



Coding I SCARA

EPSON Deutschland GmbH
Manufacturing Solutions | MS ACADEMY
Halskestr. 30 | 40880 Ratingen

Service: Tel.: +49 (0) 211 / 54229 - 009 |
service.ms@epson.de

Sales: Tel.: +49 (0) 211 / 54229 - 008 |
info.ms@epson.de

<http://www.epson.de/robots/>

EPSON®

Die Themen

Die Themen dieses Trainings

- RC+ Software
- Fahrbefehle (CP/PTP)
- Einfache Pick & Place Anwendung
- Erweiterte Pick & Place Anwendung
- SPS-Anbindung
- Multi-/ & Paralleltasks
- Lokales Koordinatensystem
- Werkzeug



RC+ - BEDIENOBERFLÄCHE

Sie können die Entwicklungsumgebung RC+ wie folgt öffnen:

- Klicken Sie auf des EPSON RC+ Icon auf Ihrem Windows Desktop:

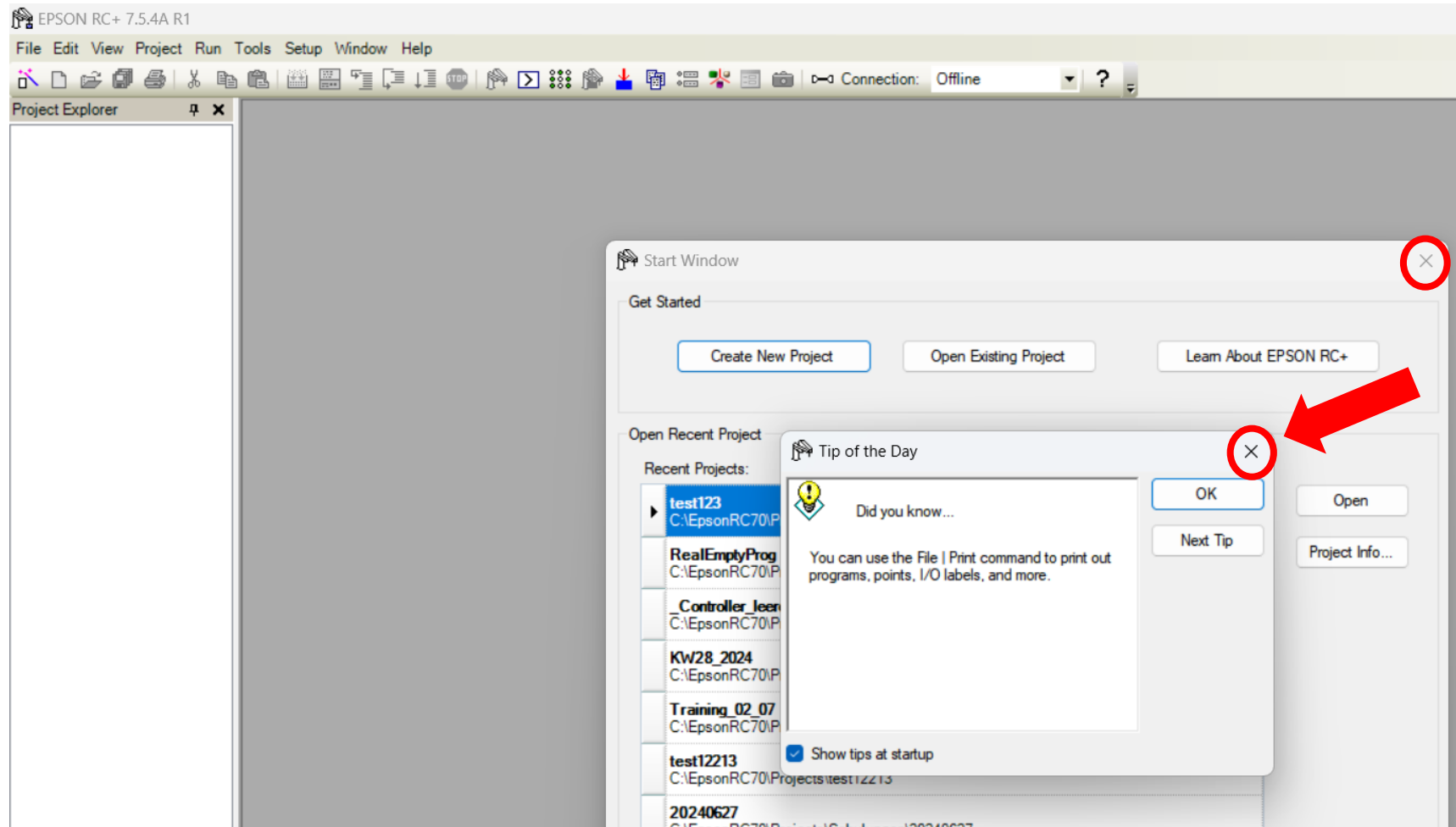


- Öffnen Sie die Software über:
Start -> Programme -> EPSON RC 7.0 -> EPSON RC+ 7.0

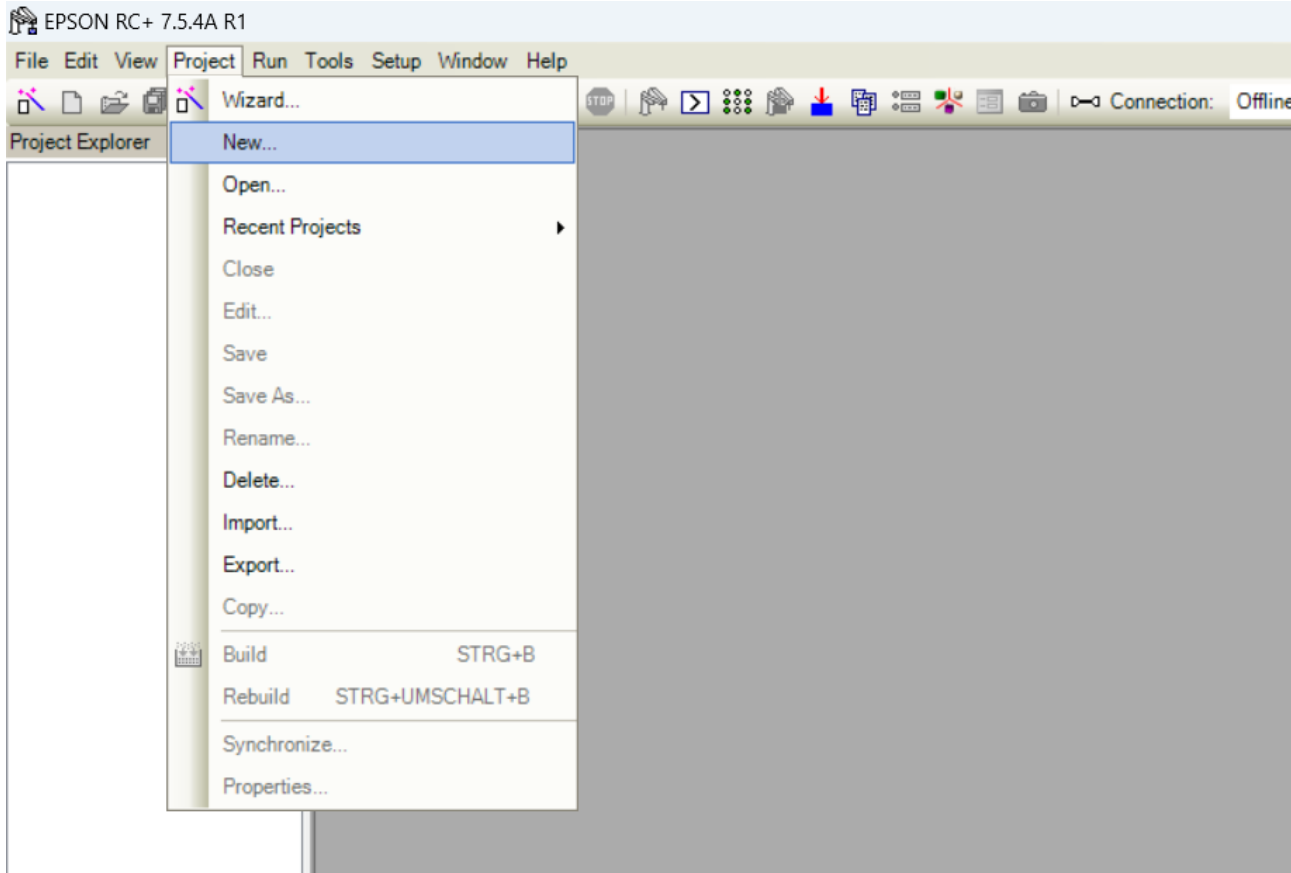
RC+ BEDIENoberfläche

Start der Software

Bitte schließen Sie das folgende Fenster:



Erstellen Sie nun ein neues Projekt:

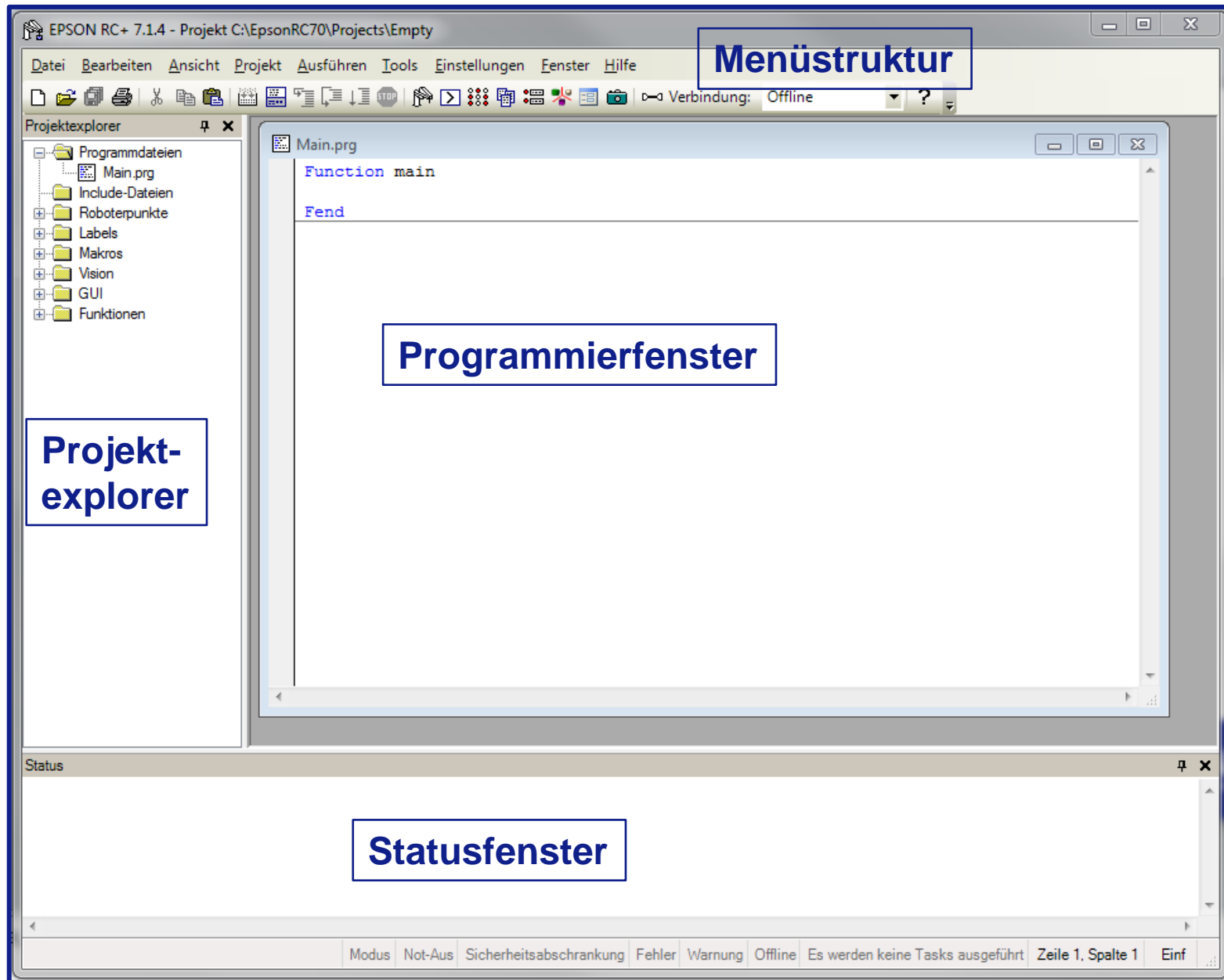


Über Menü:
„Projekt“ -> „Importieren“
können Sie ein auf der
Steuerung existierendes
Projekt in RC+
importieren.



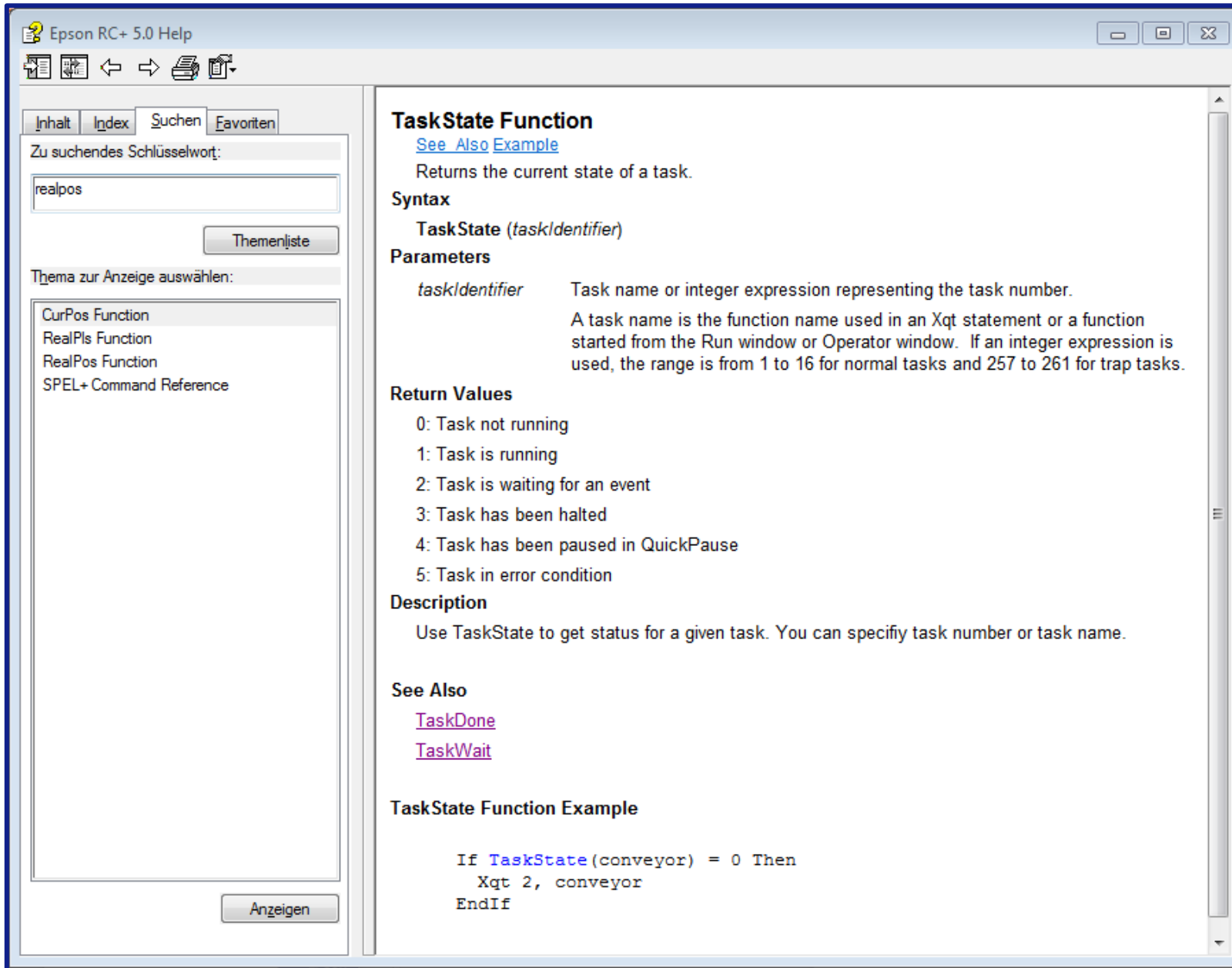
RC+ BEDIENOBERFLÄCHE

Layout



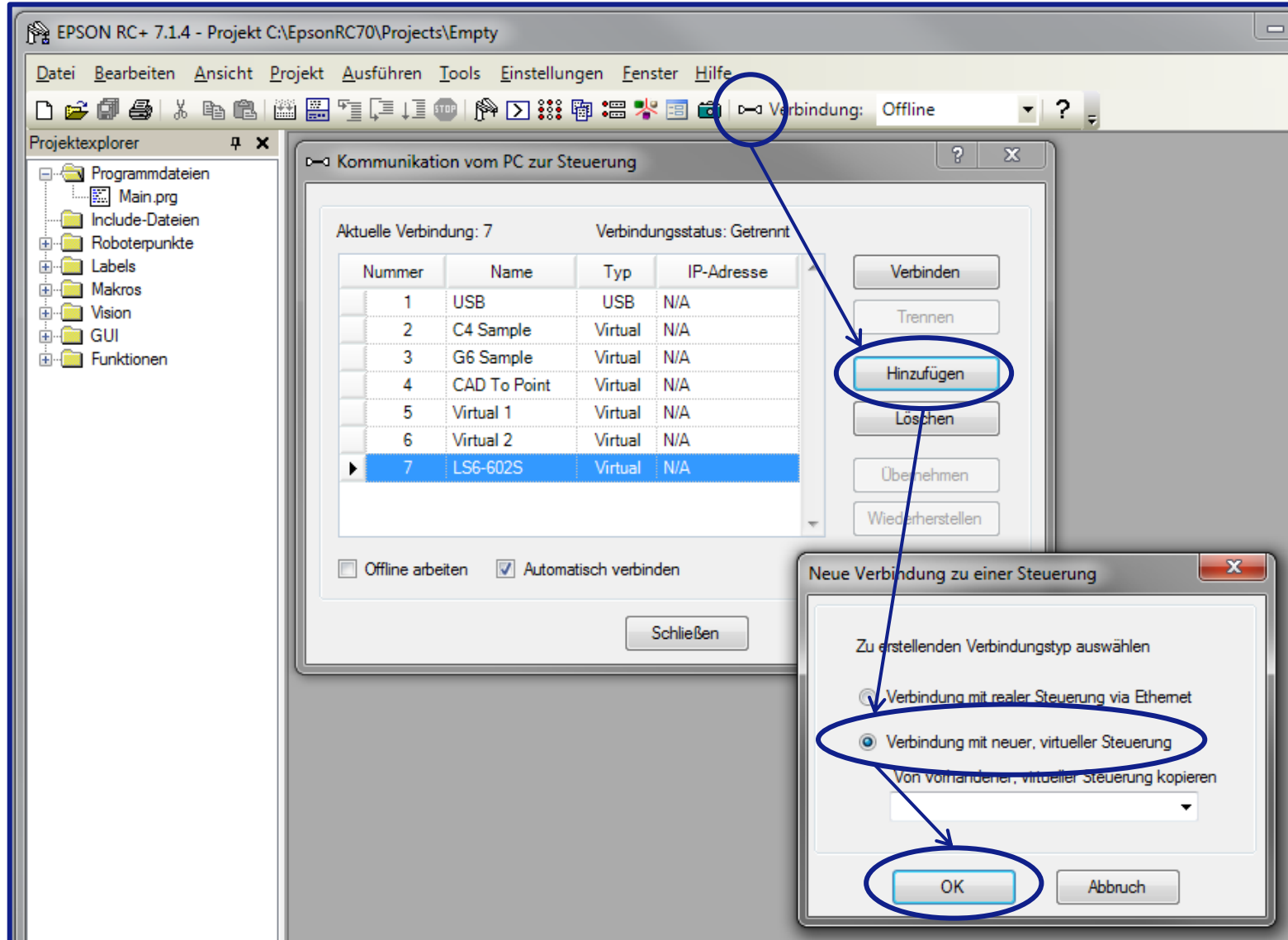
RC+ BEDIENOBERFLÄCHE

Hilfe benutzen



VIRTUELLE STEUERUNG

1. Virtuelle Verbindung anlegen



2. Zu simulierende Steuerung konfigurieren

The screenshot displays the EPSON virtual control software interface. The main window is titled "Roboter 1: Modell" and contains a table of robot specifications and a list of configuration options.

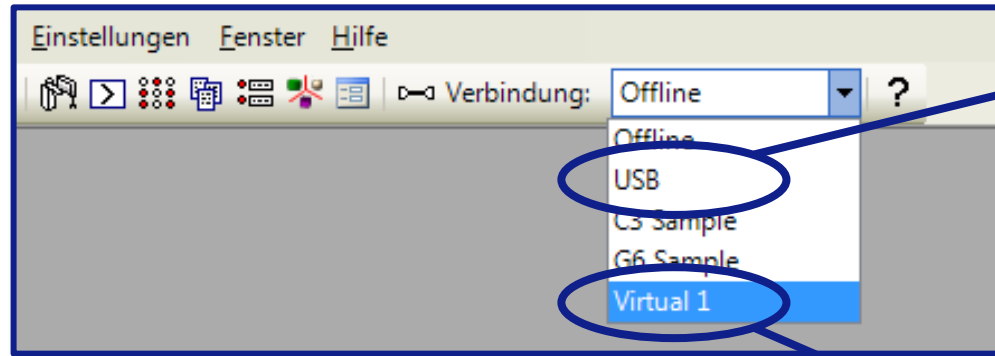
Roboter 1: Modell	
Modell:	LS6-602S
Typ:	Scara
J1 + J2 Länge:	600 mm
Z Länge:	200 mm
Max payload:	6 kg

Below the table is a small image of the robot arm. To the right of the table are several buttons: "Schließen", "Übernehmen", "Wiederherstellen", "Hinzufügen", "Löschen", and "Ändern...".

The left sidebar shows a tree view of the configuration options. The "Roboter" folder is expanded, and the "Modell" option is selected. A blue arrow points from the "Modell" option in the sidebar to the "Hinzufügen" button in the main window.

The top menu bar includes "Tools", "Einstellungen", "Fenster", and "Hilfe". The "Einstellungen" menu is open, showing options like "Kommunikation vom PC zur Steuerung...", "Systemeinstellungen...", "Voreinstellungen...", and "Optionen...". The "Systemeinstellungen..." option is highlighted with a blue circle.

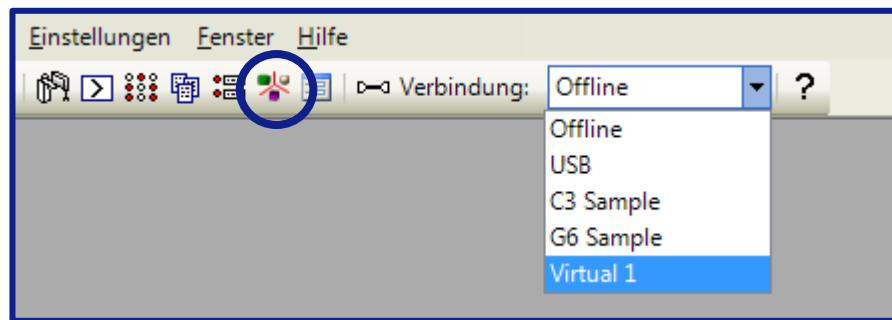
3. Verbindung auswählen



Verbindung zu einer Steuerung
über USB aufbauen

Verbindung zum virtuellen Simulator
aufbauen

4. Simulator starten

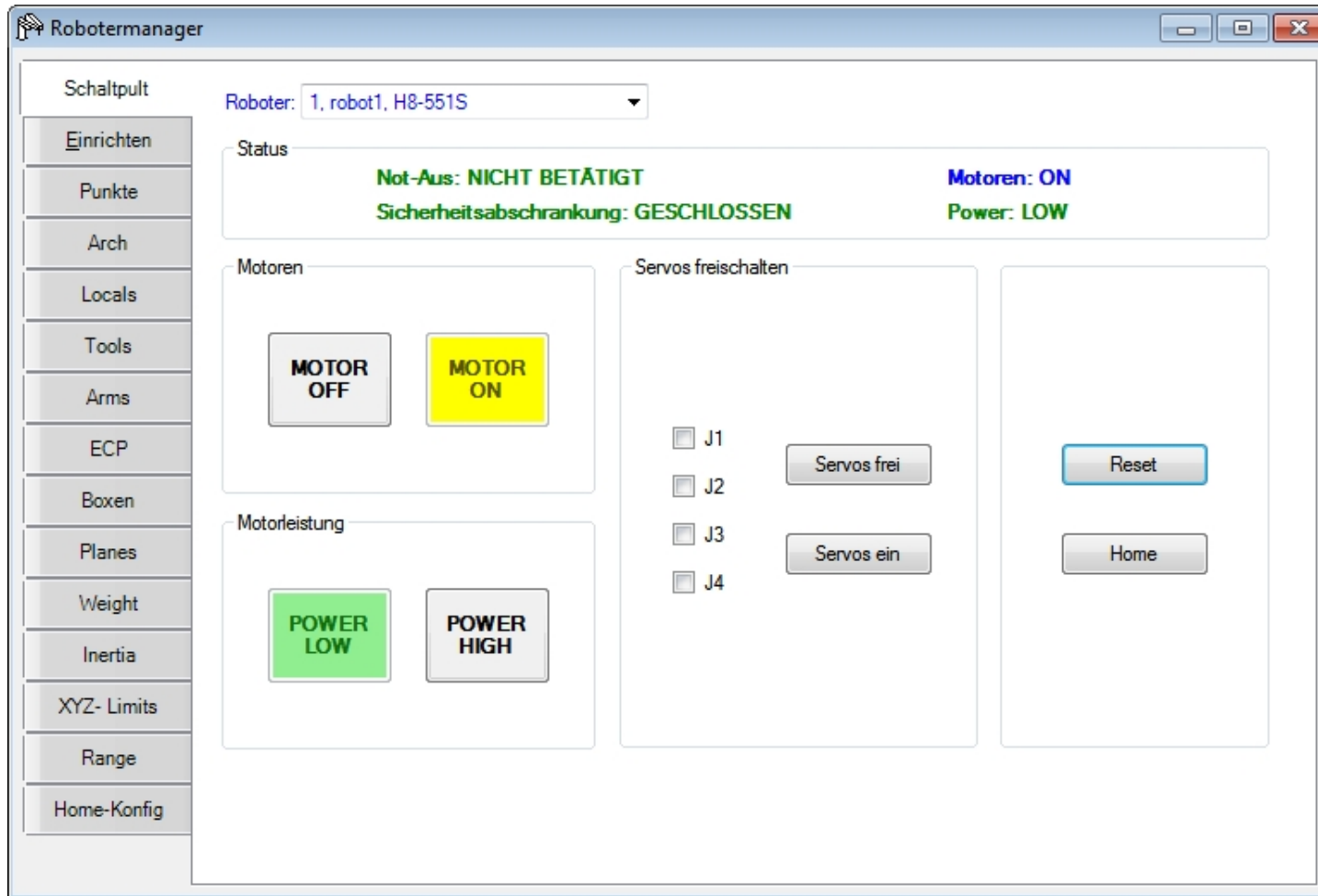


Oder über das Menü „Tools“ → „Simulator“

ROBOTER MANAGER

ROBOTER MANAGER

Tab „Schaltpult“



- Öffnen über Menü „Tools“ → „Roboter Manager“ oder Funktionstaste [F6]
- Ggf. Fehler löschen („Reset“)
- Motoren einschalten („MOTOR ON“)
- Über „Bewegungssteuerung“ in der Karte „Einrichten“ lässt sich der Manipulator nun verfahren.

