und es erscheint der Name des Bedienfelds (z.B. Haupt) und der Lampe.

Du kannst die Lampe ein- oder ausschalten, indem du dem zugehörigen Bedienfeldausgang in deinem Programm einen Befehl **Ein** oder **Aus** schickst, wie du ihn auch für richtige Lampenausgänge verwendest. Du kannst die Anzeigelampe auch über einen **=**-Befehl ein- und ausschalten. Ist der Wert größer 0, wird die Lampe eingeschaltet. Ist der Wert kleiner oder gleich 0, wird die Lampe ausgeschaltet.



Eigenschaftsfenster für Anzeigelampen



- Unter **ID / Name** solltest du zunächst einen Namen für die Anzeigelampe eingeben. Der Name ist wichtig, damit du mehrere Anzeigelampen in deinem Programm voneinander unterscheiden kannst.
- Unter **Farbe** kannst du die Farbe der Anzeigelampe ändern. Klicke dazu auf den **Bearbeiten** Knopf.
- Wenn **Am Anfang ein** angekreuzt ist, ist die Anzeigelampe an, bis das zugehörige Programmelement den ersten Befehl erhält. Anderenfalls ist die Anzeigelampe am Anfang aus.

8.2 Steuerelemente

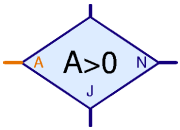
Steuerelemente werden so ähnlich verwendet wie Interfaceeingänge.

8.2.1 Knopf

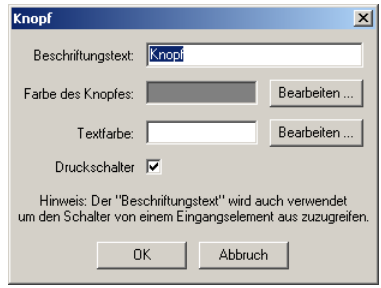
Das **Bedienelement Knopf** kannst du verwenden wie einen fischertechnik Taster oder Schalter, der an einem der Eingänge des Interface angeschlossen ist.

Der Knopf wird vom Programm aus über einen **Bedienfeldeingang** abgefragt. Den **Bedienfeldeingang** findest du in der Elementgruppe **Eingänge, Ausgänge**.

Den zum Knopf gehörenden Bedienfeldeingang kannst du wie einen Digitaleingang des Interface an alle Programmelemente mit einem Dateneingang, zum Beispiel an die **Verzweigung**, anschließen. Wenn der Knopf gedrückt ist liefert er als Wert 1, anderenfalls den Wert 0.



Eigenschaftsfenster für Knöpfe



- Unter **Beschriftungstext** kannst du die Beschriftung für den Knopf eingeben. Dies ist gleichzeitig der Name, mit dem vom Programm aus auf den Knopf zugegriffen wird. Ein zusätzliche Name/ID Feld wie bei den anderen Bedienelementen gibt es beim Knopf nicht.
- Unter **Farbe des Knopfes** und **Textfarbe** kannst du das Farbdesign des Knopfes verändern. Klicke dazu auf **Bearbeiten…**.
- Wenn bei Druckschalter ein Häkchen erscheint, wirkt der Knopf nicht wie ein Taster, sondern wie ein Schalter. Beim ersten Klick auf den Knopf wird der Knopf hineingedrückt und bleibt dann bis zum zweiten Klick gedrückt. Anderenfalls wirkt der Knopf als Taster und springt sofort wieder heraus, wenn er losgelassen wird.

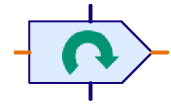
8.2.2 Regler

Den Regler kannst du wie ein Potentiometer verwenden, das an einem Analogeingang des Interface angeschlossen ist. Anders als der Knopf kann der Regler nicht nur die Werte 0 und 1 sondern wie ein Analogeingang viele Werte liefern. Der Wertebereich ist über das Eigenschaftsfenster einstellbar. Der Regler kann zum Beispiel verwendet werden um die Motordrehzahl zwischen 1 und 8 einzustellen.

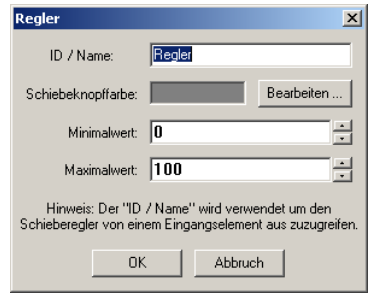
Der Regler wird vom Programm aus über einen Bedienfeldeingang abgefragt. Den **Bedienfeldeingang** findest du in der Elementgruppe **Eingänge, Ausgänge**.

Sobald du den Bedienfeldeingang über sein Eigenschaftsfenster mit einem Regler verbunden hast, ändert sich das Symbol und es erscheint der Name des Bedienfelds (z.B. Haupt) und des Reglers.

Den zum Regler gehörenden Bedienfeldeingang kannst du wie einen Analogeingang des Interface an alle Programmelemente mit einem Dateneingang anschließen. Sehr häufig wird der Regler an ein Befehlselement mit Dateneingang angeschlossen, so dass der Regler die Geschwindigkeit eines Motors steuert.