ROBOTER MANAGER

Tab „Einrichten“

Robotermanager

Schaltpult

Einrichten

Punkte

Arch

Locals

Tools

Arms

ECP

Boxen

Planes

Weight

Inertia

XYZ- Limits

Range

Home-Konfig

Roboter: 1, robot1, H8-551S

Local: 0 Tool: 0 Arm: 0 ECP: 0

Aktuelle Position

X (mm)	Y (mm)	Z (mm)
372.207	108.591	-28.589
U (deg)	V (deg)	W (deg)
5.862		

☒ XY ☐ Achsen ☐ Pulse

Aktuelle Amorientierung

Hand	Elbow	Wrist	J4Flag
Righty			
			J6Flag

Schrittweite

X (mm)	Y (mm)	Z (mm)
U (deg)	V (deg)	W (deg)

☒ Continuous ☐ Groß ☐ Mittel ☐ Klein

Punkte teachen

Bewegungssteuerung

Modus: XY Speed: Low

+X -Y +Z -X +Y -Z -U -V -W +U +V +W

Bewegungsbefehl ausführen

Punktdatei: robot1.pts Punkt: P0 - Pick

Teach Editieren

ROBOTER MANAGER

Tab „Punkte“

Robotermanager

Schaltplan

Einrichten

Punkte

Arch

Locals

Tools

Arms

ECP

Boxen

Planes

Weight

Inertia

XYZ- Limits

Range

Home-Konfig

Roboter: 1, robot1, H8-551S

Punktdatei: robot1.pts

Nummer	Name	X	Y	Z	U	Local	Hand
0	Pick	372.207	108.591	-28.589	5.862	0	Righty
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

P0 löschen

Alles löschen

Speichern

Wiederherstellen

Aufgabenstellung: 1. Ablaufprogramm

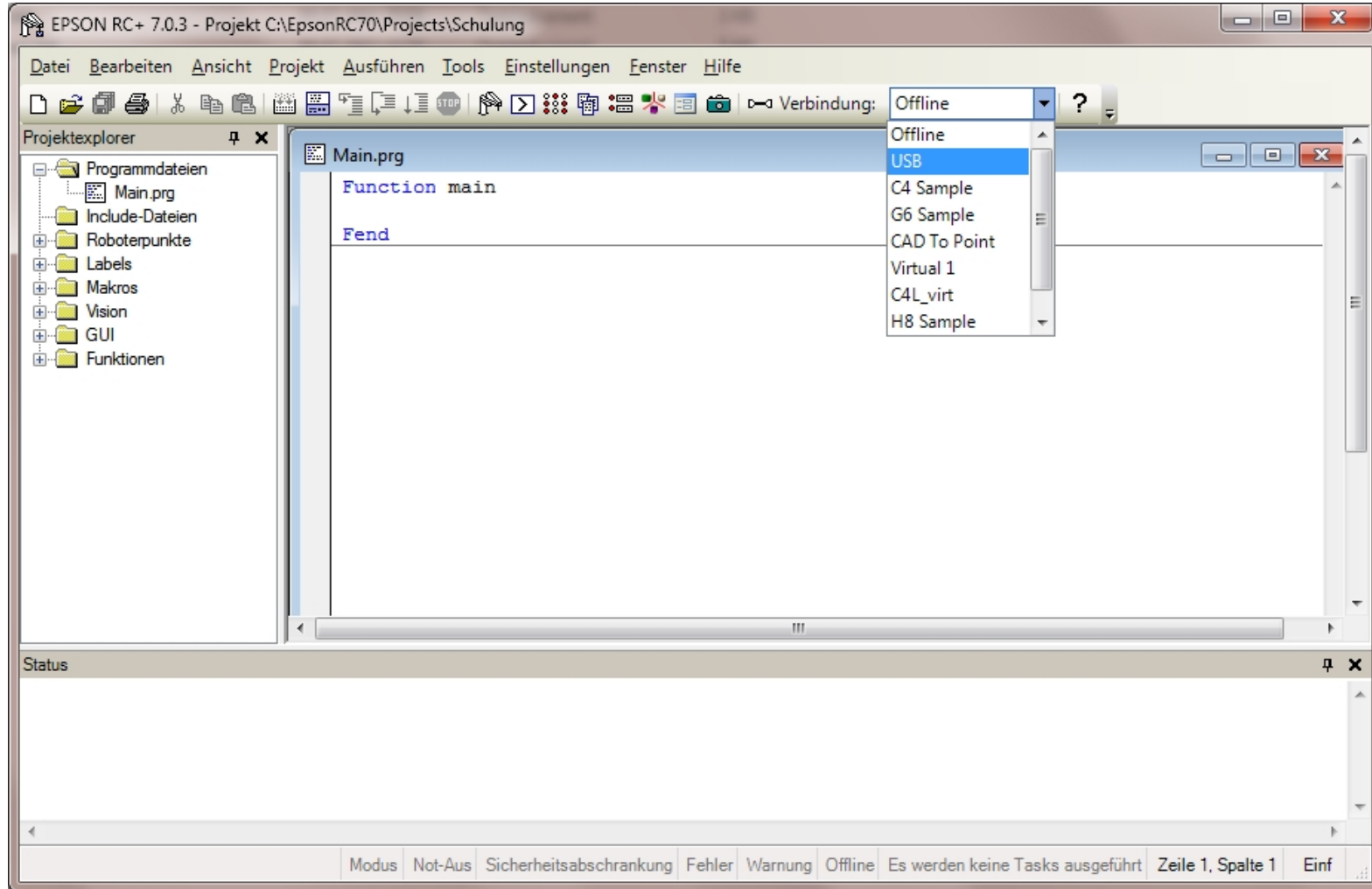


AUFGABENSTELLUNG

- Teachen Sie 2 oder mehr Raumpunkte (Roboterpositionen)
- Schreiben Sie ein Ablaufprogramm mit Jump & Go
- Geben Sie den Raumpunkten einen Namen (Label)
- Verwenden Sie die Label in Ihrem Ablaufprogramm
- Editieren Sie Punkte in der Punktedatei

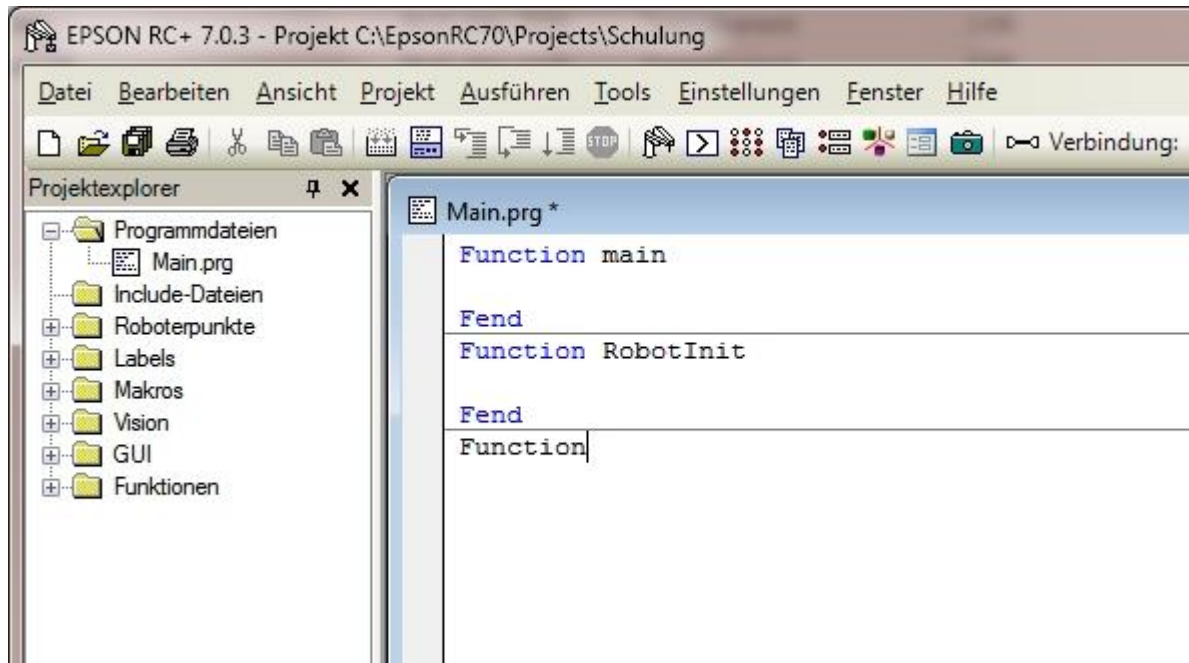
→ **Was wird hierzu benötigt?**

1. PROGRAMM USB-Verbindung



1. PROGRAMM

Programmstruktur



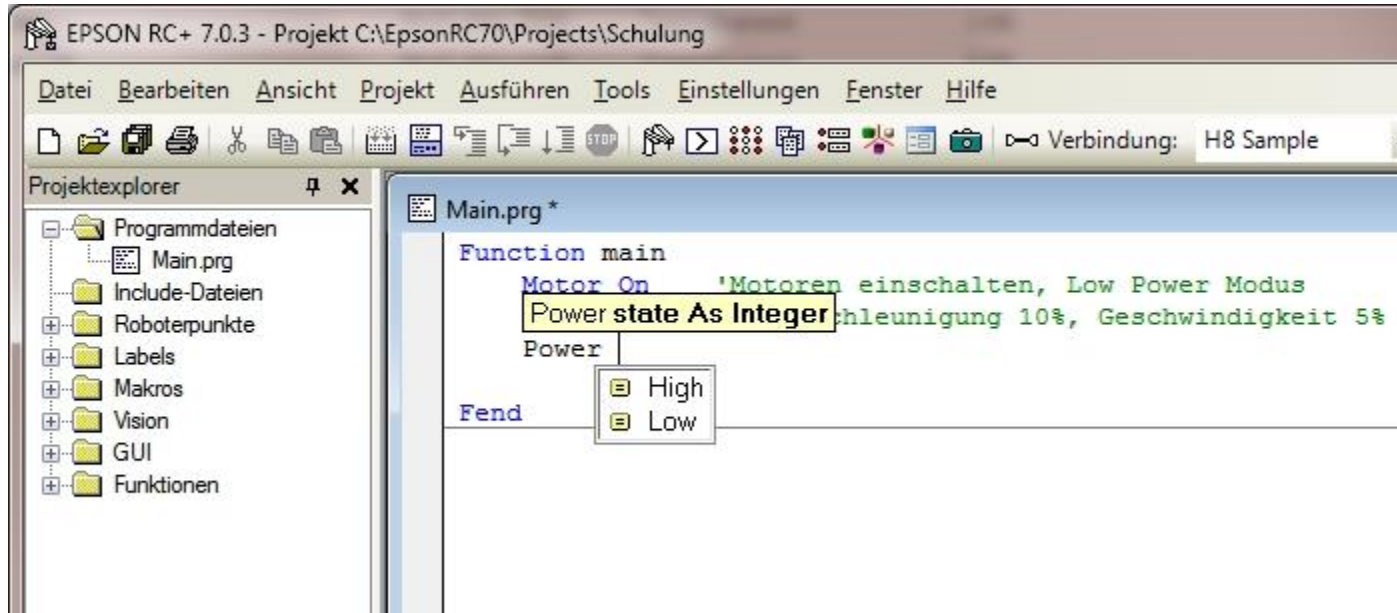
Die „Function main“ ist die Funktion, welche beim Automatikstart aufgerufen wird. Sie muss deshalb in jedem Projekt existieren.

Fehler 3050: Keine Main-Funktion definiert.



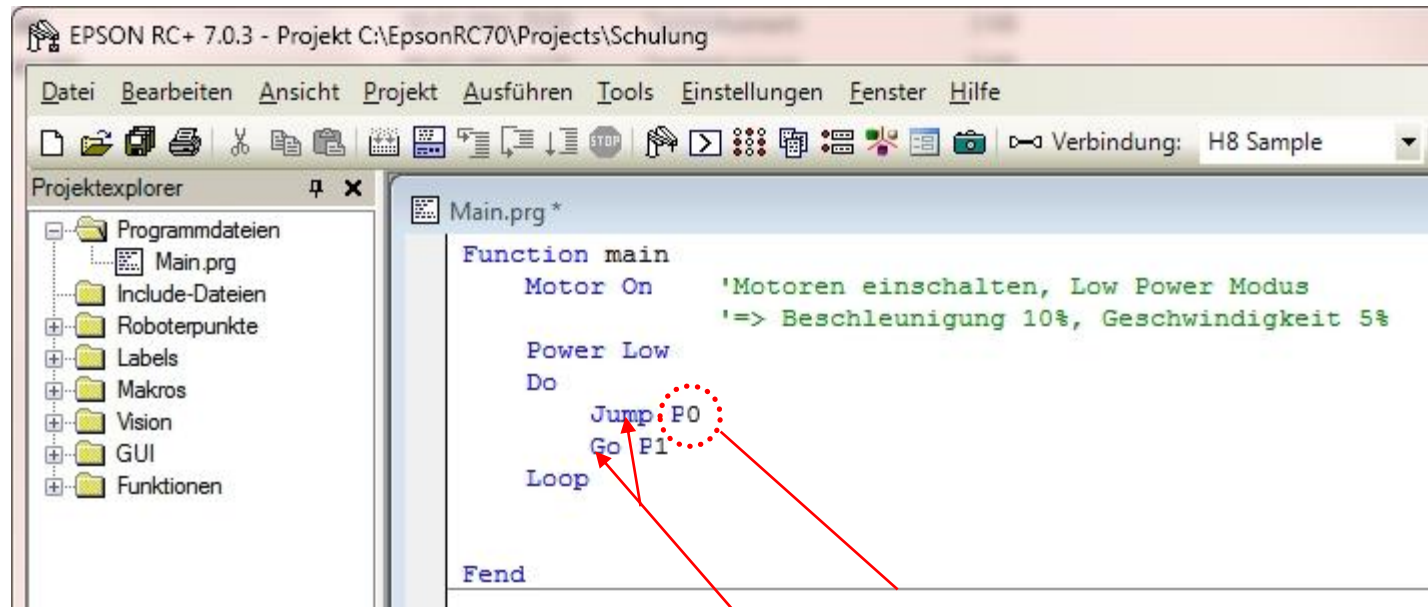
1. PROGRAMM

Keywords, Syntax-Hilfe, Kommentare



1. PROGRAMM

Einfacher Bewegungsablauf



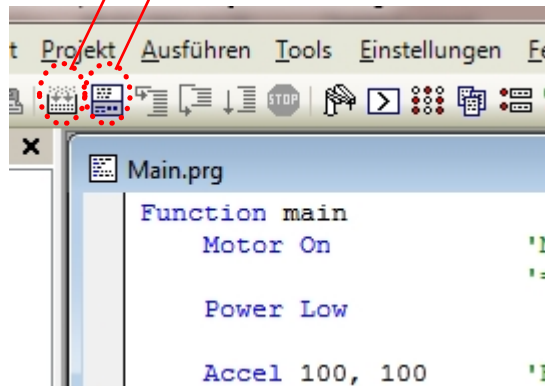
P0=Punkt mit der Nummer 0

Bewegungsbefehle

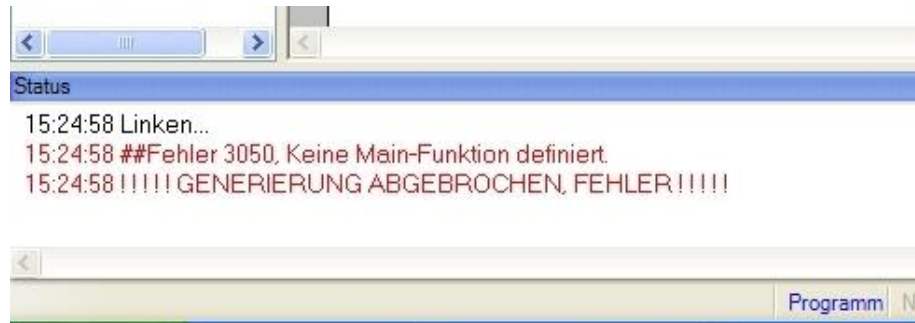
1. PROGRAMM

Kompilieren & Starten

Projekt generieren (STRG+B)
Run-Fenster öffnen (F5)



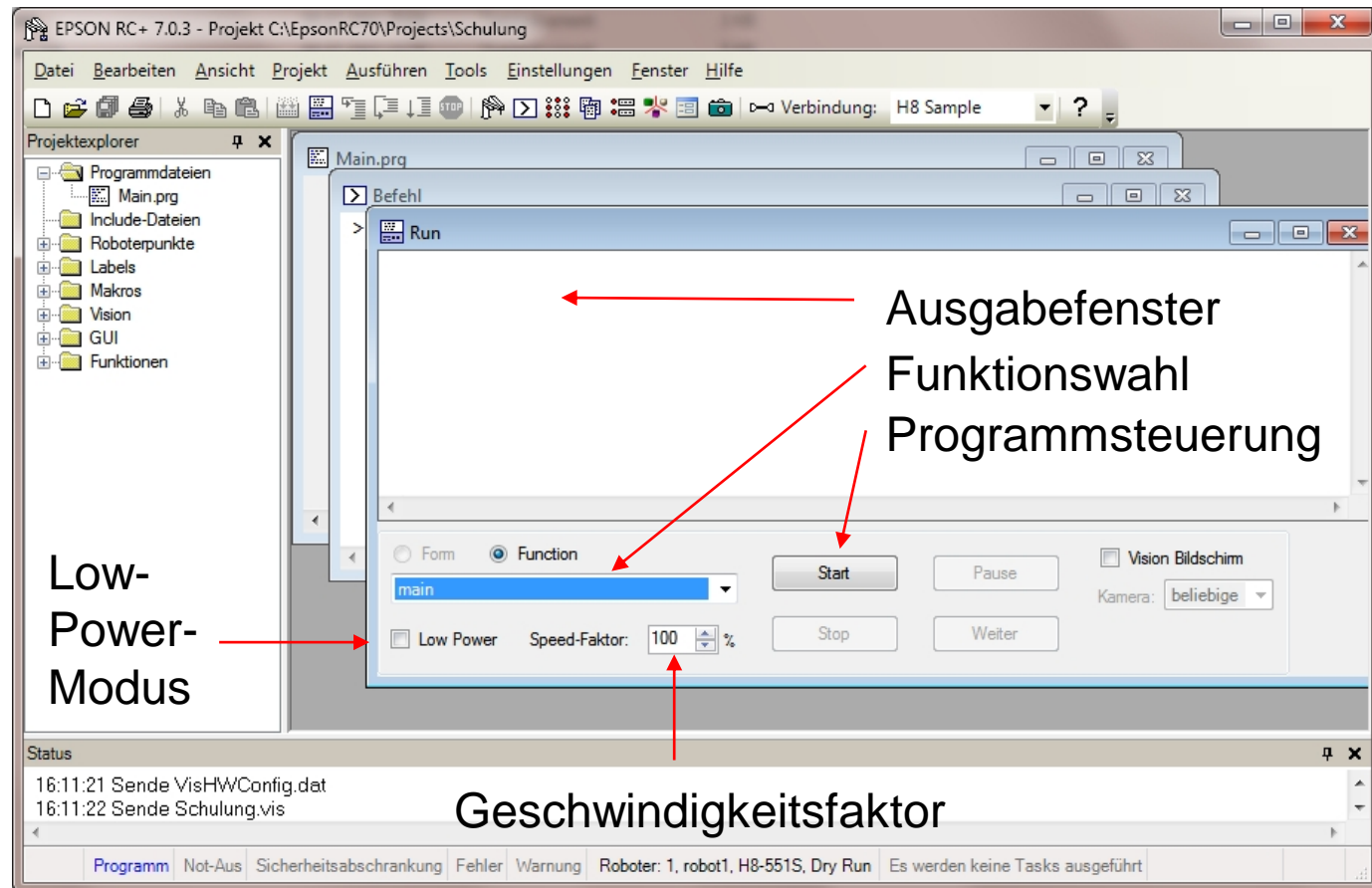
fehlerfreie Generierung



fehlerhafte Generierung

1. PROGRAMM

Kompilieren & Starten



GRUNDBEFEHLE

GRUNDBEFEHLE

Befehl: Motor ON / Motor OFF

The screenshot displays the Epson RC+ software interface, which is divided into two main windows: the Robotermanager (left) and the Main.prg editor (right).

Robotermanager (Left Window):

- Schaltpult:** A sidebar menu with options: Einrichten, Punkte, Arch, Locals, Tools, Arms, ECP, Boxen, Planes, Weight, Inertia, XYZ- Limits, Range, and Home-Konfig.
- Roboter:** A dropdown menu showing "1. robot1, H8-551S".
- Status:** Displays "Not-Aus: NICHT BETÄTIGT" and "Sicherheitsabschrankung: GESCHLOSSEN".
- Motoren:** A section containing two buttons: "MOTOR OFF" (grey) and "MOTOR ON" (yellow). This section is highlighted with a blue border.
- Motorleistung:** A section containing two buttons: "POWER LOW" (green) and "POWER HIGH" (grey).
- Servos frei:** A list of servos (J1, J2, J3, J4) with checkboxes.

Main.prg Editor (Right Window):

- Project Explorer:** A tree view showing the project structure: Program Files, Main.prg, Include Files, Robot Points, Labels, Macros, Vision, GUI, Force Control, Force Guide, and Functions.
- Main.prg *:** The main editor window showing the following code:

```
Function main
    Motor On
    Motor Off
End
```

GRUNDBEFEHLE

Befehl: Power Low / Power High

The screenshot displays the Robot Manager software interface. On the left, the 'Schaltpult' (Control Panel) is visible, featuring a list of functions: Einrichten, Punkte, Arch, Locals, Tools, Arms, ECP, Boxen, Planes, Weight, Inertia, XYZ- Limits, Range, and Home-Konfig. The 'Motorleistung' (Motor Power) section is highlighted with a blue box, containing two buttons: 'POWER LOW' (green) and 'POWER HIGH' (grey). The 'Status' section shows 'Not-Aus: NICHT BETÄTIGT' and 'Sicherheitsabschrankung: GESC'. The 'Main.prg' code editor on the right shows the following code:

```
Function main
    Power Low
    Power High
Fend
```

GRUNDBEFEHLE

Befehl: Reset

EPSON RC+ 7.5.4A R1 - Project C:\EpsonRC70\Projects\Empty_Files

File Edit View Project Run Tools Setup Window Help

Project Explorer

- Program Files
- Main.prg
- Include Files
- Robot Points
- Labels
- Macros
- Vision
- GUI
- Force Control
- Force Guide
- Functions

Main.prg *

```
Function main  
  
Reset  
  
Fend
```

Robotermanager

Schaltpult

Einrichten
Punkte
Arch
Locals
Tools
Arms
ECP
Boxen
Planes
Weight
Inertia
XYZ- Limits
Range
Home-Konfig

Roboter: 1, robot1, H8-551S

Status

Not-Aus: NICHT BETÄTIGT
Sicherheitsabschränkung: GESCHLOSSEN

Motoren: ON
Power: LOW

Motoren

MOTOR OFF MOTOR ON

Motorleistung

POWER LOW POWER HIGH

Servos freischalten

☐ J1
☐ J2
☐ J3
☐ J4

Servos frei
Servos ein

Reset
Home

FAHRBEFEHLE

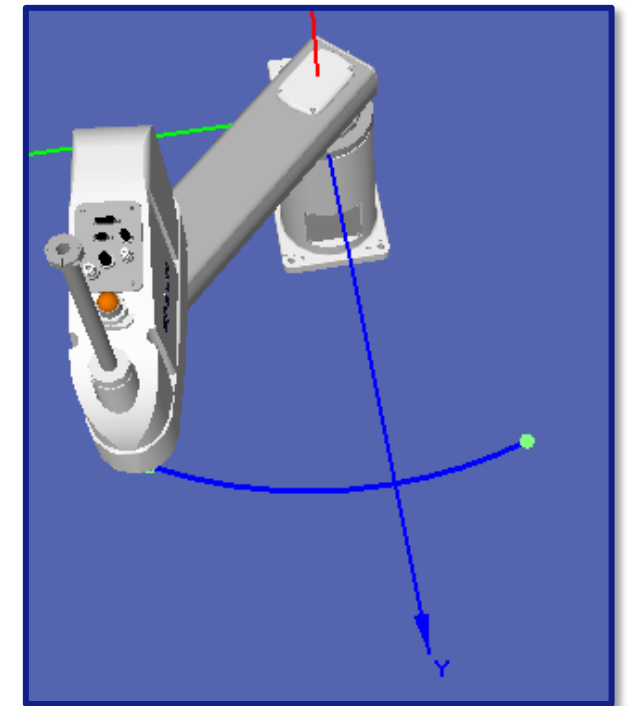
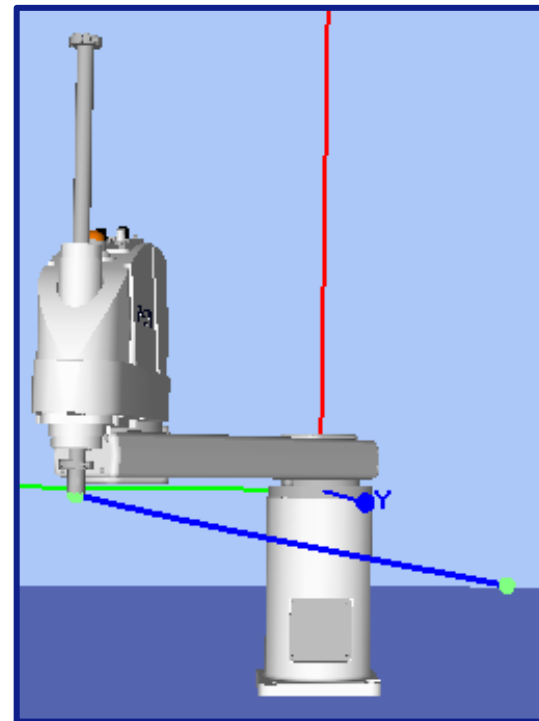
FAHRBEFEHLE

Befehl: Go

- Direkte Fahrt zwischen zwei Punkten
- Alle vier Achsen (Scara) bzw. sechs Achsen (ProSix) starten gleichzeitig und kommen gleichzeitig an der Zielposition an.
- Die Fahrt verläuft nicht direkt in einer geraden Bahn, sondern entlang eines Kreisbogens.

SYNTAX

Go P1

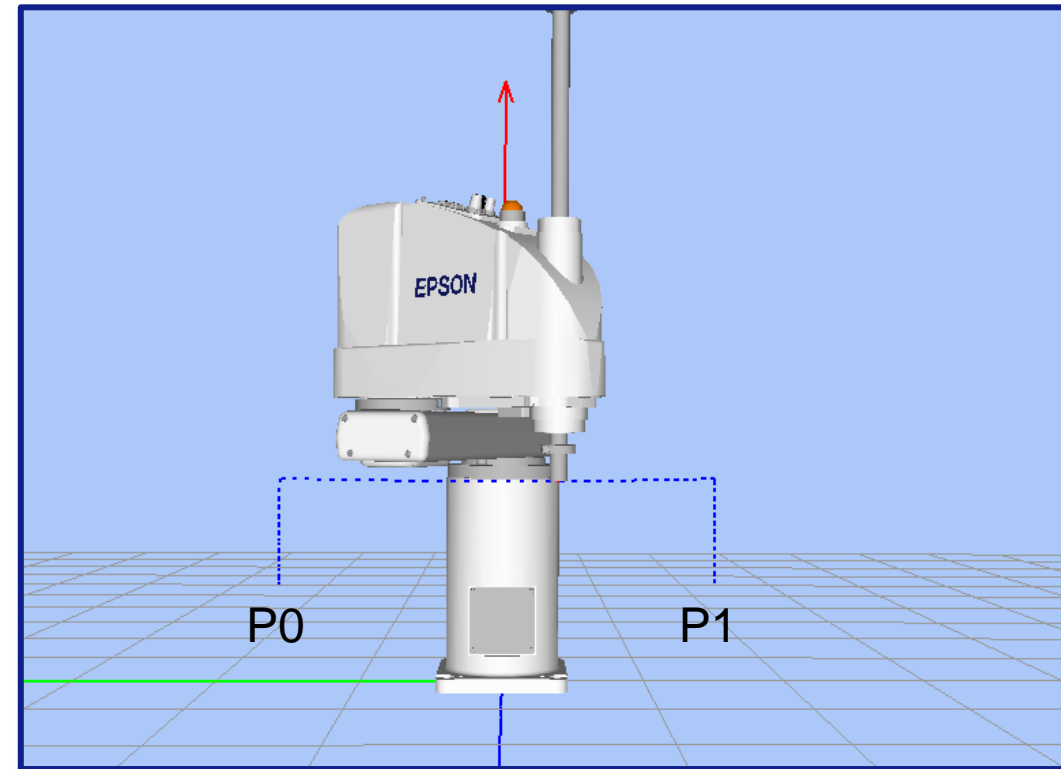


Bei Ausführung des JUMP-Befehls wird zuerst die Z-Achse auf Pos. Z(0) gefahren, bevor der Roboter mit der horizontalen Bewegung beginnt.

Der JUMP gehört (genau wie der GO Befehl) auch zur Gruppe der PTP-Bewegungsbefehle (= Point to Point).

SYNTAX

Jump P₁



Aufgabenstellung: 1. Ablaufprogramm



AUFGABENSTELLUNG

- Teachen Sie 2 oder mehr Raumpunkte (Roboterpositionen)
- Geben Sie den Raumpunkten einen Namen (Label)
- Schreiben Sie ein Ablaufprogramm mit Jump & Go
- Verwenden Sie die Label in Ihrem Ablaufprogramm
- Editieren Sie Punkte in der Punktedatei (Re-Teaching)

→ **Viel Erfolg**

OPTIMIERUNG

Der ARCH-Befehl definiert die lineare Strecke in mm, bevor der Roboter in die optimierte Bewegung übergeht.

SYNTAX

1. Definition erforderlich:

Arch *archNr, S1, S2*

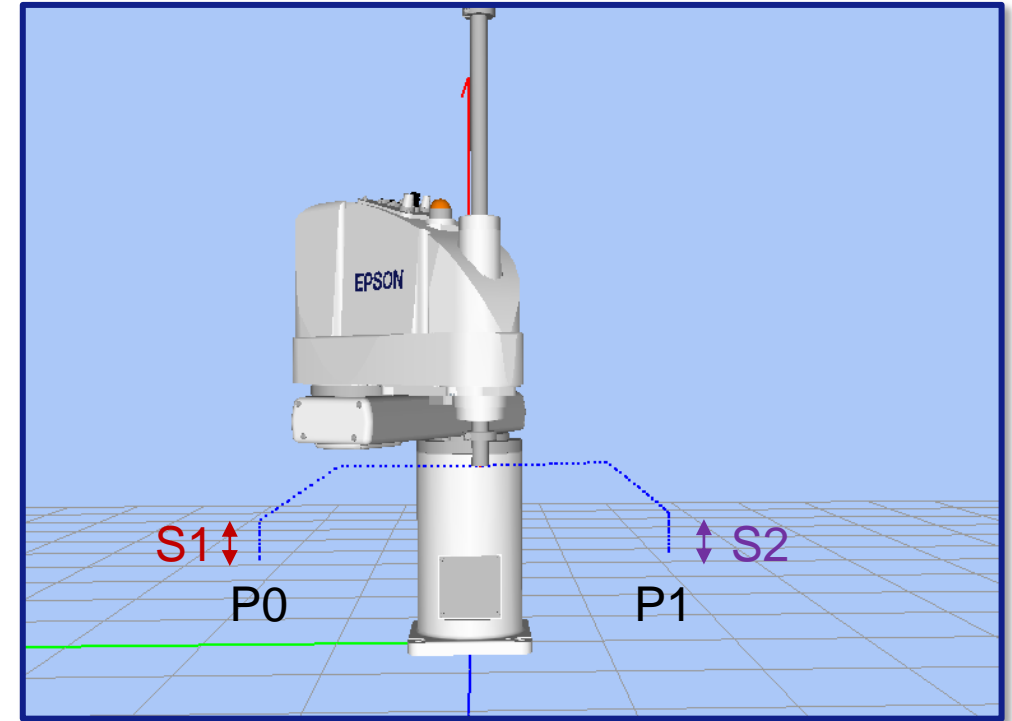
2. Anwendung:

Jump **P1** **C***archNr*

BEISPIEL-SYNTAX

Arch 0, 30, 50

Jump **P1** **C**0



S1 = linearer Weg Hubstrecke

S2 = linearer Weg Absenkstrecke

OPTIMIERUNG

Jump mit Option LIMZ

LIMZ begrenzt die Höhe der Z-Achse. Anwendung wahlweise als
Option hinter einem Fahrbefehl oder als eigenständigen
Befehlsaufruf.

SYNTAX

(globale Definition → gültig, bis wieder ein neues LIMZ definiert wird):

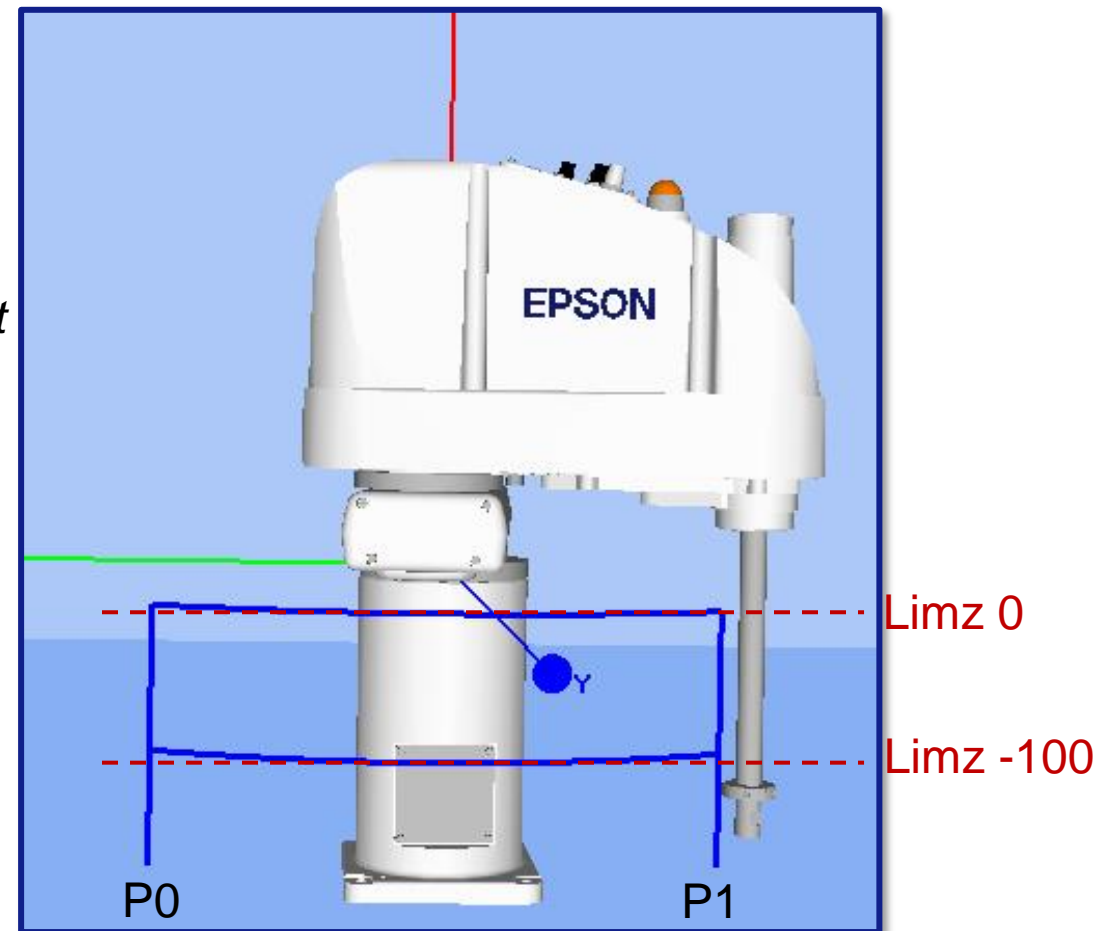
Limz *zlimit*

Alternativ-Syntax, falls das Limz nur für einen bestimmten
Jump
verwendet werden soll:

Jump P1 Limz *zlimit*

BEISPIEL-SYNTAX

Jump P1 Limz -100



Sofern der **LIMZ Befehl** zum Einsatz kommt, sind **folgende Voraussetzungen** nötig, damit keine Fehlermeldung auftritt:

- 1.) die aktuelle Position der Kugelrollspindel muss sich unterhalb der mit LIMZ eingestellten Grenze befinden
- 2.) die Zielposition muss sich unterhalb der mit LIMZ eingestellten Grenze befinden.

Der LIMZ Befehl fungiert wie eine Art „Software-Endschalter“ für die Z-Höhe.



LIMZ & ARCH können beliebig kombiniert werden !

Aufgabenstellung:

1. Erweiterung



AUFGABENSTELLUNG

- Optimieren Sie Ihre Roboterbewegungen.
- Verwenden Sie die Befehle in den verschiedenen Möglichkeiten:
 - Limz (global)
 - Limz („lokal“)
 - Arch

→ **Viel Erfolg**

Aufgabenstellung: PICK & PLACE



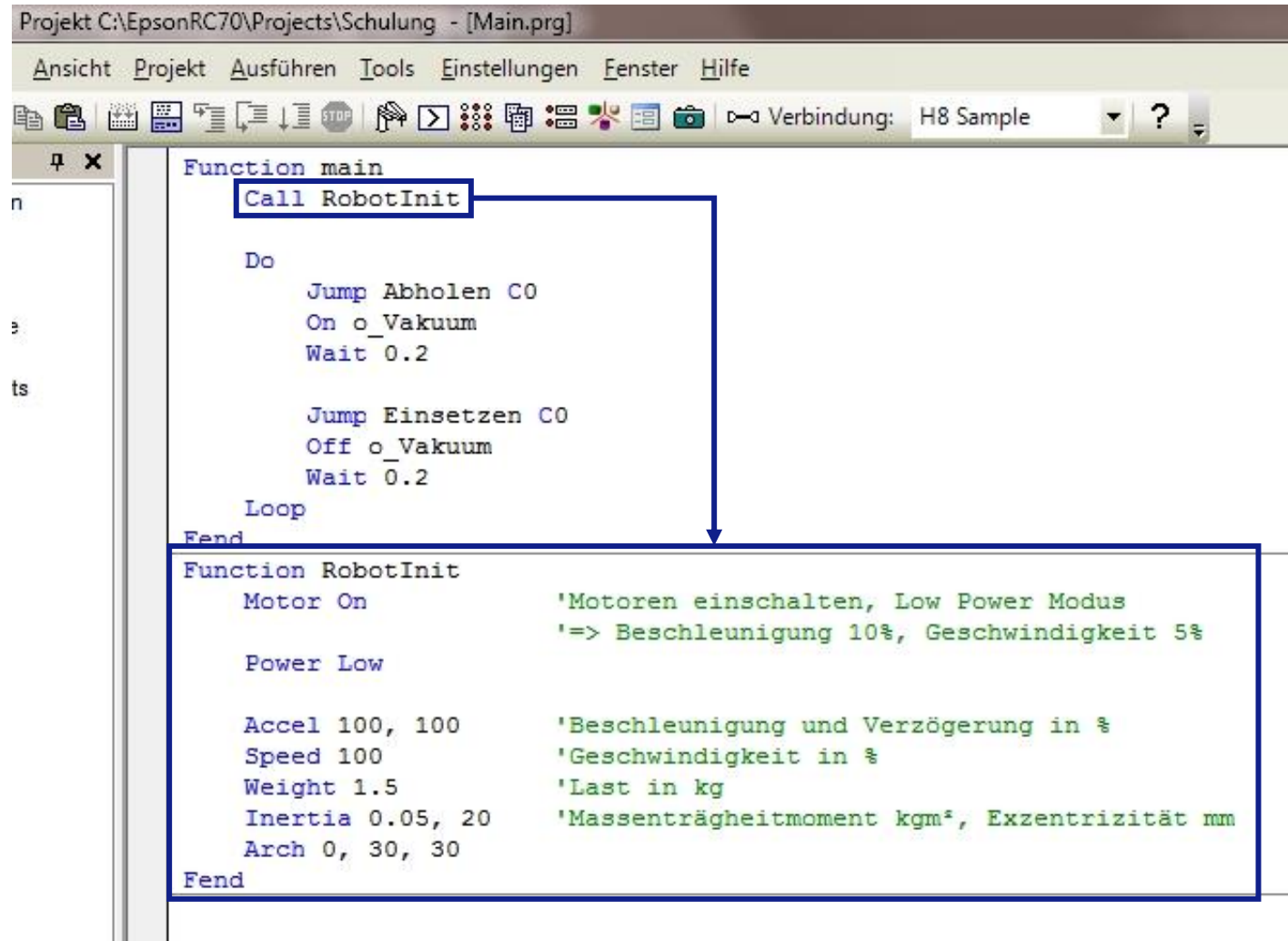
AUFGABENSTELLUNG

- Verwenden Sie eine Grundposition
(Für Start & Ende des Ablaufs)
- Abholung eines Produktes an Rutsche 1 oder 2
- Ablegen dieses Produktes auf der Standardpalette
- Verwenden Sie Till & Sense

→ **Was wird hierzu benötigt?**

Programmstruktur

Vom Beginn an ein bisschen Struktur



The screenshot shows the Epson RC70 programming software interface. The title bar indicates the project path: 'Projekt C:\EpsonRC70\Projects\Schulung - [Main.prg]'. The menu bar includes 'Ansicht', 'Projekt', 'Ausführen', 'Tools', 'Einstellungen', 'Fenster', and 'Hilfe'. The toolbar contains various icons for file operations and execution. The main window displays the code for 'Function main' and 'Function RobotInit'. A blue box highlights the 'Call RobotInit' command in the 'main' function, with a blue arrow pointing to the 'RobotInit' function definition below. The 'main' function code is as follows:

```
Function main
  Call RobotInit

  Do
    Jump Abholen C0
    On o_Vakuum
    Wait 0.2

    Jump Einsetzen C0
    Off o_Vakuum
    Wait 0.2
  Loop
Fend
```

The 'RobotInit' function code is as follows:

```
Function RobotInit
  Motor On          'Motoren einschalten, Low Power Modus
                   '=> Beschleunigung 10%, Geschwindigkeit 5%

  Power Low

  Accel 100, 100    'Beschleunigung und Verzögerung in %
  Speed 100         'Geschwindigkeit in %
  Weight 1.5        'Last in kg
  Inertia 0.05, 20  'Massenträgheitsmoment kgm², Exzentrizität mm
  Arch 0, 30, 30
Fend
```

Mit dem „Call“ Befehl, können Unterfunktionen aufgerufen werden.

Strukturieren Sie Ihr Programm mit Unterfunktionen.

*Hinweis:
Der Call Befehl ist nicht zwingend erforderlich.*

WEITERE BEFEHLE

Wait

Mit dem WAIT-Befehl haben Sie die Möglichkeit die Verarbeitung Ihres Programms für eine bestimmte Zeit, oder bis ein gewünschter Status eingetreten ist, anzuhalten.

Wait 0.2 `Verzögerung in Sekunden = 0,2s

BEISPIEL:

...

Jump *pAbholen*

Wait 0.2

Jump *pAblegen*

Wait 0.2

...

STANDARDMÄßIG VORHANDEN SIND...

24 Eingänge
(Nummer 0 bis 23)

16 Ausgänge
(Nummer 0 bis 15)

MÖGLICHKEITEN ZUR ERWEITERUNG DER I/O-KOMMUNIKATION:

Diskret:

128 E/A von Nr. 64 bis 191 (bei Verwendung aller 4 möglichen Erweiterungskarten)
oder als 16 Bytes von Nr. 8 bis 23 nutzbar
oder als 8 Worte von Nr. 4 bis 11 nutzbar

Feldbus: (z.B. Profibus: 32 Bytes / ProfiNet: 256 Bytes)

256 E/A von Nr. 512 bis 767
oder als 32 Bytes von Nr. 64 bis 95 nutzbar
oder als 16 Worte von Nr. 32 bis 47 nutzbar

ACHTUNG: Bei Ein- und Ausgängen könnte es doppelten Verwendungen kommen !!!

Ein- & Ausgänge können sowohl Bit- als auch Byte- und Wortweise gleichzeitig eingelesen oder geschrieben werden.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte:	Byte 1								Byte 0							
Wort:									Wort 0							

Zum Beispiel ist das Bit 9 auch in Byte 1, sowie in Wort 0 enthalten.

I/O-KOMMUNIKATION

Hand I/O T-Serie



Im Vergleich zu anderen Roboterserien, verfügt die T-Serie über sogenannte Hand I/O:

Hand Inputs

6 Pins

Bit-Adresse 18 – 23

Hand Outputs

4 Pins

Bit-Adresse 12 – 15

(non-polar, können sowohl als PNP oder NPN verdrahtet werden)

SPANNUNGSVERSORGUNG

Internes Netzteil T3: 24V max. 500mA (für alle Hand-I/O's)

Internes Netzteil T6: 24V max. 700mA (für alle Hand-I/O's)

→ max. jedoch nur **100mA pro Ausgangskanal**

Der Befehlssatz für die Standard-I/O ist auch für die Hand I/O zu nutzen.
Lediglich die Bit-Adressen unterscheiden sich.

The following is pin assignments of Hand I/O connector (D-sub 15 pin male).

Pin No.	Signal Name	Pin No.	Signal Name
1	Input No.18	9	Input No.19
2	Input No.20	10	Input No.21
3	Input No.22	11	Input No.23
4	Input common No.18 to 23	12	Not Used
5	+24V	13	GND
6	Output No.12	14	Output No.13
7	Output No.14	15	Output No.15
8	Output common No.12 to 15		

Connector	Standard
Hand I/O connector	D-Sub 15 pin female (manipulator side) D-Sub 15 pin male (cable side)

*I/O connector is included with shipment.

**ACHTUNG: Die Hand-I/O Anschlüsse
sind nicht kurzschlussfest!**



EPSON®

AUSGANGSBEFEHLE (BIT-OPERATIONEN)

On 5	`Schaltet den Ausgang 5 ein
Off 6	`Schaltet den Ausgang 6 aus
MemOn 6	`Schaltet das Merkerbit 6 ein
MemOff 6	`Schaltet das Merkerbit 6 aus
On 5, 0.2	`Schaltet den Ausgang 5 für 0.2s ein
Off 6, 0.3	`Schaltet den Ausgang 6 für 0.3s aus

SEQUENTIELLE ODER PARALLELE VERARBEITUNG

Off 6, 0.3, 1	`1= sequentielle Bearbeitung (Standard)
On 5, 0.2, 0	`0= Parallelbearbeitung

AKTUELLER STATUS DES AUSGANG 5 EINLESEN

If Oport(5) = on **Then...**

ANWENDUNGSBEISPIEL

Beide Programmteile erzielen den identischen Effekt: das Vakuumventil schaltet ab und der Greifer wird 0,05 Sek. lang belüftet bevor der Jump zu P20 ausgeführt wird.

```
Off outVakuum
On outBlow
Wait 0.05
Off outBlow
Jump P20
```

```
Off outVakuum
On outBlow, 0.05
Jump P20
```

BITTE BEACHTEN SIE!

Der Zustand der Ausgänge bleibt bei Programmstopp / -start erhalten.
Unter **Not-Aus** oder **Controller Neustart** wird der Zustand der Ausgänge auf **OFF** zurück gesetzt.

Bringen Sie daher Ihre benötigten Ausgänge in Ihre Initialisierungsroutine ein!

AUSGANGSBEFEHLE (BYTE-OPERATIONEN)

`Setzt das Ausgangsbyte 0 auf den dezimalen Wert 7

Out 0, 7

`Setzt das Merkerbyte 0 auf den dezimalen Wert 7

MemOut 0, 7

`Abfrage des Ausgangsbytes 0

If Out(0)= 7 **Then...**

AUSGANGSBEFEHLE (WORT-OPERATIONEN)

`Setzt das Ausgangswort 0 auf den dezimalen Wert 10000

OutW 0, 10000

`Setzt das Merkerwort 0 auf den dezimalen Wert 10000

MemOutW 0, 10000

`Abfrage des Ausgangswortes 0

If OutW(0)= 10000 **Then...**

EINGANGSBEFEHLE (BIT-OPERATIONEN)

If Sw(3) = On Then... `Eingang 3 = eingeschaltet ?

If Sw(4) = Off Then... `Eingang 4 = ausgeschaltet ?

If MemSw(3) = On Then... `Merkerbit 3 = gesetzt ?