Eigenschaftsfenster für Regler

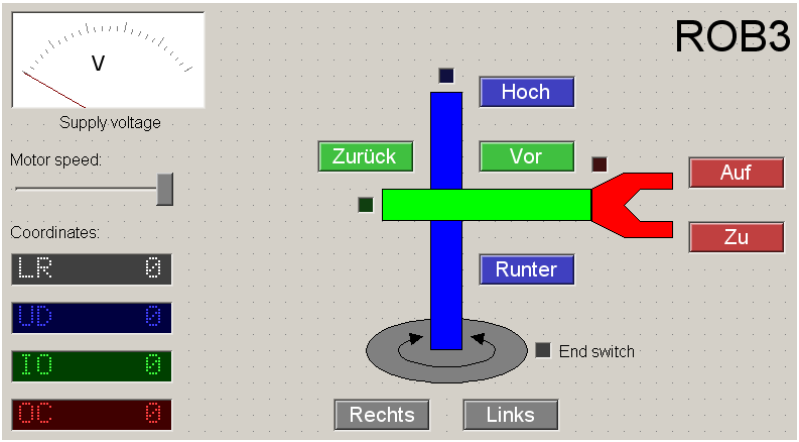


- Unter **ID / Name** solltest du zunächst einen Namen für den Regler eingeben. Der Name ist wichtig, damit du mehrere Regler im Programm voneinander unterscheiden kannst.
- Unter **Schiebeknopffarbe** kannst du die Farbe des Schiebeknopfes ändern. Klicke dazu auf **Bearbeiten**.
- Unter **Minimalwert** und **Maximalwert** gibst du den Wertebereich für den Regler ein. Wenn du den Regler zur Steuerung einer Motorgeschwindigkeit verwenden möchtest, sollte der Wertebereich von 1 bis 8 gehen.

9 Zeichenfunktionen

ROBO Pro verfügt über übliche Zeichenfunktionen. Du findest sie im Elementgruppenfenster unter **Zeichnen**. In der Untergruppe **Formen** sind Zeichenwerkzeuge für verschiedene geometrische Grundformen enthalten. In der Untergruppe **Text** findest du Textzeichenwerkzeuge für verschiedene Schriftgrößen. Die anderen Untergruppen enthalten Funktionen zum Ändern der Farbe und Linienstärke.

Mit den Zeichenfunktionen kannst du deine Bedienfelder und Programme illustrieren, um die Funktion zu verdeutlichen. Hier ist zum Beispiel ein selbst gezeichnetes Bedienfeld für einen Roboter abgebildet:



Die Knöpfe, Koordinatenanzeigen und Endschalterlampen sind jeweils in der gleichen Farbe gehalten wie die einzelnen Achsen in der schematischen Zeichnung des Roboters. Dadurch ergibt sich ein sehr übersichtliches Bedienfeld.

Die Anwendung der Zeichenfunktionen sollte keine großen Schwierigkeiten bereiten. Im Folgenden sind daher nur einige Punkte aufgeführt, die vielleicht nicht sofort klar sind:

- Grafische Objekte wie Rechtecke und Kreise werden **nicht** wie in vielen anderen Programmen mit gedrückter Maustaste aufgezogen, sondern durch zwei Mausklicks, einen in der linken oberen und einen in der rechten unteren Ecke.
- Text wird nicht in einem Dialogfenster bearbeitet, sondern direkt im Arbeitsbereich. Wenn du ein neues Textobjekt einfügest, erscheint zunächst nur ein hellblauer Rahmen. Du kannst nun einfach auf der Tastatur schreiben und der geschriebene Text erscheint dann direkt im Arbeitsbereich. Du kannst auch Text aus der Zwischenablage mit STRG+V einfügen.
- Nachdem du ein Objekt gezeichnet hast, kannst du das Objekt bearbeiten, indem du die kleinen blauen Griffpunkte verschiebst. Es gibt auch Griffpunkte zum Drehen und Verzerren von Objekten. Ein Rechteck hat oben links zwei Griffpunkte. Wenn du den zweiten größeren Griffpunkt verschiebst, kannst du die Ecken des Rechtecks abrunden. Den Bearbeitungsmodus kannst du beenden, indem du die rechte Maustaste oder die **ESC** Taste drückst.

- Wenn du ein Objekt nachträglich bearbeiten willst, wähle im Menü **Zeichnen** die Funktion **Bearbeiten**. Wenn du dann auf ein Objekt klickst, erscheinen wieder die hellblauen Griffpunkte.
- Viele Objekte haben zwei oder mehr Bearbeitungs- und Zeichenmodi. Du kannst zwischen den einzelnen Modi mit der Tabulator-Taste an der Tastatur wechseln, während du ein Objekt zeichnest oder bearbeitest. Bei einem Kreis kannst du zum Beispiel auswählen, ob du zwei Eckpunkte oder den Mittelpunkt und einen Eckpunkt angeben möchtest. Bei Polygonen kannst du zwischen Punktbearbeitung und Funktionen wie „Drehen“ wechseln. Bei Textobjekten kannst du zwischen Bearbeitung des Textes sowie Veränderung der Textgröße und des Drehwinkels umschalten.
- Im Menü **Zeichnen** gibt es die Funktionen Objekt **in den Vordergrund / Hintergrund**. Mit diesen Funktionen kannst du alle ausgewählten (rot nachgezeichneten) Objekte nach vorne oder hinten schieben, so dass sie andere Objekte überdecken oder von diesen überdeckt werden.
- Mit der Funktion **Rastersnap** im Menü **Zeichnen** kannst du das Zeichenraster ein- oder ausschalten. Du solltest aber darauf achten, dass das Raster eingeschaltet ist, wenn du dein Programm bearbeitest, da alle Programmelemente an das Raster angepasst sind.
- Bei Textobjekten kannst du die Textausrichtung verändern, indem du STRG + eine Taste von 1-9 am Nummernblock drückst. Das geht aber nur, wenn die Num-Lock Lampe an der Tastatur leuchtet. Falls nicht, musst du vorher die Num-Taste drücken.