EINGANGSBEFEHLE (BYTE- UND WORT-OPERATIONEN)

If In(0) = 5 Then... `Abfrage des Eingangsbytes 0

If InW(1) = 5 Then... `Abfrage des Eingangswort 1

If InReal(0) = 3.15 Then... `Abfrage des Eingangswort 0 im IEEE754 Real-Format

If MemIn(0) = 5 Then... `Abfrage des Merkerbyte 0

If MemInW(1) = 5 Then... `Abfrage des Merkerwort 1

I/O Verwendung

Warten auf einen Eingang

Wenn z.B. am Greifer ein Sensor verdrahtet ist, kann auch der Status des Sensors mit dem WAIT-Befehl verknüpft werden:

```
Wait Sw(1) = On           `Wartet auf Eingang 1 = ON
```

BEISPIEL: (WARTET DARAUF, DASS DER GREIFER GESCHLOSSEN HAT)

```
...  
  Jump pAbholen  
  On outGreifer  
  Wait Sw(inGreiferZu) = On           `Greifer geschlossen?  
  Jump pAblegen  
  Off outGreifer  
  Wait Sw(inGreiferZu) = Off          `Greifer offen?  
...
```

I/O Verwendung

Warten auf einen Eingang

Damit nicht unendlich lange auf den Eingang 1 gewartet wird, kann eine Zeitangabe als TimeOut definiert werden.

```
Wait Sw(1) = On, 0.5           `Wartet auf Eingang 1 = ON  
                                `max. Wartezeit: 0,5 Sek.
```

Es wäre nun interessant zu wissen, ob der Eingang oder der TimeOut gekommen ist. Dies ist mit folgender Abfrage möglich:

SYSTEMFLAG: TW

Das Systemflag TW meldet true, wenn der TimeOut abgelaufen ist, also die Bedingung der WAIT Anweisung nicht rechtzeitig eingetreten ist.

```
Wait Sw(inGreiferZu) = On, 0.5  
    If TW = True Then           `TW=TRUE -> TimeOut aktiv  
        Jump GrundPos          `Grundstellungsfahrt  
    Endif
```

I/O-KOMMUNIKATION

Befehlsübersicht

	BIT	BYTE	WORT	REAL
Eingänge:	Sw (1) = On	In (0)	InW (0)	InReal (0)
Ausgänge:	On 1 Off 1	Out 0, 0	OutW 0, 0	OutReal 0, 0.0
Merker:	MemSw (1) = On MemOn 1 MemOff 1	MemIn (0) MemOut 0, 0	MemInW (0) MemOutW 0, 0	

MÖGLICHE ZAHLENSYSTEME

&B (binär)
&H (hexadezimal)

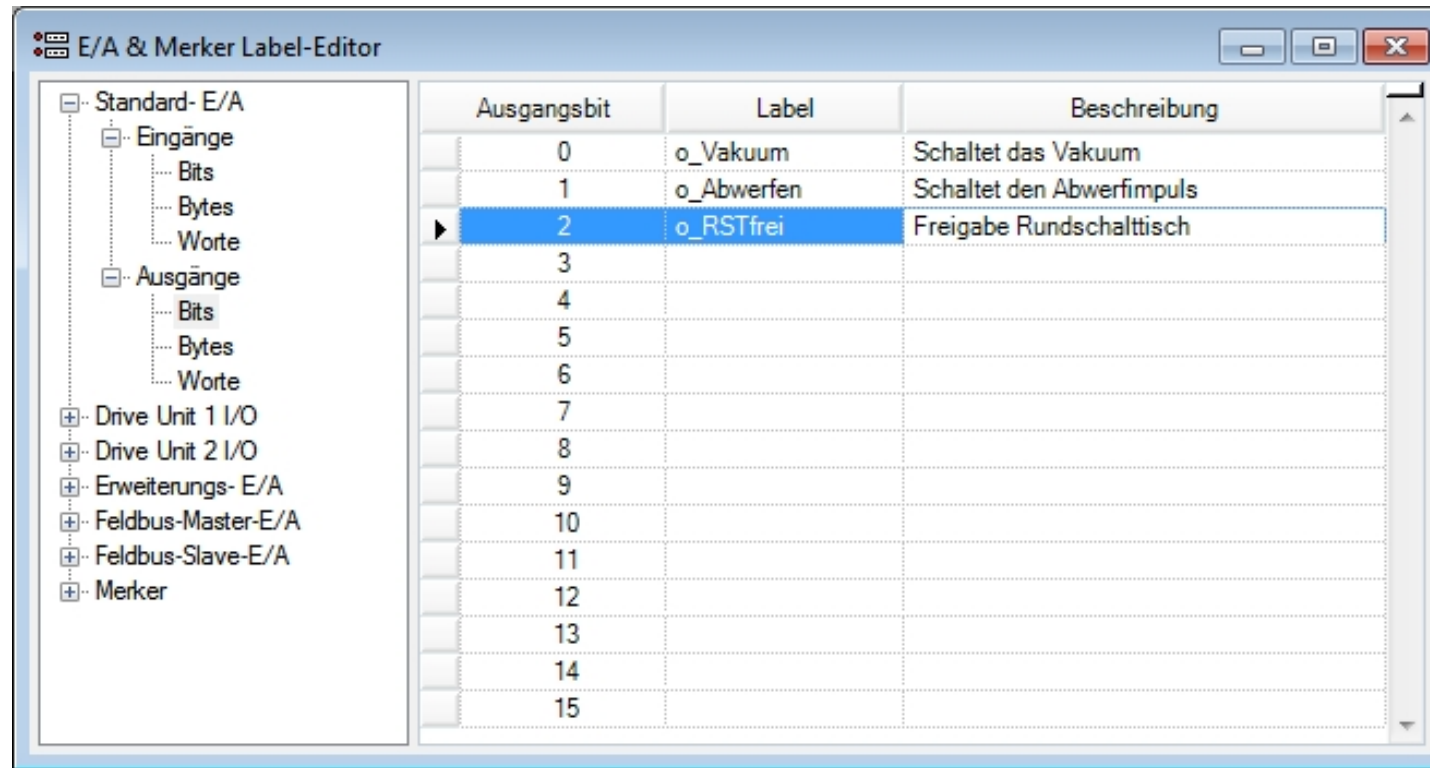
Beispiel: **Out** 0, &H7

I/O-KOMMUNIKATION

E/A-Monitor

Um Ein- und Ausgänge beschriften zu können, gibt es in der RC+ Software einen **E/A-Labeleditor**.

Menü „Tools“ – E/A-Labeleditor oder über das Icon

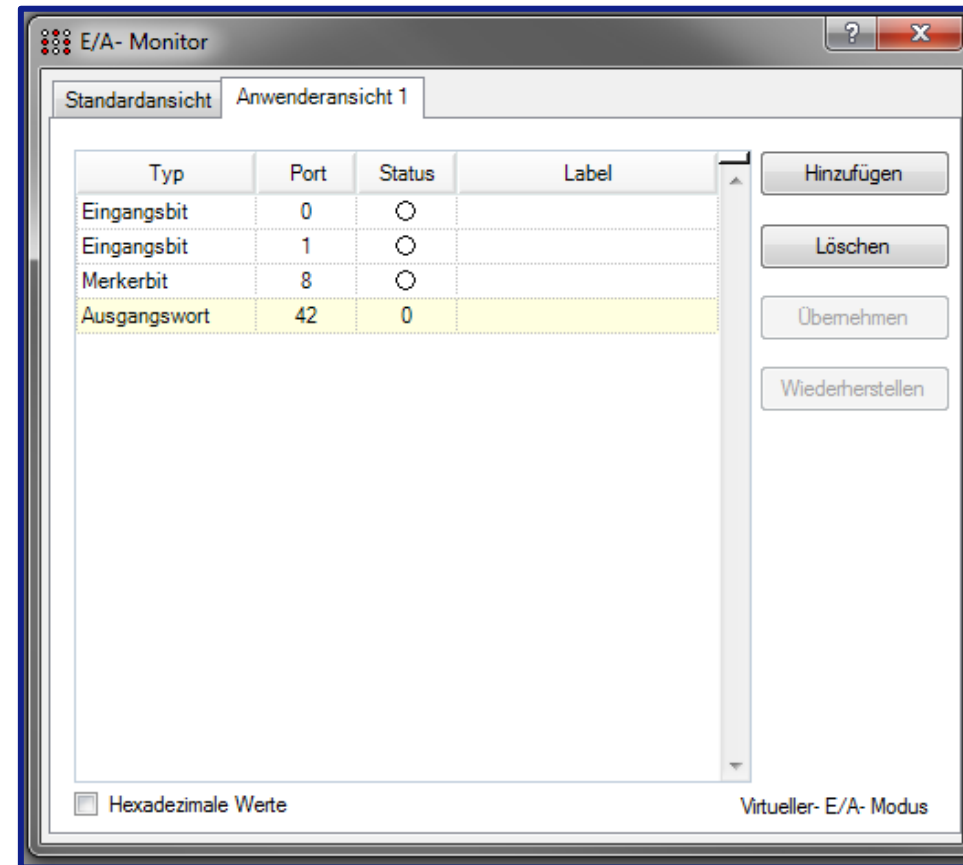
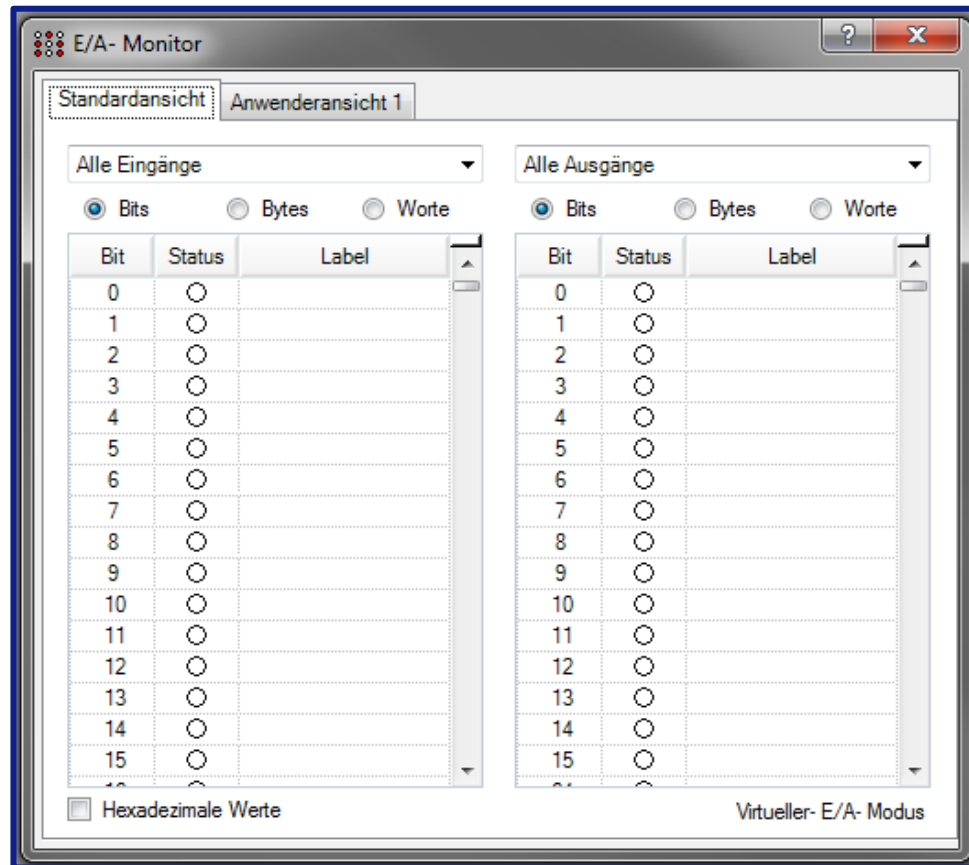


I/O-KOMMUNIKATION

E/A-Monitor

Um Ein- und Ausgänge beobachten und steuern zu können, gibt es in der RC+ Software einen **E/A-Monitor**.

Menü „Tools“ – E/A-Monitor oder über das Icon



OPTIMIERUNG

Durch ein Till, kann eine Bedingung gesetzt werden, mit der ein Bewegungsbefehl abgebrochen werden kann.

Beim Till wird die Bedingung dauerhaft während der Bewegung abgefragt.

SYNTAX

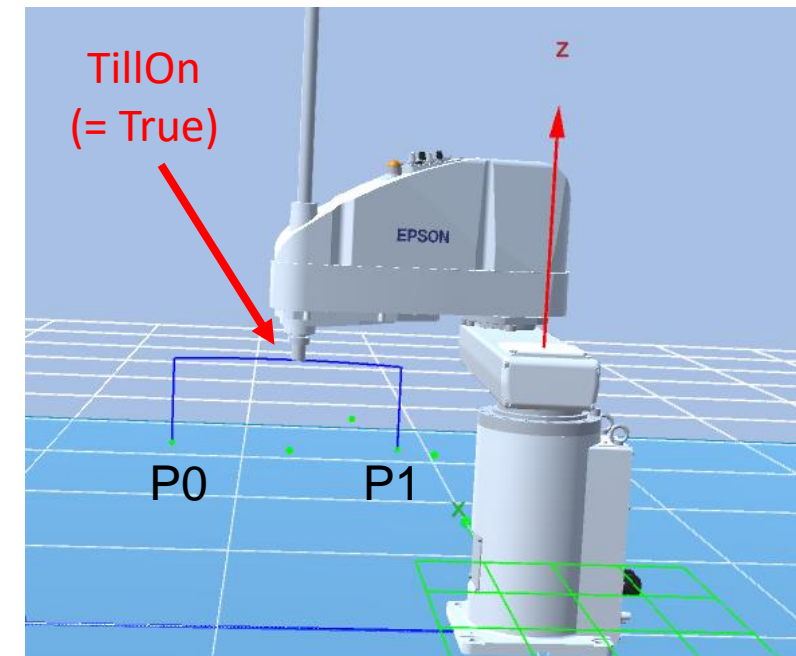
(globale Definition → gültig, bis wieder eine neue Till definiert wird):

Till „Bedingung“

BEISPIEL-SYNTAX

Till SW („Bit“)

Jump P1 Till



OPTIMIERUNG

Sense

Sense ist eine Bedingung, die bei einem Jump verwendet werden kann.

Wenn die Sense-Bedingung erfüllt ist, wird der Jump als „Beendet“ gewertet, sobald der Roboter über der Zielposition angekommen ist.

SYNTAX

(globale Definition → gültig, bis wieder eine neue Sense Bedingung definiert wird):

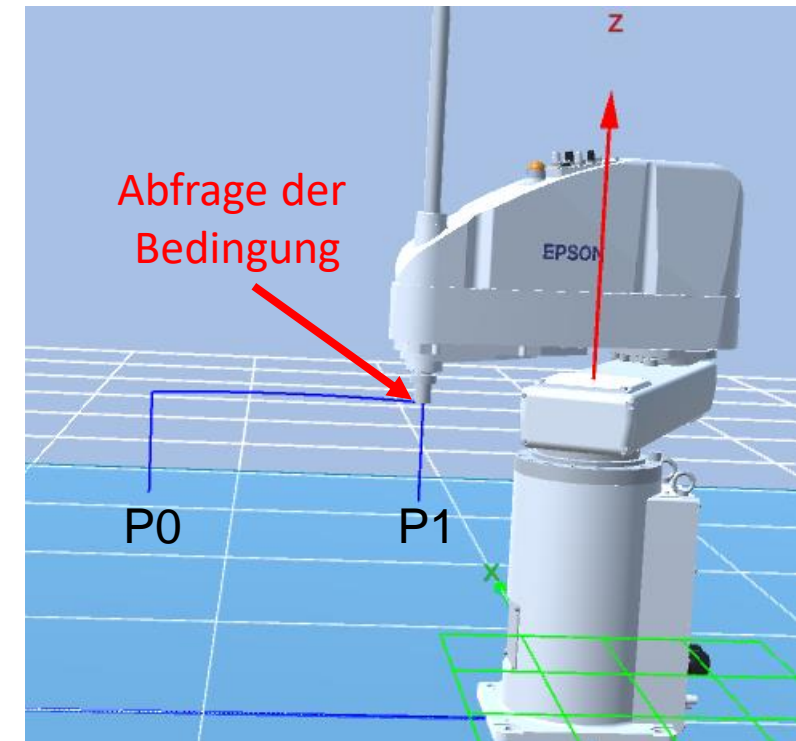
Sense „Bedingung“

Jump P1 Sense

BEISPIEL-SYNTAX

Sense SW ("Bit")

Jump P1 Sense



ACADEMY ROBOT 1 **LS6-602S mit RC700-A**

Ausgang 11	Vakuum
Ausgang 12	Druckluft

ACADEMY ROBOT 3 **LS6B-601S mit RC90B**

Ausgang 11	Vakuum
Ausgang 12	Druckluft

ACADEMY ROBOT 2 **G20-854S mit RC700-A**

Ausgang 11	Vakuum
Ausgang 12	Druckluft

ACADEMY ROBOT 4 **RS4-551S mit RC700-A**

Ausgang 512	Druckluft
Ausgang 513	Vakuum

G6-553S mit RC700-A

Ausgang 8	Vakuum
Ausgang 7	Druckluft

Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go <i>P_End</i>	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump <i>P_End</i>	Führt einen „Jump“ zum Punkt „P_End“ aus.	



AUFGABENSTELLUNG

- Verwenden Sie eine Grundposition
(Für Start & Ende des Ablaufs)
- Abholung eines Produktes an Rutsche 1 oder 2
- Ablegen dieses Produktes auf der Standardpalette
- Verwenden Sie Till & Sense

Viel Erfolg!

Aufgabenstellung: PICK & PLACE Erweiterung #1



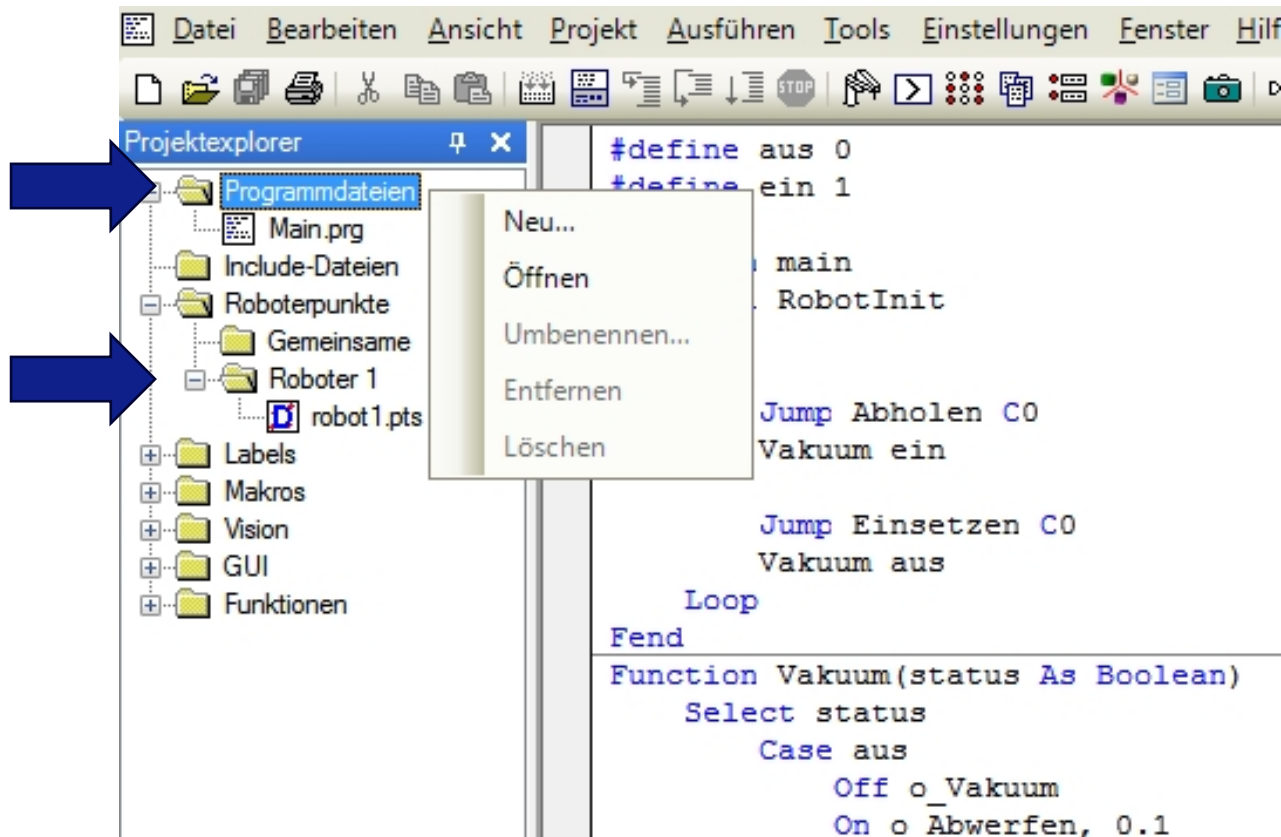
AUFGABENSTELLUNG

- Abholung von 10 Produkten an Rutsche 1
- Abholung von 10 Produkten an Rutsche 2
- Ablegen der Produkte in Palettenposition 1 bis 20

→ Was wird hierzu benötigt?

Projektexplorer

PRG-Dateien & Punktelisten

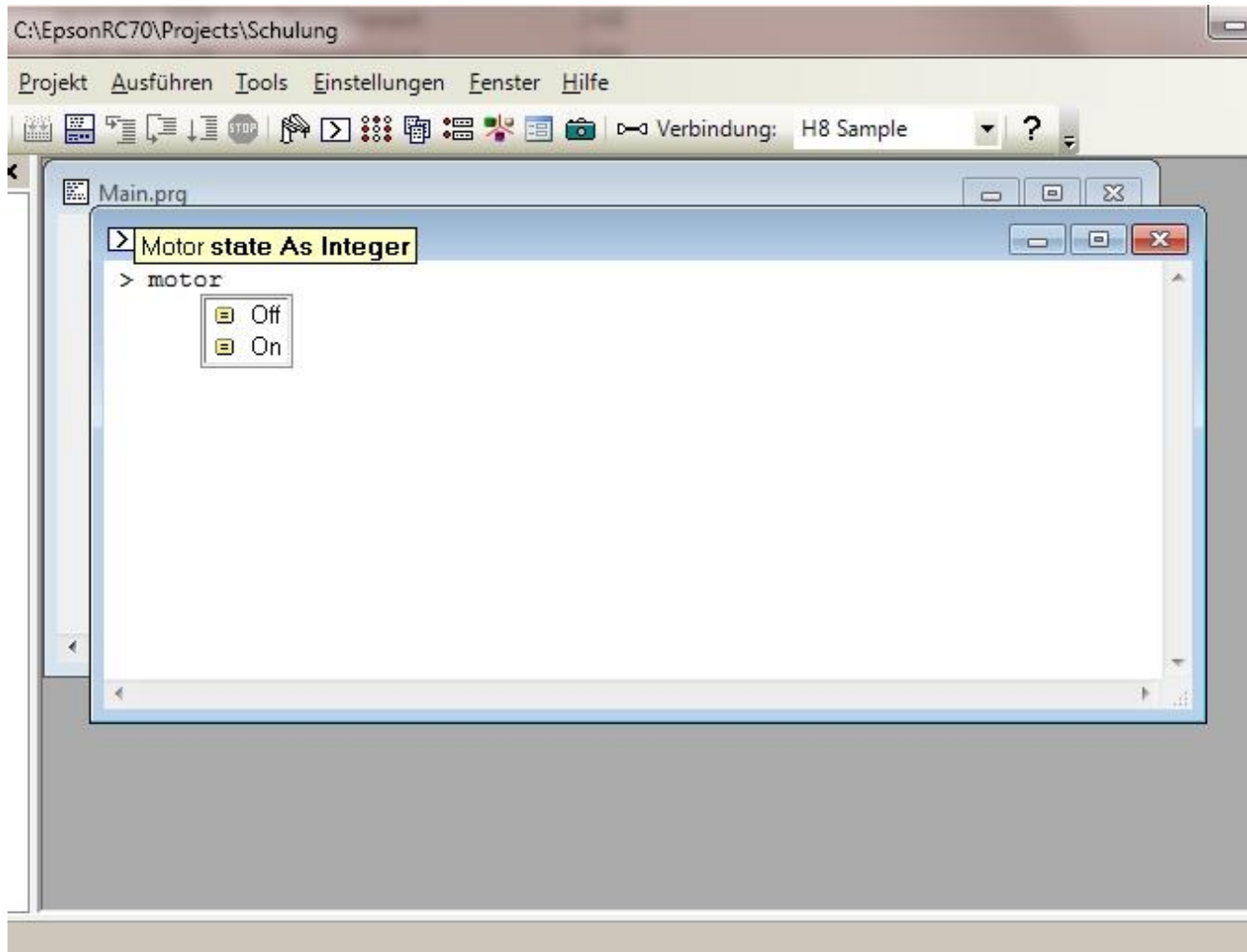


Mit einem „Rechtsklick“ auf den entsprechenden Menüpunkt können Sie neue .prg-Dateien (Programmdateien) oder .pts-Dateien (Punktelisten) im Projekt erstellen.



Befehlseingabefenster

Direkte Interaktion mit dem Roboter



Über das folgende Symbol:

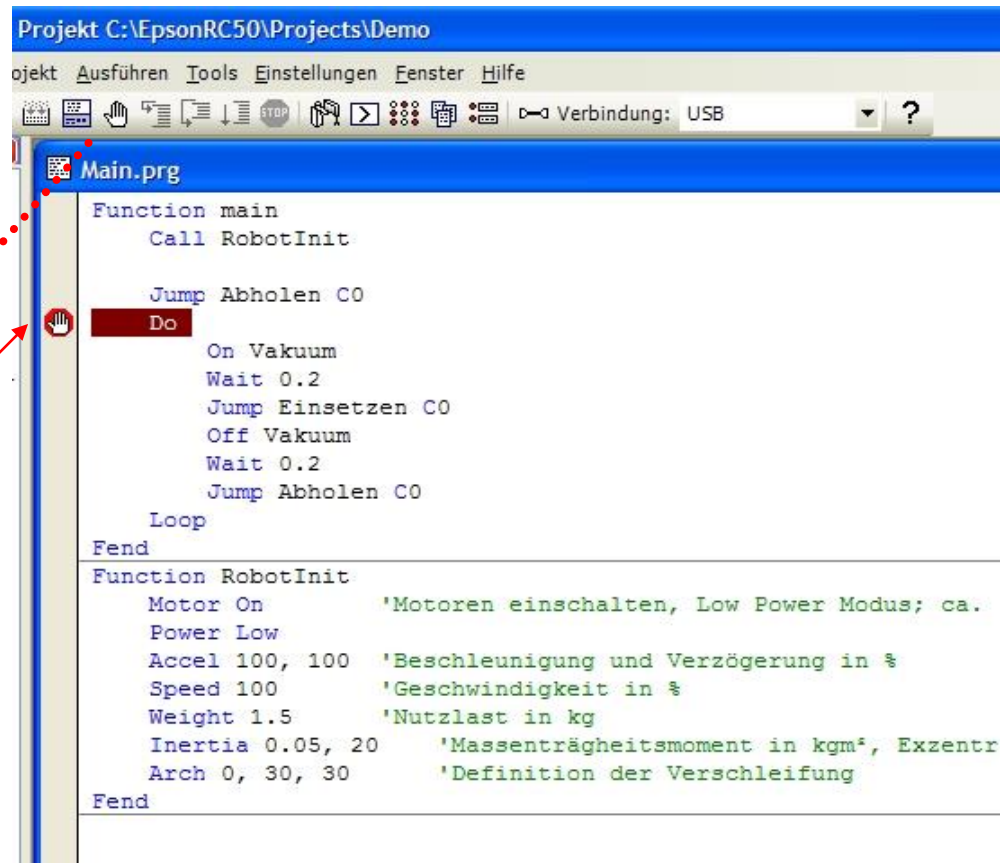


können Sie das
Befehlseingabefenster
öffnen.

Mit der Befehlseingabe
können Sie Befehle direkt
an den Roboter
übergeben.



Haltepunkt / Debugger



ein- und
ausschalten
oder
hier klicken

F10: Prozedurschritt

F11: Einzelschritt

F12: Bewegungsschritt

**F7: Fortsetzen (bis zum
nächsten Haltepunkt)**

Hinweis:

**Ein Haltepunkt kann
jederzeit gesetzt oder
gelöscht werden.**



Variablen & Arrays

Datentyp	Größe	Wertebereich
Boolean	2 Bytes	True oder False False = 0 / True = -1
Byte	2 Bytes	-128 ... +127
Integer	2 Bytes	-32768 ... +32767
Long	4 Bytes	-2147483648 ... +2147483647
Real	4 Bytes	-3.40E+38 ... 3.40E+38. (Auflösung: 6 Nachkommastellen)
Double	8 Bytes	-1.79E+308 ... 1.79E+308 (Auflösung: 14 Nachkommastellen)
String	256 Bytes	255 mögliche Zeichen + 1 Byte Längenangabe

Hier finden Sie Beispiele für die verschiedenen Variable-Deklarationen:

Function fInit

Integer LoopCount1, LoopCount2

String Name\$, PLC\$

Fend

`lokale Variablen

`lokale Variablen

Global Long Counter

Global Preserve Byte PalPos

Real InputValue

`globale Variable

`remanente Variable (immer global)

`Modulvariable

Function main

Integer LoopCount1, LoopCount2

String Name\$, PLC\$

Fend

`lokale Variablen

`lokale Variablen

VARIABLEN & DATENSTRUKTUREN

Array-Struktur

Es besteht die Möglichkeit, für **jeden Datentyp** eine Arraystruktur anzulegen. Der Syntax für die Deklaration und den Zugriff auf die Struktur sieht wie folgt aus:

```
Function main
```

```
Integer ProductID(9)
```

```
Integer A(2)
```

```
Integer B(2, 2)
```

```
Integer C(2, 2, 2)
```

```
Integer Zaehler
```

```
For Zaehler = 0 To 9
```

```
    ProductID(Zaehler) = Zaehler * 2
```

```
Next
```

```
End
```

10 Werte möglich:
0...9

1-dimensional			
Register	0	1	2
	0	25	6

2-dimensional			
Register	0	1	2
0	3	25	6
1	8	10	12
2	0	5	9

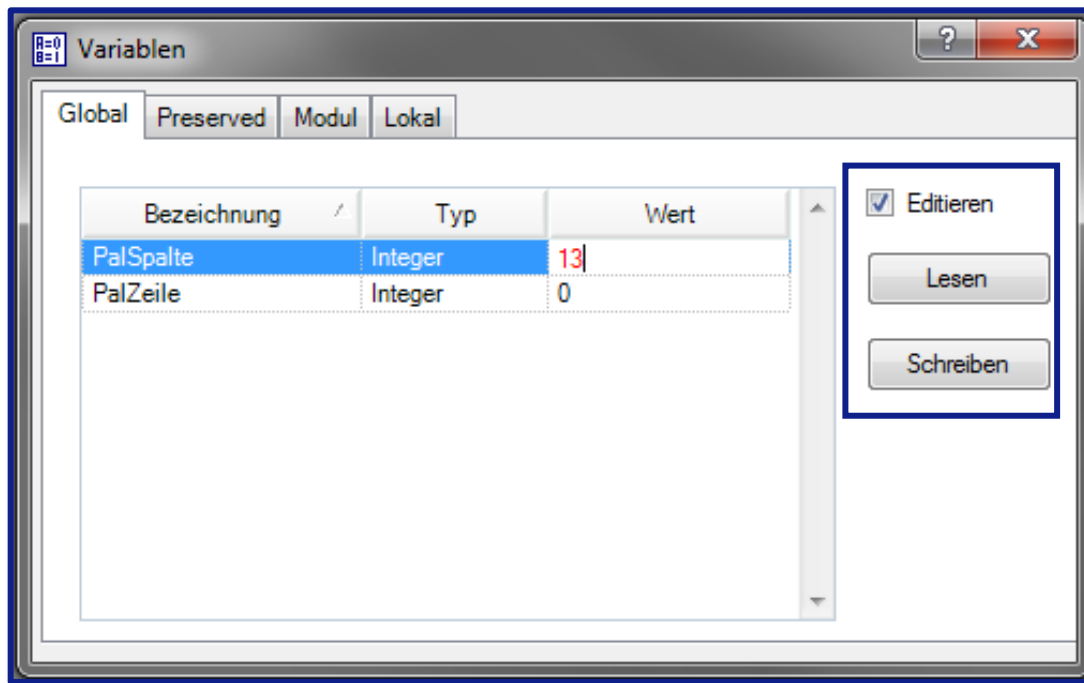
3-dimensional									
Register	0			1			2		
	0	1	2	0	1	2	0	1	2
0	5	2	3	15	56	29	81	22	6
1	4	6	8	33	18	7	36	45	12
2	10	9	0	1	90	25	59	51	9

Variablenübersicht

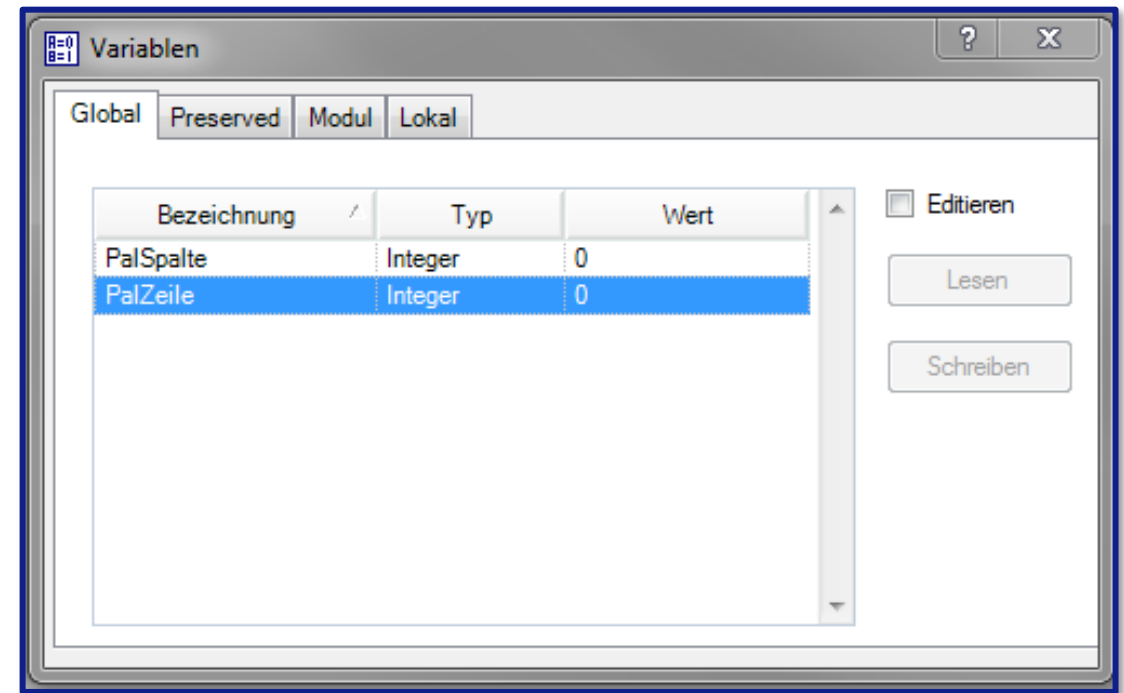
Editieren & Beobachten von Variablen

Unter dem Menüpunkt **Ausführen – Variablenübersicht** finden Sie Auflistung aller im Programm deklarierten Variablen, getrennt nach dem Gültigkeitsbereich.

Editieren:



Beobachten:



‘Beispiel 1

If Sw(5)= On Then

Jump Ablegen
C0

Endlf

‘Beispiel 2

If Sw(5)= On Then

Jump Ablegen **C0**

Else

Jump NIOAbwurf **C1**

Endlf

‘Beispiel 3

If Sw(5)= On Then

Jump Ablegen1 **C0**

Elseif Sw(6)= On Then

Jump Ablegen2 **C0**

Else

Jump NIOAbwurf **C1**

Endlf

‘Beispiel 1

Select *VarWert*

Case 1

Call UP1

Case 2

Call UP2

Send

‘Beispiel 2

Select On

Case Sw(5)

Jump Ablegen1 **C0**

Case Sw(6)

Jump Ablegen2 **C0**

Default

Jump NIOAbwurf **C1**

Send

‘Beispiel 3

Select True

Case Sw(5)= On And Sw(6)= Off

Jump Ablegen1 **C0**

Case Sw(5)= Off And Sw(6)= On

Jump Ablegen2 **C0**

Default

Jump NIOAbwurf **C1**

Send

PROGRAMMSTEUERUNG

Mögliche Verknüpfungsvarianten

Für die Programmsteuerung mittels **IF...THEN...ELSE** können die verschiedensten Verknüpfungsvarianten verwendet werden:

$a = b$

$a < b$

$a \leq b$

$a > b$

$a \geq b$

$a \neq b$

a ist **gleich** b

a ist **kleiner** als b

a ist **kleiner oder gleich** b

a ist **größer** als b

a ist **größer oder gleich** b

a ist **ungleich** b

Zusätzlich können auch mehrere Ausdrücke **miteinander verknüpft** werden:

$((a=b) \text{ AND } (c=d))$

$((a=b) \text{ OR } (c=d))$

$((a=b) \text{ XOR } (c=d))$

a gleich b **und gleichzeitig** c gleich d

a gleich b **oder** c gleich d

a gleich b **exklusiv-oder** c gleich d

‘Beispiel 1

Function main

Integer i

...

For i= 1 **To** 10

Jump Abholen C0

On o_Vakuum

Wait 0.2

Jump P(i) C0

Off o_Vakuum

Wait 0.2

Next i

...

Fend

‘Beispiel 2

Function main

Integer i

...

For i= 1 **To** 10 **Step** 3

Jump Abholen C0

On o_Vakuum

Wait 0.2

Jump P(i) C0

Off o_Vakuum

Wait 0.2

Next i

...

Fend

‘Beispiel 3

Function main

Integer i

...

For i= 10 **To** 1 **Step** -1

Jump Abholen C0

On o_Vakuum

Wait 0.2

Jump P(i) C0

Off o_Vakuum

Wait 0.2

Next i

...

Fend

Beispiel 1

Function main

Do

Jump Abholen C0

On i_Vakuum

Wait 0.2

Jump Einsetzen C0

Off i_Vakuum

Wait 0.2

Loop

Fend

Beispiel 2

Function main

Integer i

Do While i < 10

i= i+1

Jump Abholen C0

On i_Vakuum

Wait 0.2

Jump P(i) C0

Off i_Vakuum

Wait 0.2

Loop

Fend

Beispiel 3

Function main

Integer i

Do

i= i+1

Jump Abholen C0

On i_Vakuum

Wait 0.2

Jump P(i) C0

Off i_Vakuum

Wait 0.2

Loop Until i >= 10

Fend

Hinweis:
*Eine leere Do...Loop
Schleife lässt die
CPU-Last gegen
100% ansteigen!*



PALETTEN-FUNKTIONEN

Zweidimensionale Palette

Wenn viele Produkte in einem Tray, Blister oder einem ähnlichen Behälter an definierten Punkten liegen, muss nicht jede einzelne Position geteacht werden. Hierfür kennt der Roboter sogenannte Paletten-Funktionen. Man unterscheidet hier zwischen ein- und zweidimensionalen Paletten.

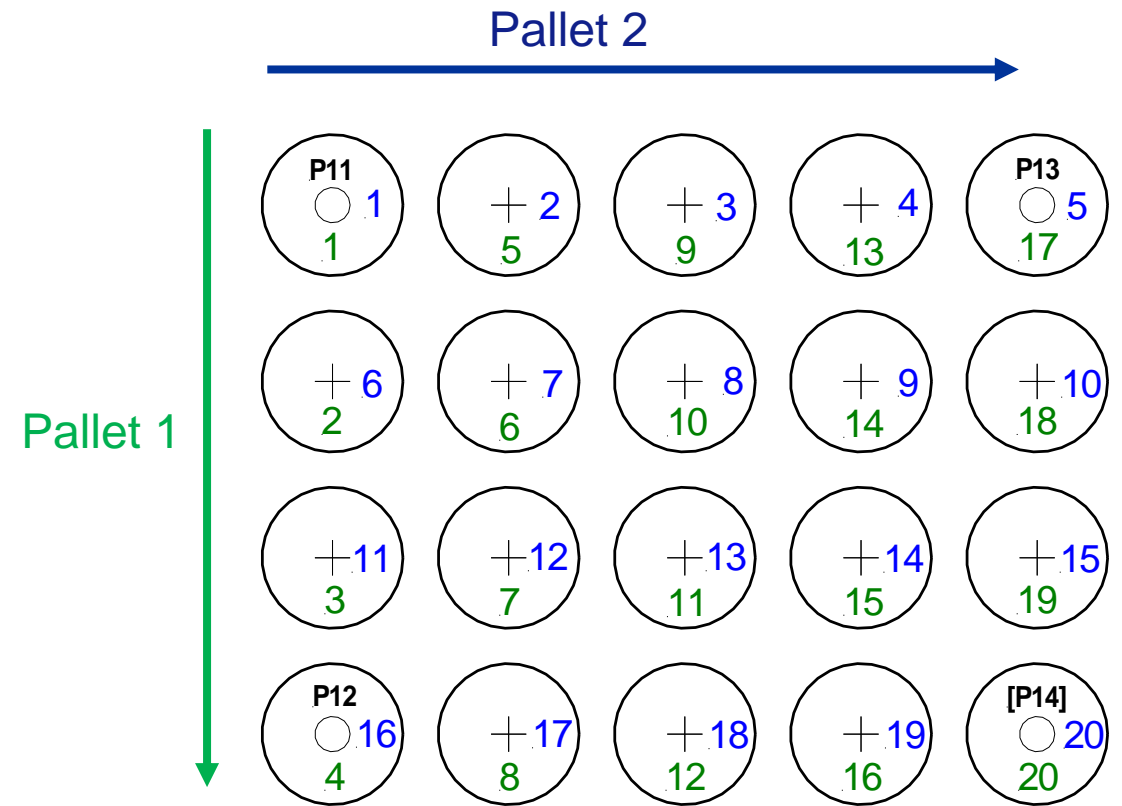
ZWEIDIMENSIONAL

Pallet 1, P11, P12, P13, 4, 5

Pallet 2, P11, P13, P12, 5, 4

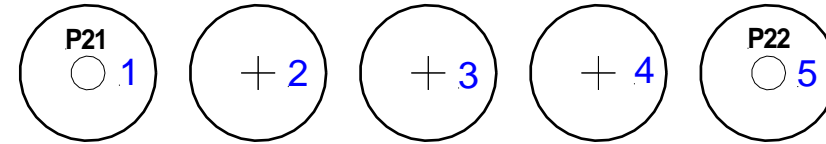
Pallet 2, P11, P13, P12, P14, 5, 4

1. Punkt: Ursprung
2. Punkt: Endpunkt Hauptarbeitsrichtung
3. Punkt: Endpunkt Nebenarbeitsrichtung



EINDIMENSIONAL

```
Pallet 3, P21, P21, P22, 1, 5  
Pallet 3, P21, P22, P21, 5, 1  
Pallet 3, P21, P22, P22, 5, 1
```



Aufruf der Palette im Programm

...

```
Jump Pallet (PalettenID, PalettenNest)
```

...



GENERELL GILT FÜR DIE PALETTENBEARBEITUNG:

Der Werkzeuganstellwinkel (U-Stellung) sowie die Armorientierung (Righty / Lefty) wird ausschließlich vom **Ursprungspunkt** der Palette verwendet !

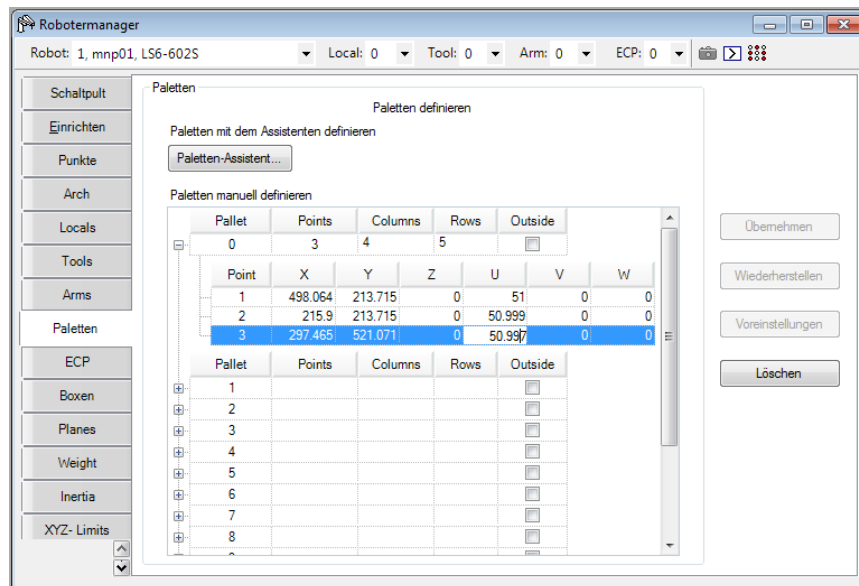
PALETTEN-FUNKTIONEN

Der Assistent

Neu in der RC+7 Version, ist ein Paletten-Assistent, um die Definition einer Palette geführt vornehmen zu können.

Der Assistent birgt allerdings **zwei große Nachteile**:

- 1.) die Palettenpunkte können im Nachhinein nicht mehr ohne weiteres angefahren werden, um diese zu überprüfen.
- 2.) die Palettendefinitionen werden **steuerungsbezogen** und nicht projektbezogen abgespeichert !!



Ab **RC+7** ist die Paletten-Definition eine **steuerungsbezogene** Einstellung.

In der **RC+5** muss die Paletten-Definition beim Programmstart durchlaufen werden.

Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go <i>P_End</i>	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump <i>P_End</i>	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Pallet <i>ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR</i>	Definition einer Palette Syntax zum Aufruf: Pallet (ID, NestID)	Definitionen



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If <i>Bedingung</i> Then ... Elseif <i>Bedingung</i> Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen

Syntax:	Hinweis:	Gruppe:
For <i>ZählerVar = Start To End</i> Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While <i>Bedingung</i> ... Loop	<i>Kopfgesteuerte</i> Do-Loop Schleife	
Do ... Loop Until <i>Bedingung</i>	<i>Fußgesteuerte</i> Do-Loop Schleife	



AUFGABENSTELLUNG

- Abholung von 10 Produkten an Rutsche 1
- Abholung von 10 Produkten an Rutsche 2
- Ablegen der Produkte in Palettenposition 1 bis 20

Viel Erfolg!

Zwischenstopp: PUNKT- MANIPULATION

PUNKT-MANIPULATION

Offsets für Punkte setzen

Es ist möglich während der Programmbearbeitung, einen geteachten Punkt entsprechend zu verändern / zu manipulieren. Dies ist für alle Koordinaten (X, Y, Z, U sowie V und W beim ProSix Roboter) möglich:

Go StartPos	= 100,100,-100,0
Go StartPos +U (30)	= 100,100,-100,30
Go StartPos -Z (30)	= 100,100,-130,0
Go StartPos :Z (0)	= 100,100,0,0

Mit einem **Doppelpunkt** gekennzeichnete Offsets werden direkt als Absolutposition zugewiesen.

Ebenfalls ist es möglich, Punktpositionen X, Y, Z und U als Einzelwerte zu definieren (für den ProSix Roboter zusätzlich dann noch die V und W Koordinate):

```
P1= XY (100, 250, -100, 90) /R
```

Alternativ können die Werte dann auch durch Variablen ersetzt werden:

```
P2= XY (Xval, Yval, Zval, Uval) /R
```

Natürlich können auf die Art und Weise auch direkte Fahrbefehle definiert werden:

```
Jump XY (Xval, Yval, Zval, Uval) /R
```



Bitte beachten!

Die Punktdefinition erfordert unbedingt die Stellung des Arms:

/R Rechtsarm

/L Linksarm

PUNKT-MANIPULATION

Teile einer Punktkoordinate bearbeiten

<code>CX (P1) = 140</code>	'Koordinate einstellen
<code>X= CX (P1)</code>	'Koordinate lesen
<code>P7= P6 :X (X)</code>	'X-Koordinate von P6 als Absolutwert P7 zuweisen (P6 bleibt P7 ändert sich)

Natürlich können so auch alle anderen Koordinaten beeinflusst werden (`CY`, `CZ`, `CU`, `CV`, `CW`).

<code>Print Hand (P3)</code>	'Armorientierung ausgeben
<code>Hand P8, Lefty</code>	'Armorientierung zuweisen
<code>Hand P7, Hand (P3)</code>	'Armorientierung übertragen

Es ist möglich, die aktuelle Position des Roboters im Programm auszulesen:

```
'Aktuelle Position im P1 ablegen:
```

```
P1 = RealPos
```

```
'Aktuelle Position im Run-Fenster ausgeben:
```

```
Print RealPos
```

```
'Distanzberechnung zwischen zwei Punkten:
```

```
If Dist(pGrundpos, Realpos) > 1 then
```

```
    Jump pGrundpos
```

```
Endif
```


Es besteht die Möglichkeit mit verschiedenen Punktedateien (*.pts) zu arbeiten. Standardmäßig arbeitet der Roboter mit der Defaulttabelle „robot1.pts“.

Um zur Laufzeit die aktive Punktetabelle zu laden oder zu speichern, verwenden Sie folgende Befehle:

'Punktedatei laden und vorhandene Punkte im Arbeitsspeicher löschen

```
LoadPoints "typ3.pts"
```

'Punktedatei laden und mit vorhandenen Punkten im Arbeitsspeicher

'zusammenführen (bereits bestehende Punkte werden **überschrieben**)

```
LoadPoints "typ3.pts", Merge
```

'Punktliste aus dem Arbeitsspeicher in Punktedatei speichern

```
SavePoints "typ3.pts"
```

Aufgabenstellung: PICK & PLACE Erweiterung #2



AUFGABENSTELLUNG

- Abholung von 20 Produkten
(Wechsel von Rutsche 1 zur Rutsche 2 - Sensorabhängig)
- Ablegen der Produkte in Palettenposition 1 bis 20
- Inbetriebnahme der Remote-Funktionalität

→ Was wird hierzu benötigt?

Mittels des PRINT Befehls können Textausgaben im RUN- bzw. Remote-Fenster ausgegeben werden um z.B. Informationen für den Anlagenbediener darzustellen.

```
Print "Warten auf Start..."
```

Es können auch Inhalte von Variablen oder Berechnungen ausgegeben werden.

```
Function main
    Integer iWert1, iWert2
    iWert1 = 10
    iWert2 = 2
    Print "Wert1: ", iWert1
    Print "Wert2: ", iWert2
    Print "Ergebnis Addition: ", (iWert1 + iWert2)
Fend
```

BENÖTIGTE ZYKLUSZEIT FÜR PRINT AUSGABE

im Ausgabefenster: ca. **10ms** (über Netzwerk ggf. länger)

Mit dem PRINT Befehl ist es auch möglich, **Textausgaben auf dem Teachpanel** vorzunehmen. Hierzu muss der PRINT Befehl um die entsprechende Port-Nummer des jeweilig angeschlossenen Panels erweitert werden:

```
Global Integer iDevPortID
```

```
Global Integer iPunktID
```

```
Function main
```

```
Reset
```

```
Motor On
```

```
iDevPortID = DispDev
```

```
'Port-Adresse des angeschlossenen OP/TP ermitteln  
'#21 RC+ / #23 OP / #24 TP1 / #20 TP3
```

```
Cls #(iDevPortID)
```

```
'TP1-Bildschirm leeren
```

```
Print #(iDevPortID), "Anzufahrende Punktnummer angeben:"
```

```
Input #(iDevPortID), iPunktID
```

```
Print #(iDevPortID), ""
```

```
Print #(iDevPortID), "Punkt " , iPunktID, " wird angefahren"
```

```
Jump P(iPunktID)
```

```
'Fahrbefehl ausführen
```

```
Print #(iDevPortID), "*** FAHRAUFTRAG BEENDET ***"
```

**Sofern der PC angeschlossen ist, erfolgt
die Ausgabe zuerst immer auf
Port 21 (RC+).**

PARAMETERÜBERGABE

Wertübergabe zwischen Funktionen

BEISPIELAUFRUF MIT PARAMETERÜBERGABE:

```
Function fPlacePallet (iPalletID as Integer, iPlacePos as Integer) as Integer

    Jump Pallet(iPalletID, iPlacePos)
    fPlacePallet = iPlacePos          'Rückmeldung der abgelegten Position

Fend
```

```
Function main
    Integer iZaehler, iPlacedPosNumber
    For iZaehler = 1 to 20
        iPlacedPosNumber = fPlacePallet(1, iZaehler)
        Print iPlacedPosNumber
    Next
Fend
```



VORZEITIGER ABBRUCH

Funktionen können mit folgenden Befehl auch vorzeitig verlassen werden:

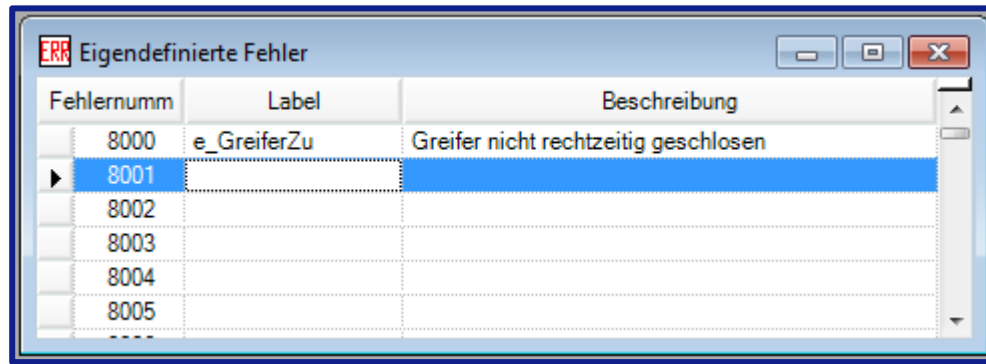
`Exit Function`

FEHLERMELDUNGEN

Individuelle Fehlermeldungen

Wie im Programmierbeispiel der vorherigen Seite gezeigt, wird hier eine individuelle Fehlermeldung generiert: „e_GreiferZu“. Benutzerdefinierte Fehlermeldungen können im Bereich von **8000 – 8999** definiert werden.

Hierzu im Menü „**Tools – Eigene Fehlermeldungen definieren**“ die gewünschten Fehlermeldungen definieren:



Die hier definierten Fehlermeldungen werden in der Systemhistorie auch entsprechend mitgeloggt.

REMOTE E/A-KONFIGURATION

Unter dem Menüpunkt **Einstellungen – Steuerung** in der Registerkarte **Konfiguration** kann unter dem Auswahlpunkt Steuergerät „Remote E/A“ auswählen. Anschließend kann unter „Remote-Steuerung“ für die einzelnen Funktionen (wie Start, Stop, Reset, ...) ein Eingang verwendet werden.

Die verwendeten Eingänge und Ausgänge sind im E/A-Monitor blau eingefärbt!

Einstellungen der Steuerung

Remote-Steuereingänge

Eingangssignal	Eingang Nr.
Start	0
SelProg1	frei
SelProg2	frei
SelProg4	frei
Stop	1
Pause	2
Continue	3
Reset	4
SetMotorsOn	frei
SetMotorsOff	frei

E/A-Monitor

Standardansicht | Anwenderansicht 1

Alle Eingänge | Alle Ausgänge

☒ Bits ☐ Bytes ☐ Worte

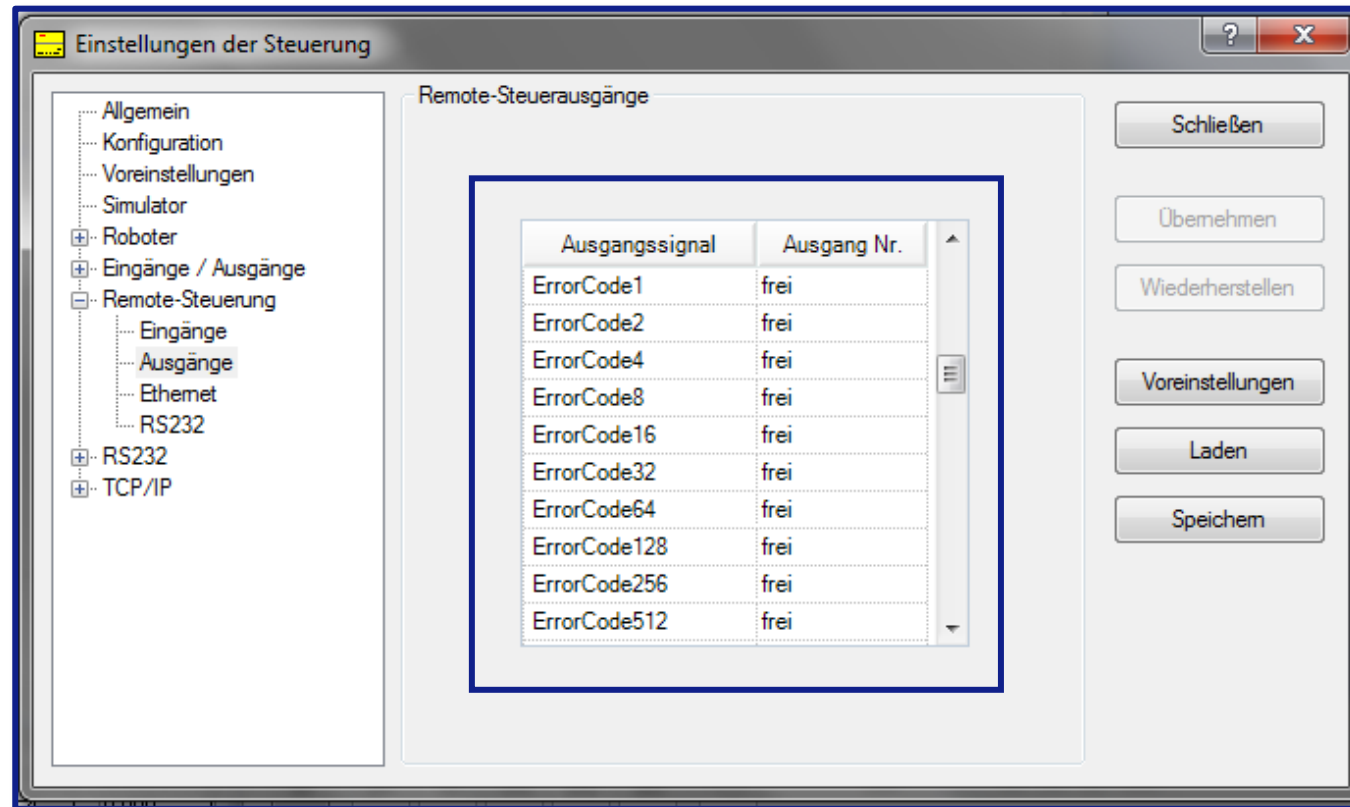
Bit	Status	Label
0	<input type="radio"/>	Start
1	<input type="radio"/>	Stop
2	<input type="radio"/>	Pause
3	<input type="radio"/>	Continue
4	<input type="radio"/>	Reset
5	<input type="radio"/>	
6	<input type="radio"/>	
7	<input type="radio"/>	
8	<input type="radio"/>	
9	<input type="radio"/>	

☐ Hexadezimale Werte

Virtueller- E/A- Modus

FEHLERMELDUNGEN ÜBER REMOTE AUSGEBEN

Ereignis-Nummern können per Remote an die SPS übergeben werden:

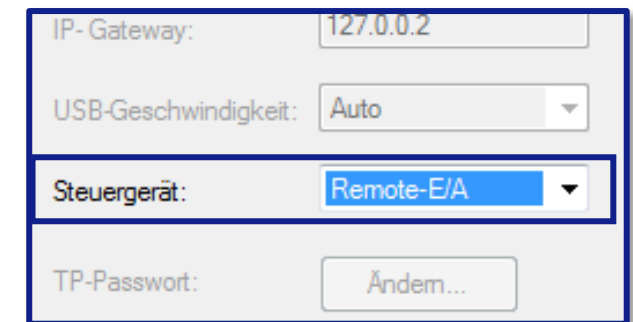
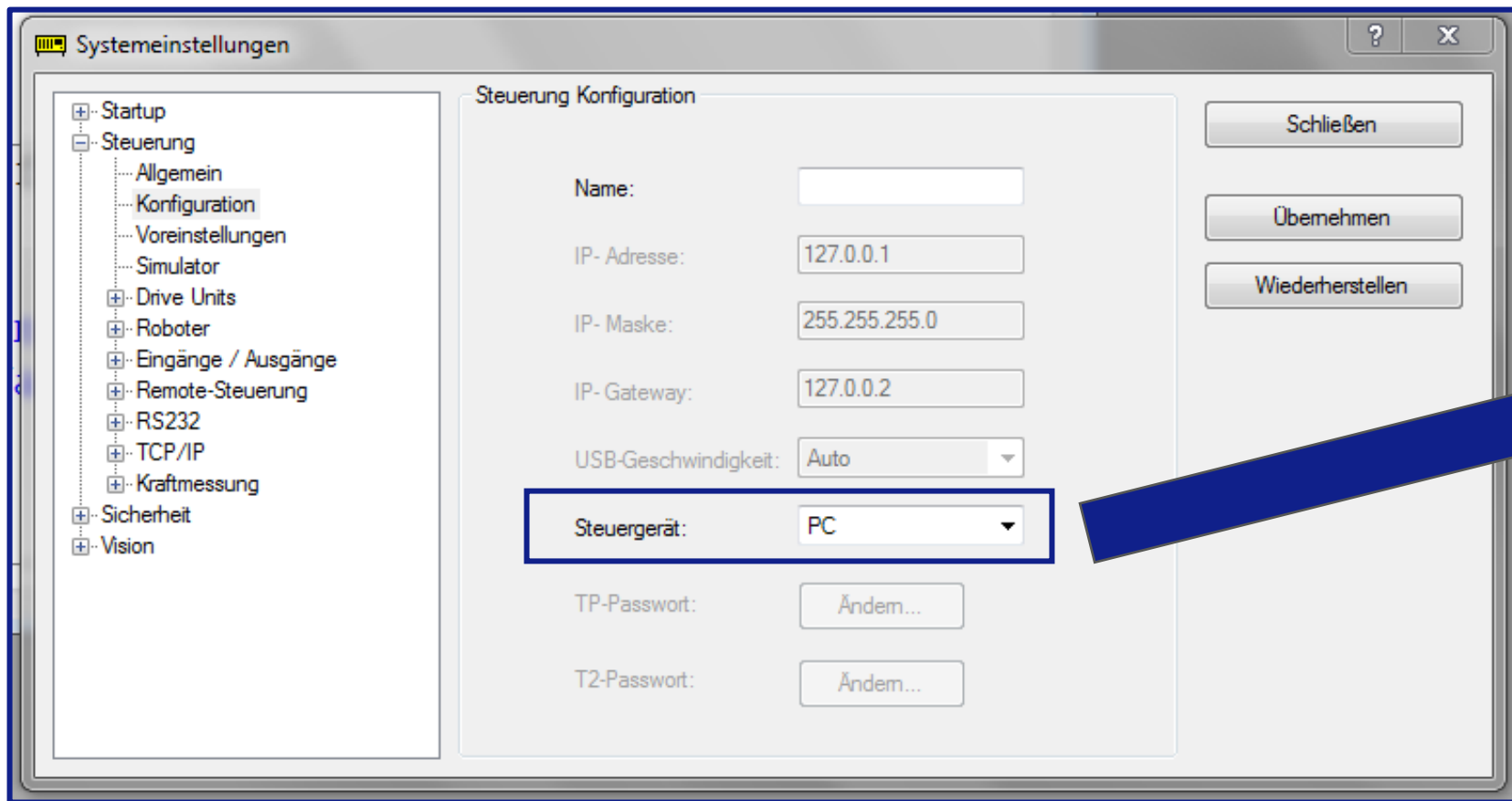


SPEICHERORT TEXTDATEI KLARTEXTMELDUNGEN

C:\EpsonRC70\system\german\ErrMsgG.txt

STEUERGERÄT EINSTELLEN

Zusätzlich muss das Steuergerät unter **Systemeinstellungen – Steuergerät** von „PC“ auf „**Remote-E/A**“ umgestellt werden. Andernfalls reagiert der Controller noch nicht auf eingehende Signale der Remote Schnittstelle.

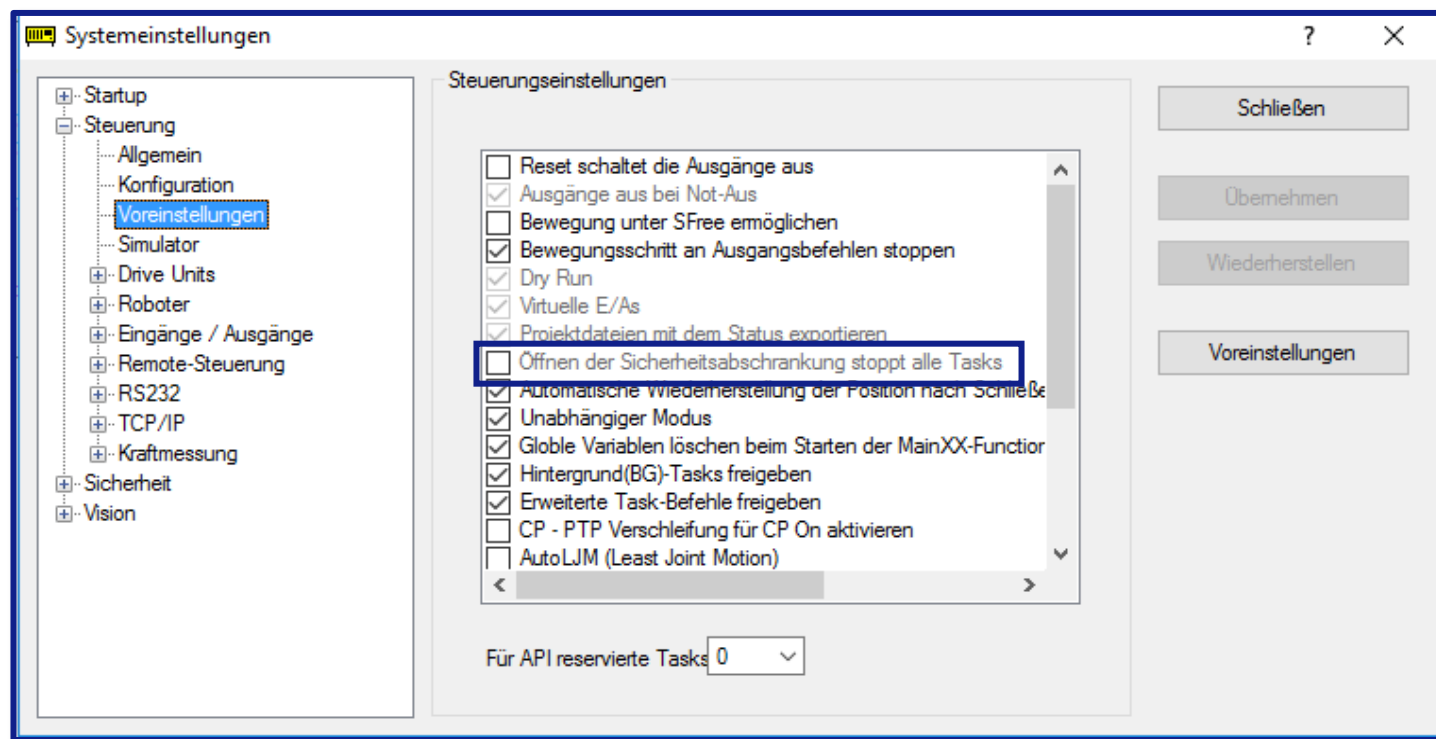


VERHALTEN BEI SAFEGUARD (= OFFENER TÜRE)

Sollte bei geöffneter Schutztüre, der Roboter von Hand bewegt worden sein, führt das System zwingend eine Recovery-Fahrt durch (GO-Bewegung) bevor das Programm fortgesetzt werden kann.

RECOVERY FAHRT UNTERBINDEN

Sollte es Recovery-Fahrt in der Anlage nicht möglich sein, müssen die laufenden Tasks abgebrochen werden. Dies kann zum Einen über die Systemeinstellung der Steuerung (RC+7 - Steuerungen) oder über einen kleinen Programmcode erfolgen.



Hierzu die folgende Einstellung aktivieren:

„Öffnen der Sicherheitsabschränkung stoppt alle Tasks“

REMOTE-KONFIGURATION

Überwachung Safeguard / Recovery-Fahrt

```
Function fSafeguardMonitor_RC70
    Do
        Wait SafetyOn = True
        Wait SafetyOn = False
        If Dist(RecoverPos, RealPos) > 10 Then
            Quit All
        EndIf
    Loop
Fend
```

```
Function fSafeguardMonitor_RC50
    Do
        Wait SafetyOn = True
        Wait 0.5
        P999 = RealPos
        Wait SafetyOn = False
        If Dist(P999, RealPos) > 10 Then
            Quit All
        EndIf
    Loop
Fend
```

```
Function Main
    Xqt 2, fSafeguardMonitor_RC70, NoPause
    ...
Fend
```



ACHTUNG

Für RC+5 und RC+7 Software sind **unterschiedliche** Unterfunktionen zu verwenden.

Konfiguration Remote-E/A zur SPS-Nutzung

REMOTE AUSGÄNGE

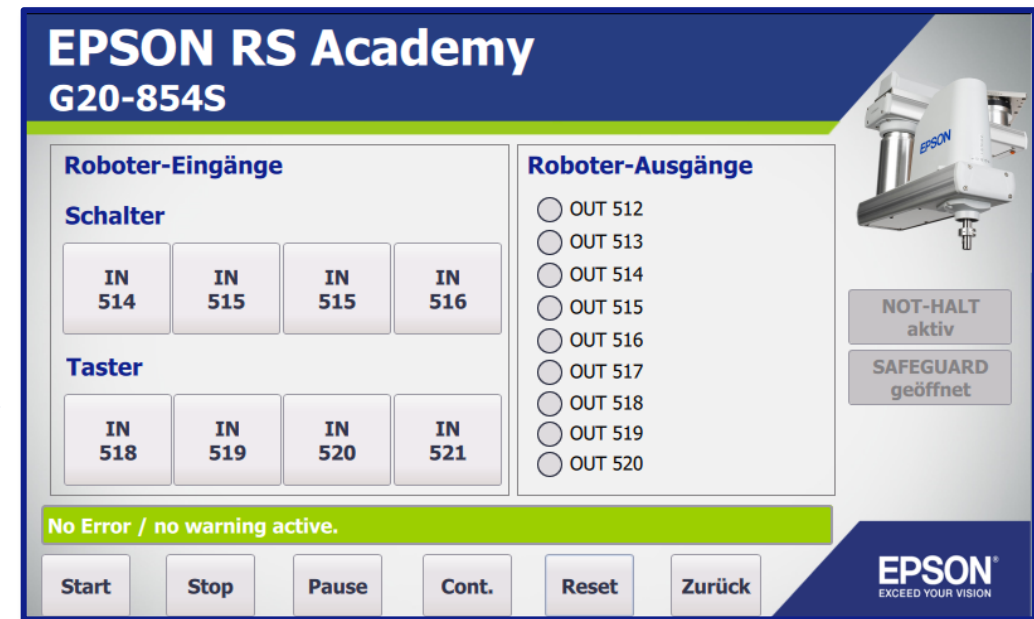
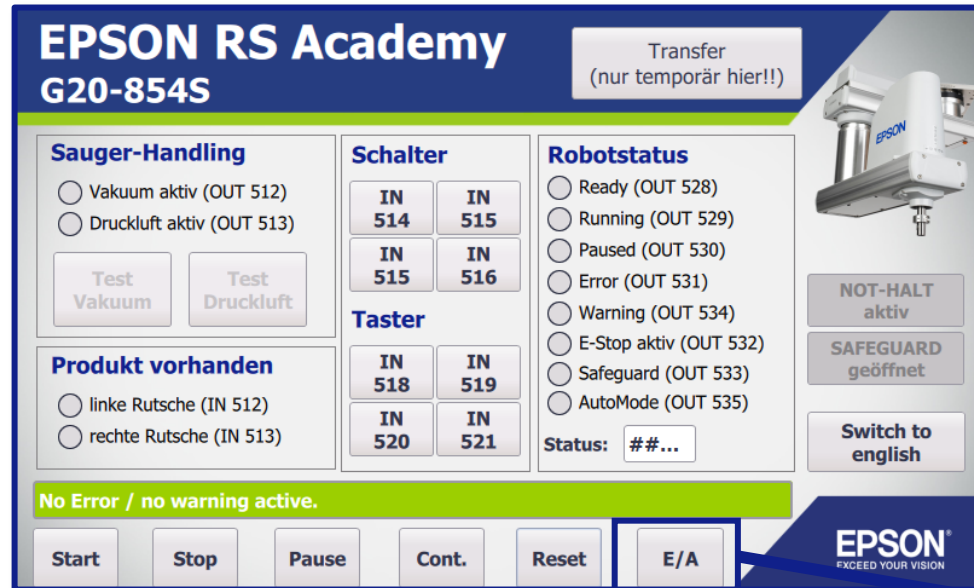
Ready	OUT 528
Running	OUT 529
Paused	OUT 530
Error	OUT 531
EStopON	OUT 532
SafeguardON	OUT 533
Warning	OUT 534
AutoMode	OUT 535

ErrorCode1	OUT 544
ErrorCode2	OUT 545
ErrorCode4	OUT 546
ErrorCode8	OUT 547
ErrorCode16	OUT 548
ErrorCode32	OUT 549
ErrorCode64	OUT 550
ErrorCode128	OUT 551
ErrorCode256	OUT 552
ErrorCode512	OUT 553
ErrorCode1024	OUT 554
ErrorCode2048	OUT 555
ErrorCode4096	OUT 556
ErrorCode8192	OUT 557

REMOTE EINGÄNGE

Start	IN 528
Stop	IN 529
Pause	IN 530
Continue	IN 531
Reset	IN 532

Panel Bilder Simatic HMI



Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go <i>P_End</i>	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump <i>P_End</i>	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Pallet <i>ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR</i>	Definition einer Palette Syntax zum Aufruf: Pallet (ID, NestID)	Definitionen



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If <i>Bedingung</i> Then ... ElseIf <i>Bedingung</i> Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen

Syntax:	Hinweis:	Gruppe:
For <i>ZählerVar = Start To End</i> Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While <i>Bedingung</i> ... Loop	<i>Kopfgesteuerte</i> Do-Loop Schleife	
Do ... Loop Until <i>Bedingung</i>	<i>Fußgesteuerte</i> Do-Loop Schleife	



Befehlsübersicht

Local, Offset, Boxen & Planes

Syntax:	Hinweis:	Gruppe:
Go Punkt +U(Wert)	Verändert den „U-Wert“ Merke: Die Punktinformationen bleiben unberührt!	Offset
Go Punkt -Z(Wert)	Verändert den „Z-Wert“ Merke: Die Punktinformationen bleiben unberührt!	
Go Punkt :Z(Wert)	Für die Zeile in der dieser Befehl steht, wird dieser angegebene „Z-Wert“ für den gewünschten Punkt verwendet. Merke: Die Punktinformationen bleiben unberührt!	
Punkt = XY(X_Wert, Y-Wert, Z_Wert, U_Wert)	Die im Punkt vorhanden Werte werden mit Hilfe der Punktmanipulation überschrieben.	Punktmanipulation



AUFGABENSTELLUNG

- Abholung von 20 Produkten
(Wechsel von Rutsche 1 zur Rutsche 2 - Sensorabhängig)
- Ablegen der Produkte in Palettenposition 1 bis 20
- Inbetriebnahme der Remote-Funktionalität

Viel Erfolg!

Aufgabenstellung: CP-Bewegung



AUFGABENSTELLUNG

- Entleeren Sie die Palette mit dem Roboter
- Teachen Sie die notwendigen Punkte
- Bringen Sie die Produkte zurück auf die Rutschen

→ Was wird hierzu benötigt?

CP-Bewegung



Statt einer Bogenfahrt zwischen P0 und P1 kann auch eine interpolierte Fahrt (geradlinig) zwischen den beiden Punkten erfolgen.

Der Befehl MOVE zählt zu der Gruppe der „CP“ – Fahrbefehle.

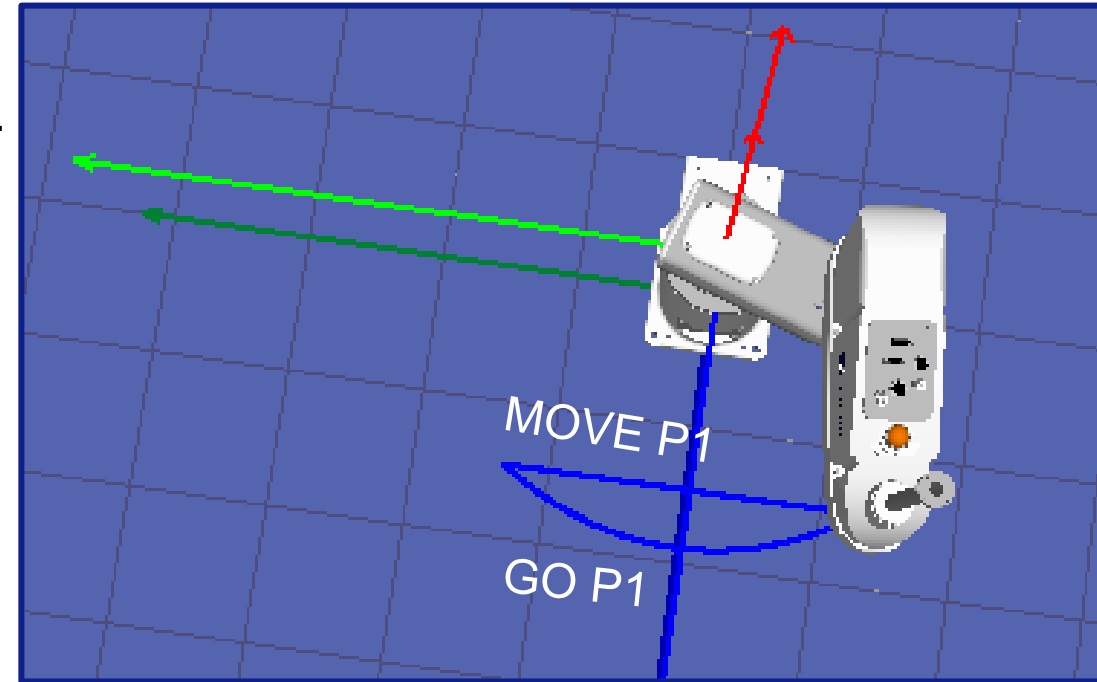
SYNTAX

MOVE P1

Wenn ausschließlich der **Anstellwinkel des Tools** auf einer XY-Position geändert werden soll, ist der Move-Befehl durch den Parameter ROT zu erweitern.

SYNTAX

MOVE P1 +U(90) ROT



ÜBERGÄNGE



ÜBERGÄNGE

Bewegungen mit CP verflüssigen

Um eine flüssigere Bewegung (ohne Stopp an der Zielposition zu erreichen, kann das Verschleifen mit dem CP-Befehl programmiert werden.

Function main	Function main
Move P1 CP	CP ON
Move P2	Move P1
Fend	Move P2
	Move P3
	CP OFF
	Fend

Bei der Verwendung von CP ON/OFF: Möchte man einen Punkt dennoch nicht verschleifen, kann dies mit dem Befehl **WaitPos** erreicht werden:

```
CP ON
Move P1
Move P2
WaitPos
Move P3
CP OFF
```


ÜBERGÄNGE

Bewegungen mit CP verflüssigen

Um Hindernisse zu umfahren, kann die Option CP ebenfalls verwendet werden:

```
Jump P10 C0 Limz -20 CP
```

```
Jump P2 C0 Limz -20
```



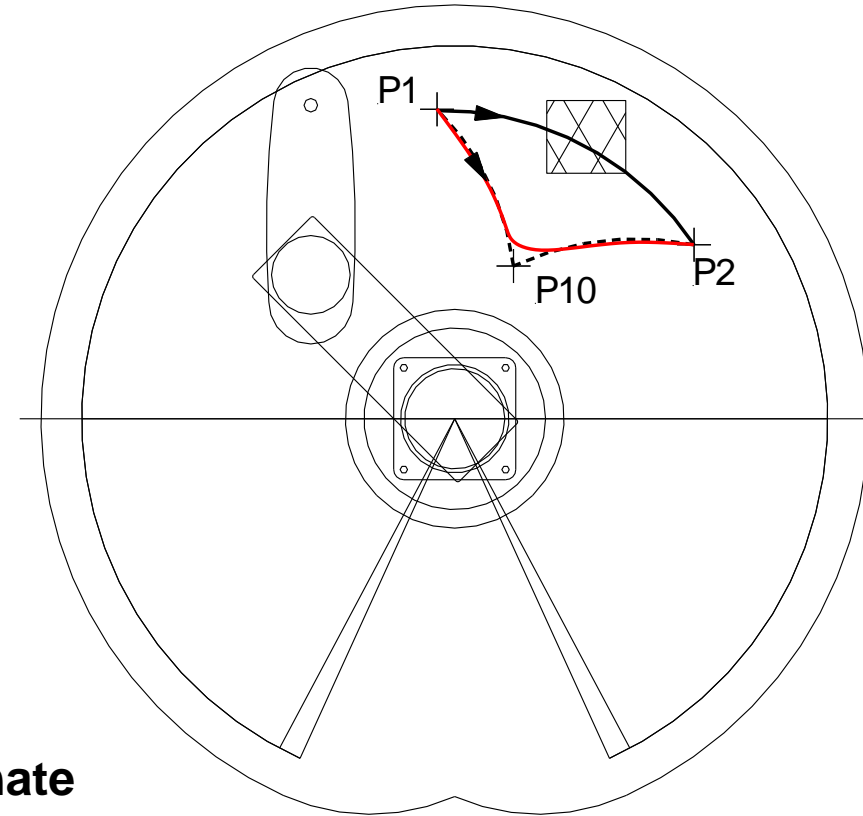
WICHTIG!

Damit der Roboter nicht am Zwischenpunkt stoppt, muss die Z-Koordinate dieses Punktes gleich dem Limz-Wert sein.

Alternativ:

```
Jump P10 :Z(Limz) C0 CP
```

```
Jump P2 C0
```



ÜBERGÄNGE

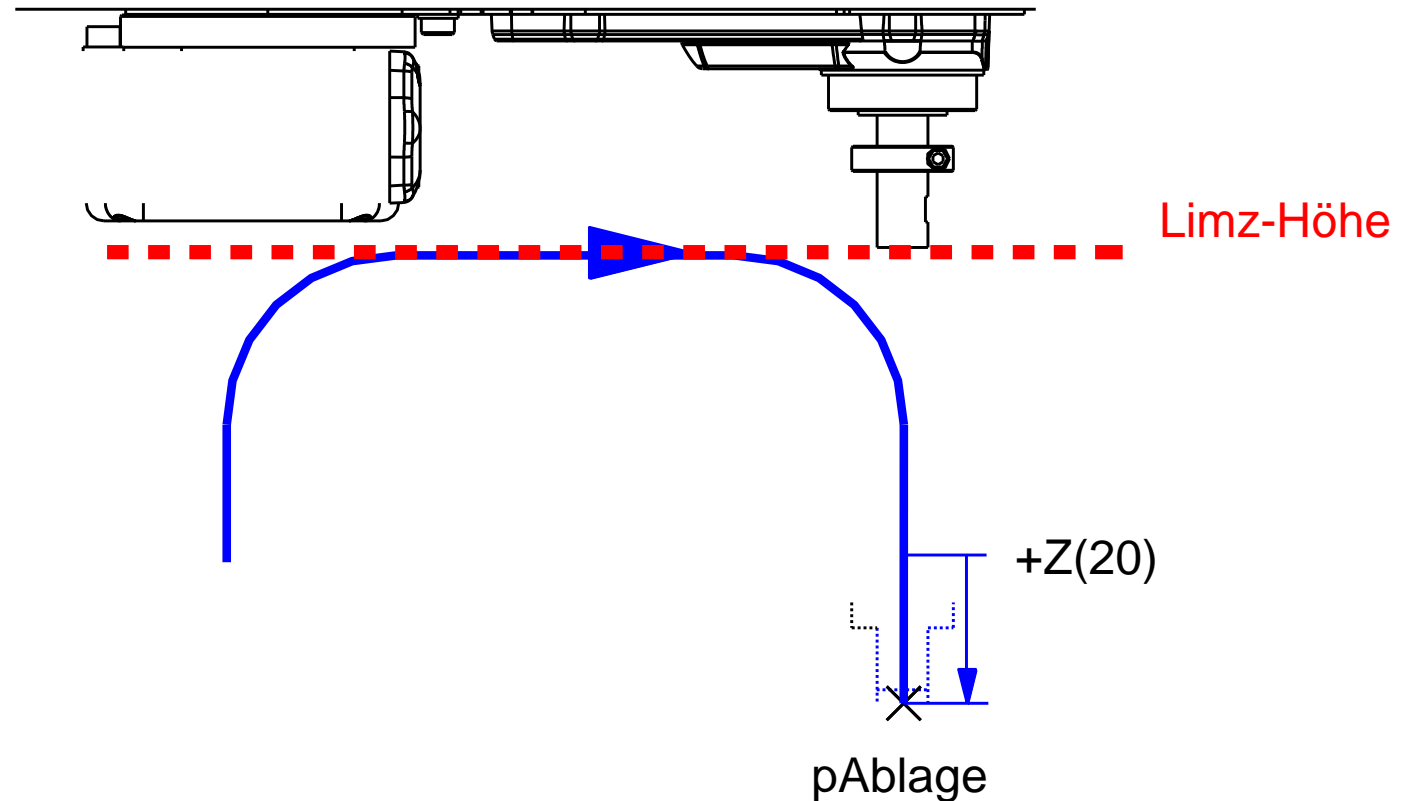
Bewegungen mit CP verflüssigen

WARTEN AUF FREIGABE

```
Jump pAblage +Z(20) C0 Limz -20 CP  
Wait Sw(Wt_Bereit)= On  
Go pAblage
```

ÄNDERN DER GESCHWINDIGKEIT

```
Speed 100  
Jump pAblage +Z(20) C0 Limz -20 CP  
Speed 5  
Go pAblage  
Speed 100
```



GESCHWINDIGKEIT



GESCHWINDIGKEIT

Die verschiedenen Parameter

BEFEHLSGRUPPE

PTP

CP

Befehle:

Jump / Go

Move

Move Rot

Geschwindigkeit:

Speed [%]

SpeedS [mm/s]

SpeedR [°/s]

Beschleunigung:

Accel [%]

AccelS [mm/s²]

AccelR [°/s²]

CP-Option:

Zum Umfahren
von Hindernissen

Zum Überschleifen
von Bewegungen

Einfluss auf Geschwindigkeit

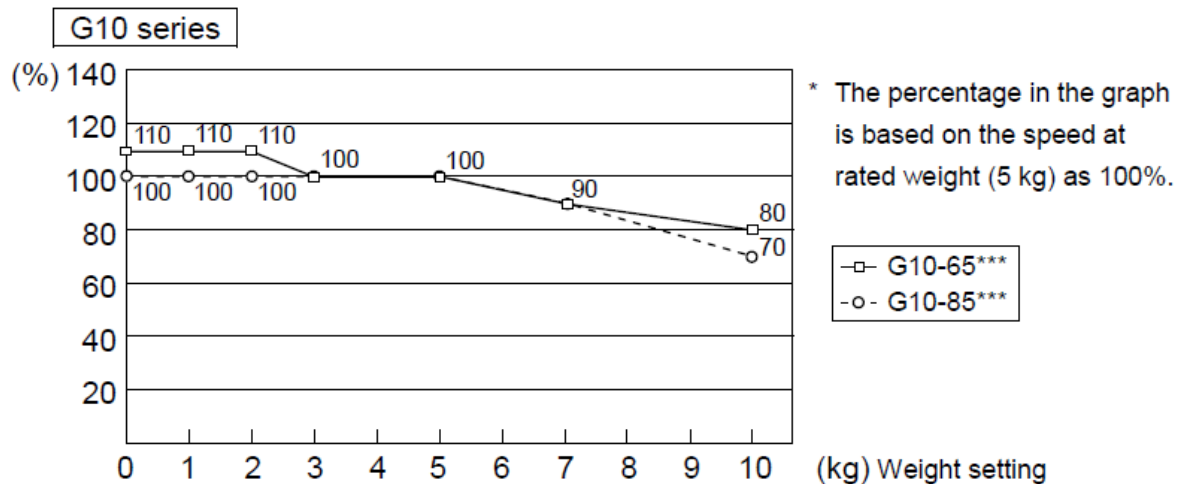
& Beschleunigungen:

Weight [kg]
Inertia
Pos. der Z-Achse

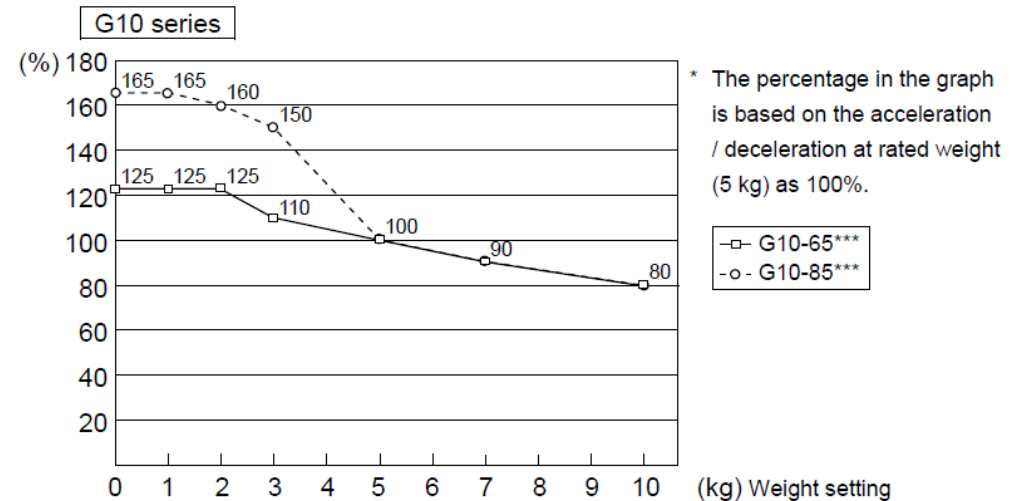
- / -

Auswirkungen auf SPEED / ACCEL (am Beispiel eines G10-Roboters)

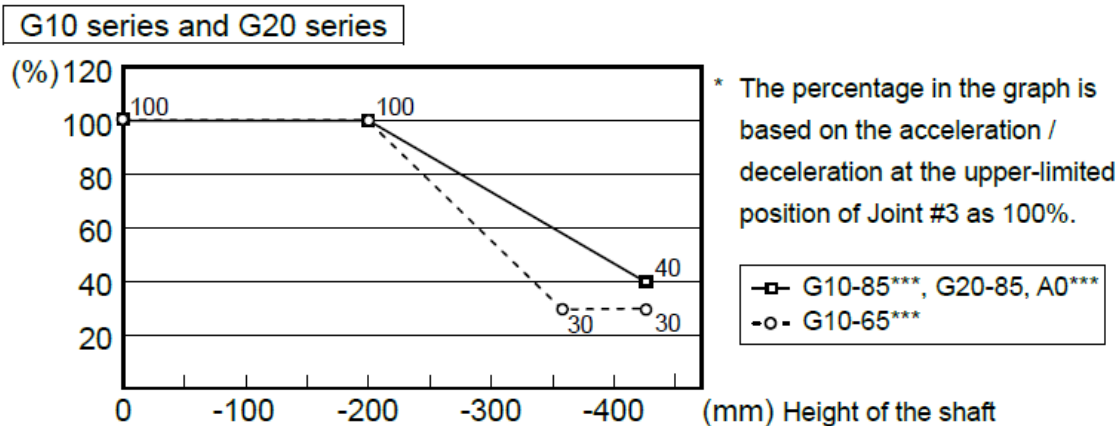
Automatic speed setting by Weight



Automatic acceleration/deceleration setting by Weight



Automatic acceleration/deceleration vs. Joint #3 position



GESCHWINDIGKEIT

Initialisierungsroutine

```
Function fInit
'Initialisierungsroutine
'*****

'ARCH Definitionen
Arch 0, 30, 30
Arch 1, 40, 40

'Greifergewicht einstellen
Weight 4.0      'kg

'Lastträgheitsmoment und Exzentrizität
Inertia 0.0200, 0.0

'Motoren einschalten
Motor On

'Geschwindigkeiten für High Power Modus setzen
Speed 80          'in % für PTP-Bewegung
Accel 80, 80      'in % für PTP-Bewegung

Speeds 1000       'in mm/s für CP-Bewegung
Accels 7500, 7500 'in mm/s2 für CP-Bewegung

SpeedR 200        'in °/s für CP mit ROT-Erweiterung
AccelR 200, 200   'in °/s2 für CP mit ROT-Erweiterung

'Grundposition anfahren
'ohne LIMZ oder ARCH
Jump pGrundposition

'LIMZ Wert einstellen
LimZ -50

Fend
```

Achtung:
Standardwert
für G10!

In der Initialisierungsroutine werden Standard-Einstellungen festgelegt, die ab sofort für das ganze Programm gelten sollen.

Hierzu zählen u.a.:

- ARCH Definitionen
- Greifergewicht / Lastträgheitsmoment
- Geschwindigkeitsvorgaben
- Standardzustände für Ausgänge & Merker setzen
- Standardwerte für Variablen setzen
- TOOL und LOCAL Einstellungen setzen

Wichtig:

Erst **nachdem** der Roboter die Grundposition erreicht hat, schalten Sie bitte bei Bedarf in den Power High Betrieb um !

Für Geschwindigkeiten, Beschleunigungen und Modus-Auswahl gilt nachfolgende Besonderheit:

Nach dem Eintreten eines der **nachfolgenden Ereignisse** oder der Verwendung eines der nachfolgenden Befehle, werden die Vorgabewerte wieder auf die Initialisierungswerte / Standardwerte zurückgesetzt:

- Neustart des Controllers
- MOTOR ON
- SFREE / SLOCK / BRAKE
- RESET / Reset Error
- Stopp Button / Quit All (zum Stoppen aller laufenden Tasks)

Hiervon betroffen sind unter anderem folgende Befehle:

- SPEED / SPEEDS / SPEEDR
- ACCEL / ACCELS / ACCEL R
- POWER (wird zu LOW gesetzt)
- LIMZ / FINE

Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go <i>P_End</i>	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump <i>P_End</i>	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Move <i>P_End</i>	Führt einen „Move“ zum Punkt „P_End“ aus.	CP-Fahrbefehl
Pallet <i>ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR</i>	Definition einer Palette Syntax zum Aufruf: <i>Pallet (ID, NestID)</i>	Definitionen



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If <i>Bedingung</i> Then ... Elseif <i>Bedingung</i> Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen

Syntax:	Hinweis:	Gruppe:
For <i>ZählerVar = Start To End</i> Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While <i>Bedingung</i> ... Loop	<i>Kopfgesteuerte</i> Do-Loop Schleife	
Do ... Loop Until <i>Bedingung</i>	<i>Fußgesteuerte</i> Do-Loop Schleife	



Befehlsübersicht

Local, Offset, Boxen & Planes

Syntax:	Hinweis:	Gruppe:
Go Punkt +U(Wert)	Verändert den „U-Wert“ Merke: Die Punktinformationen bleiben unberührt!	Offset
Go Punkt -Z(Wert)	Verändert den „Z-Wert“ Merke: Die Punktinformationen bleiben unberührt!	
Go Punkt :Z(Wert)	Für die Zeile in der dieser Befehl steht, wird dieser angegebene „Z-Wert“ für den gewünschten Punkt verwendet. Merke: Die Punktinformationen bleiben unberührt!	
Punkt = XY(X_Wert, Y-Wert, Z_Wert, U_Wert)	Die im Punkt vorhanden Werte werden mit Hilfe der Punktmanipulation überschrieben.	Punktmanipulation



AUFGABENSTELLUNG

- Entleeren Sie die Palette mit dem Roboter
- Teachen Sie die notwendigen Punkte
- Bringen Sie die Produkte zurück auf die Rutschen

Viel Erfolg!

Zwischenstopp: BOXEN & PLANES

BOXEN & PLANES

Box-Definition

Durch Boxen und Planes kann die **Position des Roboters innerhalb des Arbeitsbereichs** geprüft werden. Diese Boxen und Planes sind frei konfigurierbar. Wahlweise programmatisch zur Laufzeit:

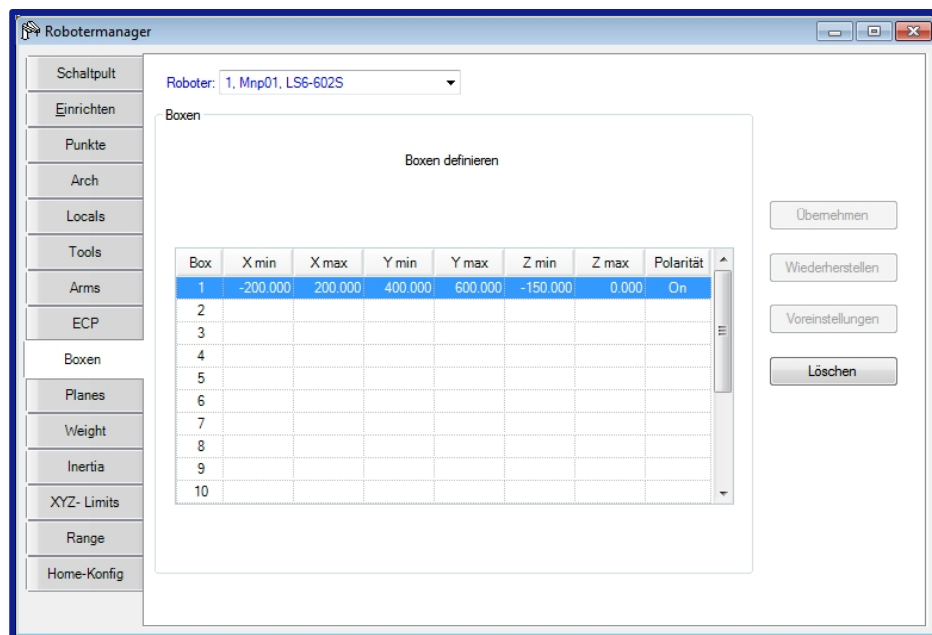
SYNTAX:

`Box Nr, RobotNr, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax, Logik`

BEISPIEL:

`Box 1, 1, -200, 200, 400, 600, -150, 0, On`

Oder im **Roboter-Manager** in den Registerkarten „Boxen“ und „Planes“:



LOGIK:

Dient der Statusinformation für die Remote-Schnittstelle:

ON = Remote Ausgang EIN
OFF = Remote Ausgang AUS
... wenn der Roboter sich innerhalb der Box befindet.

BOXEN & PLANES

Plane-Definition

Mit einer Plane wird eine Ebene in den Arbeitsraum gelegt. Abhängig von der Roboterposition wird so ein True oder False zurück geliefert.

SYNTAX

`Plane Nr, RobotNr, XY(X, Y, Z, U, V, W)`

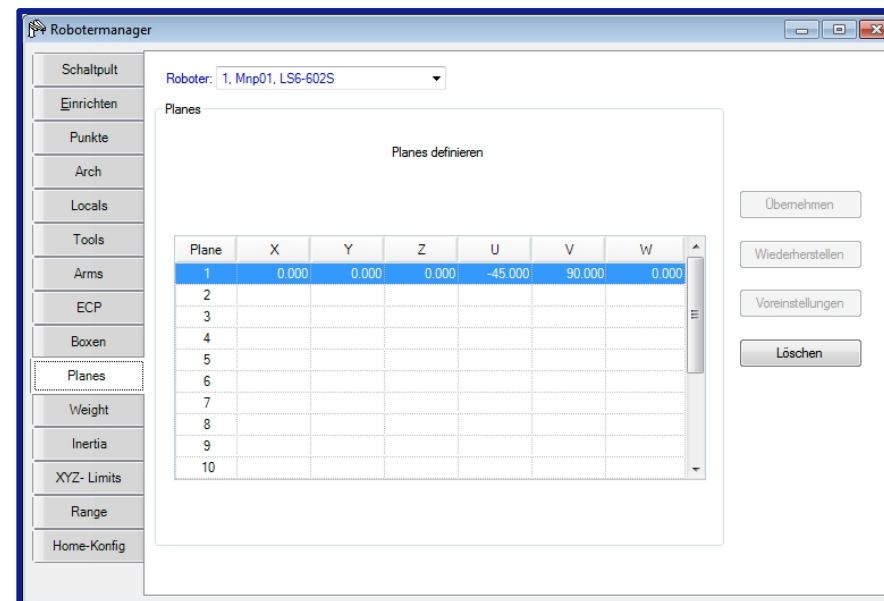
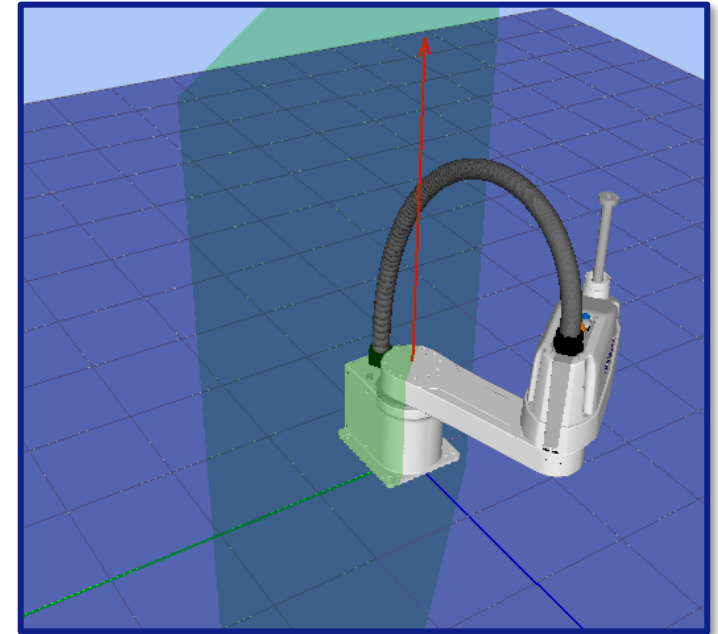
`Plane Nr, RobotNr, Ursprungspunkt`

BEISPIEL

`Plane 1, 1, XY(0, 0, 0, -45, 90, 0)`

`Plane 1, 1, P1`

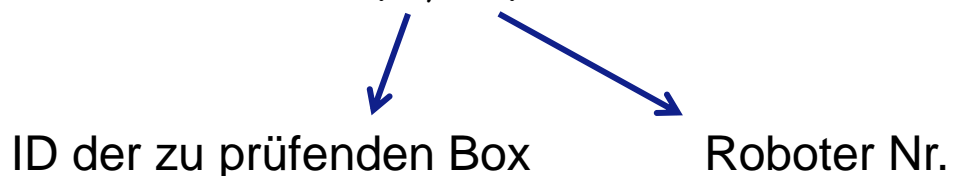
Bei V=0 liegt die Plane waagerecht im Arbeitsraum.



Zur Statusabfrage, ob der Roboter sich innerhalb oder außerhalb einer definierten Box oder vor bzw. hinter des definierten Planes steht, gibt es zwei Möglichkeiten:

1. ABFRAGE ZUR PROGRAMMLAUFZEIT

If InsideBox(1, 1) = True Then...



If InsidePlane(1, 1) = True Then...



2. REMOTE SCHNITTSTELLE

Über die Remote-Schnittstelle können digitale Ausgänge definiert werden, die auch bei gestopptem Roboterprogramm automatisch aktualisiert werden.

Aufgabenstellung: PALLET OUTSIDE



AUFGABENSTELLUNG

- Erweiterung des bisherigen Programms um eine weitere Palette (inkl Typumschaltung)
- Speichern Sie die belegten Paletten-Nester
- Ergänzen Sie eine Abfrage ob sich der Roboter im Bereich einer Palette befindet. Übermitteln Sie diese Information mit einem Paralleltask
-> Sperrfunktionalität für weitere Bearbeitungsschritte der SPS
(Verwenden Sie Merker, Paralleltask und Boxen)

Optional:

- Messung der Gesamttaktzeit für einen kompletten Palettendurchlauf

→ **Was wird hierzu benötigt?**

PALETTEN-FUNKTIONEN

Pallet Outside

Um solche Palettenlayouts verwenden zu können, ist die Erweiterung des Palettenbefehls um die Option **OUTSIDE** notwendig. Damit ist es möglich, auch Punkte außerhalb der definierten Palette zu erreichen.

Function main

```
Pallet Outside, 1, P11, P12, P13, 2, 3
```

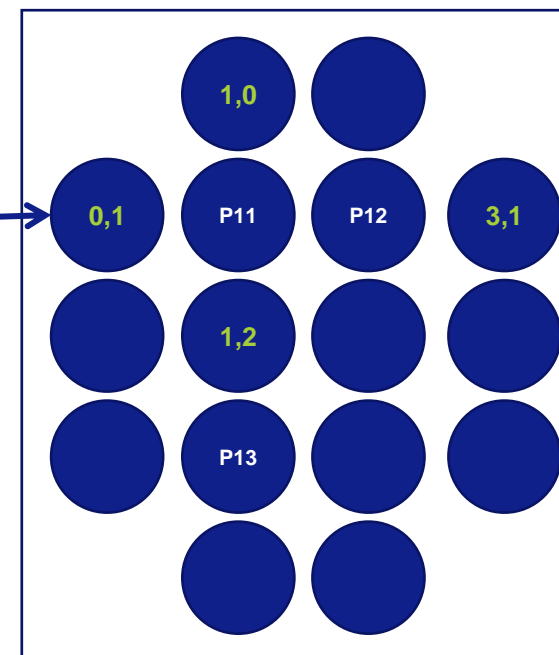
```
Jump Pallet(1, 0, 1)
```

Fend

Paletten-ID

Paletten Nest
in **Hauptarbeits-**
richtung

Paletten Nest
in **Nebenarbeits-**
richtung



PALETTEN-FUNKTIONEN

Pallet Outside

WEITERE PALETTEN-BEISPIELE AUS DER RC+ HILFE:

-2,10						
			P3			
			1,5	2,5	3,5	4,5
			1,4	2,4	3,4	4,4
			1,3	2,3	3,3	4,3
			1,2	2,2	3,2	4,2
			1,1	2,1	3,1	4,1
			P1		P2	

Pallet Outside, 1, P1, P2, P3, 4, 5
Jump Pallet(1, -2, 10)

Sample						
		1,6			4,6	
-1,4		1,4	2,4	3,4	4,4	6,4
		1,3	2,3	3,3	4,3	
		1,2	2,2	3,2	4,2	
-1,1		1,1	2,1	3,1	4,1	6,1
		1,-1			4,-1	

MULTITASK



EPSON®

MERKER HANDLING

Anstelle von Ein- und Ausgängen oder Variablen können zur Speicherung von Zuständen auch Merker verwendet werden.

MERKER SETZEN/RÜCKSETZEN (BIT-OPERATIONEN)

MemOn 1	`Merker 1 setzen
MemOff 2	`Merker 2 rücksetzen
If MemSw (3) = On Then...	`Merker 3 = eingeschaltet ?
If MemSw (4) = Off Then...	`Merker 4 = ausgeschaltet ?

MERKERBEFEHLE (BYTE- UND WORT-OPERATIONEN)

MemIn (0)	`Merkerbyte 0 lesen
MemOut 0, &h5	`5 ins Merkerbyte 0 schreiben
MemInW (0)	`Merkerwort 0 lesen
MemOutW 0, &h5	`5 ins Merkerwort 0 schreiben

MULTITASKING & PARALLELE PROZESSE

Parallel-Prozess !...!

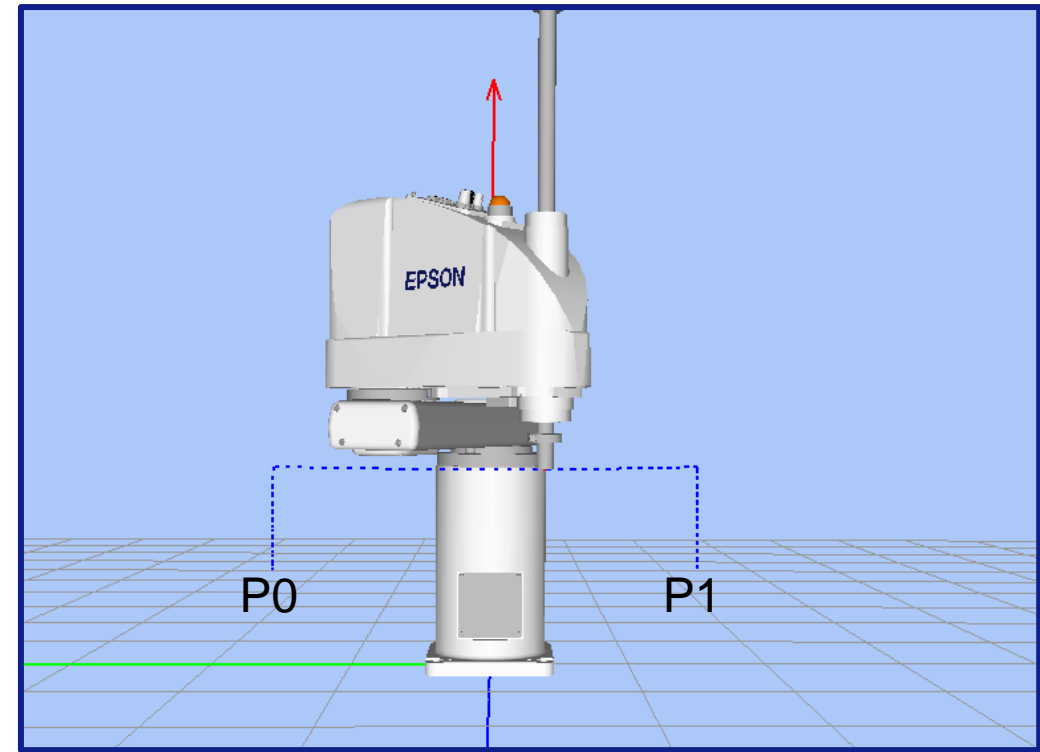
Während eines Bewegungsbefehls kann ein Ausgang entsprechend des zurückgelegten Weges gesetzt oder zurückgesetzt werden.

Jump P1 ! **D50**; **On** 1 !

Jump P1 ! **D0**; **On** 1; **D50**; **Off** 1 !

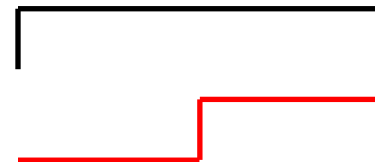
Einleitung des parallelen Prozess mit einem **Ausrufezeichen (!)**.

D50 = Distanz 50% des Fahrwegs
(die %-Angabe bezieht sich nur auf den **horizontalen** Weg des Jumps)



Fahrweg P0 nach P1

Ausgang 1



MÖGLICHE WEITERE FAHRBEFEHLE

Arc, Arc3, Move, Go, Jump, Jump3, Jump3CP

MULTITASKING & PARALLELE PROZESSE

Parallel-Prozess !...!

Im Rahmen des parallelen Prozesses (!...!) können **folgende Befehle** verwendet werden:

On / Off <i>n</i>	Ausgänge setzen / rücksetzen
MemOn / MemOff <i>n</i>	Merker setzen / rücksetzen
Out / OutW	Ausgänge Byte-/Wortweise beschreiben
MemOut / MemOutW	Merker Byte-/Wortweise beschreiben
Wait	Wait-Funktion in allen möglichen Varianten
Print	Textausgaben ins Run-Fenster oder TP

PROGRAMMIERBEISPIEL

```
Function fParallelProzess
    Move P1 !D10; On 5; Wait 0.5; Off 5!
Fend
```

Ein Task ist **eine Funktion** des aktuellen Projekts.

EIGENSCHAFTEN

- Limitierte Task-Anzahl:
RC+5-Software: bis zu 16 Tasks (Nr. 1 - 16)
RC+7-Software: bis zu 32 Tasks (Nr. 1 - 32)
- Timesharing-Verfahren: 2ms je Task
- Die Function Main ist immer Task Nummer 1
- Bearbeitung immer von 1 bis 16 bzw. 32
- Task an *Wait* hat Vorrang



ACHTUNG

Es kann immer nur eine Funktion oder ein Task auf den **Manipulator** zugreifen!

MULTITASKING & PARALLELE PROZESSE

Multitasking: Beispielaufruf

```
Function fBlink
  Do
    On 0, 1
    Wait 1.0
  Loop
Fend
```

`Neue Funktion

`Ausgang 0 für 1 Sek. EIN

`1 Sek. warten

```
Function main
  Xqt 6, fBlink

  Motor On
  ...
Fend
```

`Funktion fBlink starten



ACHTUNG

Bei Verwendung eines **Breakpoints**, wird nur der betreffende Task angehalten. Die restlichen Tasks laufen normal weiter.

Tasks können zur Laufzeit angehalten, gestartet und wieder beendet werden.

If Sw(1) = On Then	`Wenn Eingang 1 = On
Halt fBlink	`dann Task Blink anhalten
Else	
Resume fBlink	`sonst Task Blink weiter
	`führen
EndIf	

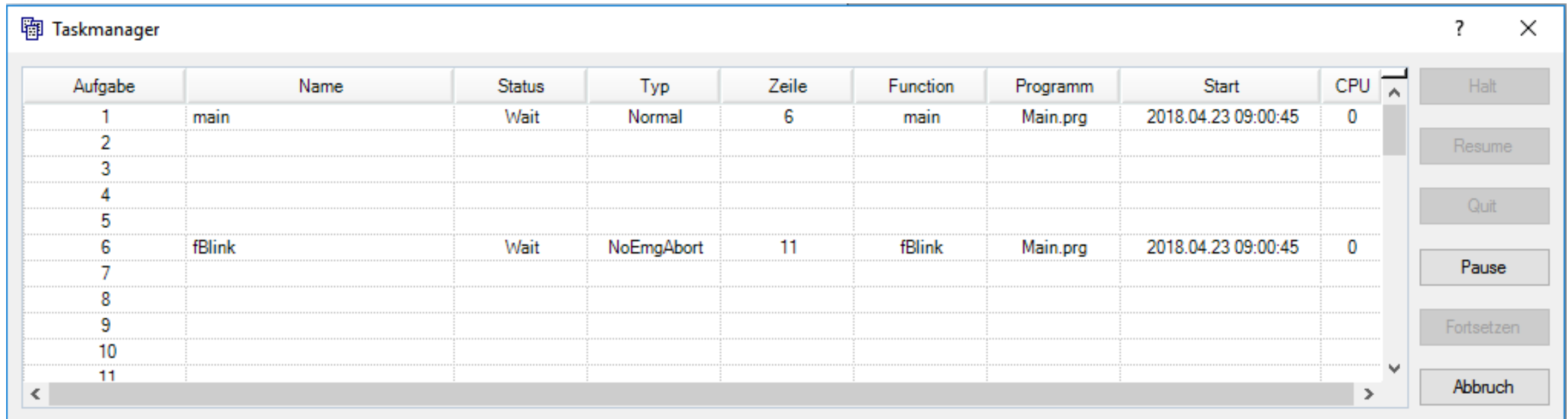
Wenn ein Task mit **HALT** angehalten wird, wird eine eventuell laufende Bewegung des Arms noch zu Ende geführt. Wenn mit **RESUME** der Task fortgeführt wird, startet die Abarbeitung des Programms in der darauffolgenden Programmzeile des Tasks.

Mit `Quit fBlink` wird der Task gelöscht / abgebrochen.

MULTITASKING & PARALLELE PROZESSE

Task-Manager

Alternativ kann im Taskmanager (Menü „Tools“ – „Taskmanager“) alle laufenden und angehaltenen Tasks beobachtet und gesteuert werden:



The screenshot shows the Windows Task Manager window. The title bar reads "Taskmanager" with a question mark and a close button. The main area contains a table with the following columns: Aufgabe, Name, Status, Typ, Zeile, Function, Programm, Start, and CPU. The table lists two tasks: 'main' (Task 1) and 'fBlink' (Task 6). To the right of the table is a vertical scrollbar. On the far right, there is a panel with six buttons: Halt, Resume, Quit, Pause, Fortsetzen, and Abbruch.

Aufgabe	Name	Status	Typ	Zeile	Function	Programm	Start	CPU
1	main	Wait	Normal	6	main	Main.prg	2018.04.23 09:00:45	0
2								
3								
4								
5								
6	fBlink	Wait	NoEmgAbort	11	fBlink	Main.prg	2018.04.23 09:00:45	0
7								
8								
9								
10								
11								

MULTITASKING & PARALLELE PROZESSE

Unterschiedliche Task-Typen

Bei den unterschiedlichen Betriebszuständen des Roboters reagieren die Tasks unterschiedlich:

Normal-Modus	Gestartete Tasks laufen normal
Safeguard	Alle Tasks stehen in Pause, sofern nicht das Flag „ NoPause “ beim Aufruf verwendet wurde. Ausgenommen hiervon ist der Task mit der Roboter-Bewegung.
Not-Aus	Alle Tasks werden abgebrochen, sofern nicht das Flag „ NoEmgAbort “ verwendet wurde. Ausgenommen hiervon ist der Main-Task sowie der Task mit der Roboter-Bewegung.

Ausgänge werden bei **NOT-AUS abgeschaltet**. In weiterlaufenden Tasks, können diese jedoch zwangsweise weiter gesteuert werden:

ON 0, 1.0, **forced**

Zur Zeitmessung stehen bis **64 Timer** zur Verfügung (Timer-ID 0 bis 63).
Die Verwendung von Timern hat in **zwei Schritten** zu erfolgen:

1. TIMER RÜCKSETZEN

```
TmReset (0)
```

```
`Rücksetzen Timer 0
```

2. AKTUELLE LAUFZEIT DES TIMERS SEIT LETZTEM RESET AUSGEBEN

```
Tmr (0)
```

```
`Aktuelle Zeit Timer 0
```

BEISPIEL: ZEITMESSUNG FAHRBEFEHL

...

```
TmReset (5)
```

```
Jump P0
```

```
Print "Zeit:", Tmr(5), "sek."
```

```
`Reset Timer 5
```

```
`Fahrt zu P0
```

```
`Vergangene Zeit seit
```

```
`Reset ausgeben
```

...

Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go P_End	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump P_End	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Move P_End	Führt einen „Move“ zum Punkt „P_End“ aus.	CP-Fahrbefehl
Pallet ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR	Definition einer Palette Syntax zum Aufruf: Pallet (ID, NestID)	Definitionen
Pallet Outside, ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR	Definition einer Outside-Palette Syntax zum Aufruf: Pallet (ID, NestID in HR, NestID in NR)	
Xqt TaskNr, Funktionsname	Aufruf eines Paralleltasks	Paralleltask



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If Bedingung Then ... Elseif Bedingung Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen
Select Auswahlkriterium Case Kriterium1 ... Case Kriterium2 ... Default ... Send	Selektion durch ein Auswahlkriterium	

Syntax:	Hinweis:	Gruppe:
For ZählerVar = Start To End Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While Bedingung ... Loop	Kopfgesteuerte Do-Loop Schleife	
Do ... Loop Until Bedingung	Fußgesteuerte Do-Loop Schleife	



Befehlsübersicht

Local, Offset, Boxen & Planes

Syntax:	Hinweis:	Gruppe:
Go Punkt + U (Wert)	Verändert den „U-Wert“ Merke: Die Punktinformationen bleiben unberührt!	Offset
Go Punkt - Z (Wert)	Verändert den „Z-Wert“ Merke: Die Punktinformationen bleiben unberührt!	
Go Punkt : Z (Wert)	Für die Zeile in der dieser Befehl steht, wird dieser angegebene „Z-Wert“ für den gewünschten Punkt verwendet. Merke: Die Punktinformationen bleiben unberührt!	
Punkt = XY (X_Wert, Y-Wert, Z_Wert, U_Wert)	Die im Punkt vorhanden Werte werden mit Hilfe der Punktmanipulation überschrieben.	Punktmanipulation
Box BoxID, RobotNr, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax, Logik	Definition einer Box. Mit Logik ist „Inside“ oder „Outside“ gemeint.	Boxen
Plane PlaneID, RobotNr, XY (X, Y, Z, U, V, W)	Erzeugt eine Plane an dem definierten Punkt	Plane
Plane PlaneID, RobotNr, PlanePunkt		



AUFGABENSTELLUNG

- Erweiterung des bisherigen Programms um eine weitere Palette (inkl Typumschaltung)
- Speichern Sie die belegten Paletten-Nester
- Ergänzen Sie eine Abfrage ob sich der Roboter im Bereich einer Palette befindet. Übermitteln Sie diese Information mit einem Paralleltask
-> Sperrfunktionalität für weitere Bearbeitungsschritte der SPS (Verwenden Sie Merker, Paralleltask und Boxen)

Optional:

- Messung der Gesamttaktzeit für einen kompletten Palettendurchlauf



→ **Viel Erfolg**

Aufgabenstellung: LOKALES KOORDINATENSYSTEM

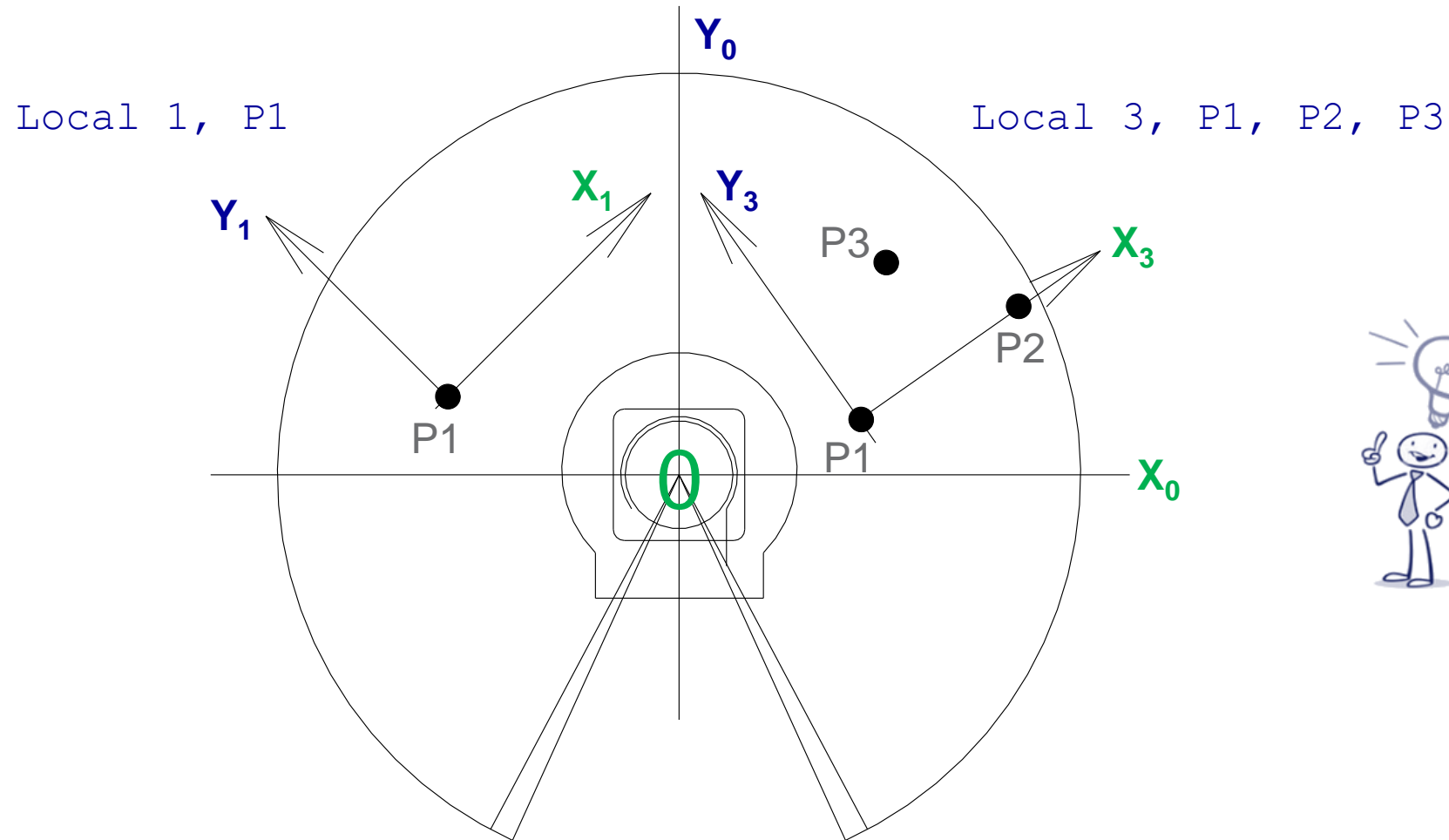


AUFGABENSTELLUNG

- Inbetriebnahme von zwei lokalen Koordinatensystemen

→ Was wird hierzu benötigt?

Lokale Koordinatensysteme werden, ähnlich wie Paletten, auch über **drei Punkte** definiert.



INITIALISIERUNG

Positionieren Sie die LOCAL-Definition in Ihrer Initialisierungsroutine.

TOOL- & LOCAL-KOORDINATENSYSTEME

Local-Koordinatensystem

Neben dem Basiskoordinatensystem des Roboters können durch den Anwender weitere 15 Koordinatensysteme definiert werden.

Local 0 Basiskoordinatensystem des Roboters (fest)

Local 1 bis Local 15 Frei definierbar

The screenshot shows the 'Locals' configuration window in the EPSON robot software. On the left is a sidebar with navigation buttons: Schaltpult, Einrichten, Punkte, Arch, Locals (selected), Tools, Arms, ECP, Boxen, Planes, Weight, Inertia, XYZ- Limits, Range, and Home-Konfig. The main area has a dropdown menu for 'Roboter:' set to '1, robot1, H8-551S'. Below this, the 'Locals' section contains the instruction 'Lokale Koordinatensysteme innerhalb des Basiskoordinatensystems definieren.' and two options: 'Locals mit Assistenten definieren' (with a 'Local-Assistent...' button) and 'Locals manuell definieren'. The 'Locals manuell definieren' section features a table with 9 rows (numbered 1 to 9) and 7 columns (labeled X, Y, Z, U, V, W). Row 1 is highlighted in blue. To the right of the table are four buttons: 'Übernehmen', 'Wiederherstellen', 'Voreinstellungen', and 'Löschen'.

Local	X	Y	Z	U	V	W
1						
2						
3						
4						
5						
6						
7						
8						
9						

SYNTAX UMSCHALTUNG KOORDINATENSYSTEM

```
Jump pAblegen /3 C0
```

Hinter „/“ wird die Nummer des Koordinatensystems angegeben, auf welches die in pAblegen angegebenen Koordinaten angewendet werden sollen. Die Umschaltung kann wahlweise auch mit einer **Variablen** erfolgen. Hierzu verwenden Sie dann bitte folgende Syntax:

```
Jump pAblegen /(iLocalID) C0
```

UMRECHNUNG ZWISCHEN LOKALEN KOORDINATENSYSTEMEN

```
P10 = P10 @3
```

(anschließend Savepoints-Befehl nicht vergessen!)

BOXEN & PLANES

Boxen können auch in einem lokalen Koordinatensystem verwendet werden:

```
Box 1, 1, -200, 200, 400, 600, -150, 0, On /LOCAL1
```

Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go <i>P_End</i>	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump <i>P_End</i>	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Move <i>P_End</i>	Führt einen „Move“ zum Punkt „P_End“ aus.	CP-Fahrbefehl
Pallet <i>ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR</i>	Definition einer Palette Syntax zum Aufruf: <i>Pallet (ID, NestID)</i>	Definitionen
Pallet Outside, <i>ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR</i>	Definition einer Outside-Palette Syntax zum Aufruf: <i>Pallet (ID, NestID in HR, NestID in NR)</i>	
Xqt <i>TaskNr, Funktionsname</i>	Aufruf eines Paralleltasks	Paralleltask



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If Bedingung Then ... Elseif Bedingung Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen
Select Auswahlkriterium Case Kriterium1 ... Case Kriterium2 ... Default ... Send	Selektion durch ein Auswahlkriterium	

Syntax:	Hinweis:	Gruppe:
For ZählerVar = Start To End Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While Bedingung ... Loop	Kopfgesteuerte Do-Loop Schleife	
Do ... Loop Until Bedingung	Fußgesteuerte Do-Loop Schleife	



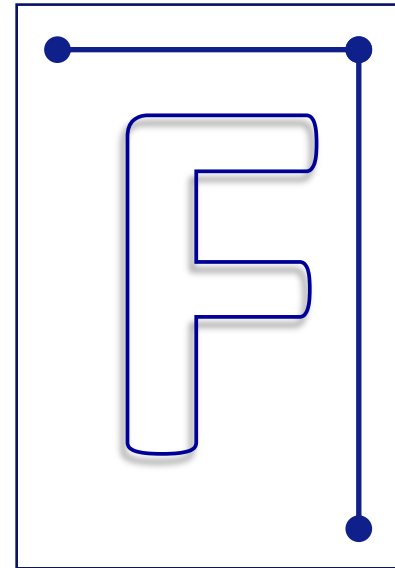
Befehlsübersicht

Local, Offset, Boxen & Planes

Syntax:	Hinweis:	Gruppe:
Go Punkt + U (Wert)	Verändert den „U-Wert“ Merke: Die Punktinformationen bleiben unberührt!	Offset
Go Punkt - Z (Wert)	Verändert den „Z-Wert“ Merke: Die Punktinformationen bleiben unberührt!	
Go Punkt : Z (Wert)	Für die Zeile in der dieser Befehl steht, wird dieser angegebene „Z-Wert“ für den gewünschten Punkt verwendet. Merke: Die Punktinformationen bleiben unberührt!	
Punkt = XY (X_Wert, Y-Wert, Z_Wert, U_Wert)	Die im Punkt vorhanden Werte werden mit Hilfe der Punktmanipulation überschrieben.	Punktmanipulation
Box BoxID, RobotNr, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax, Logik	Definition einer Box. Mit Logik ist „Inside“ oder „Outside“ gemeint.	Boxen
Plane PlaneID, RobotNr, XY (X, Y, Z, U, V, W)	Erzeugt eine Plane an dem definierten Punkt	Plane
Plane PlaneID, RobotNr, PlanePunkt		
Local LocalID, Ursprung, X-Orientierung, Vorgabe der Y+ Richtung	LocalID = 0 ist nicht möglich (Basiskoordinatensystem) Mit Hilfe des Local können Hilfskoordinatensysteme erstellt werden.	Local



- Umbau des Tools auf die **Saugereinheit** und schalten Sie auf das **Tool 0** im Roboter Manager zurück.
- Erstellen Sie ein lokales Koordinatensystem mit folgendem Befehl im **Befehlseingabefenster** **LOCAL 1, P10, P11, P12** (Punktnummern bitte anpassen!) und teachen Sie die Bahnpunkte im neuen Koordinatensystem.
Erstellen Sie ein kleines Verfahrprogramm um die Eckpunkte jeweils mit einem MOVE zu verbinden.
- Nehmen Sie eine zweite Vorlage.
- Erstellen Sie ein zweites Koordinatensystem.
LOCAL 2, P20, P21, P22
(P20-P22 sind jedoch weiterhin im **Local 0** zu teachen!)
- Passen Sie Ihr Programm auf das Local 2 an.
- Fahren Sie die Bahn erneut ab.



Aufgabenstellung:

2. GREIFER



AUFGABENSTELLUNG

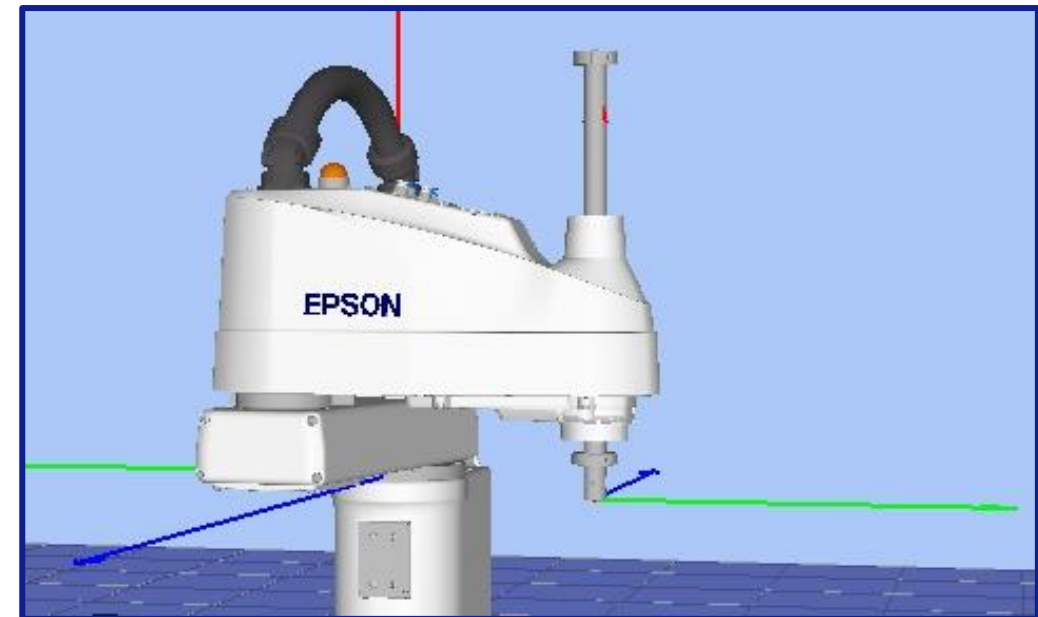
- Mechanischer Umbau auf Doppelgreifer
- Abholung mit zwei Greifern von der Rutsche
- Absetzen mit zwei Greifern im Tray

→ **Was wird hierzu benötigt?**

TOOL- & LOCAL-KOORDINATENSYSTEME

Tool-Koordinatensystem

- Der Tool-Center-Point (TCP) ist der **Referenzpunkt am Robotergreifer**.
- Auf diesen Punkt beziehen sich die Koordinaten des **verwendeten Koordinatensystems**.
- Der TCP hat sein eigenes Koordinatensystem das sog. **Werkzeug-koordinatensystem**.
- Es können **bis zu 16 TCPs** verwendet werden.
- Die TCPs 1 bis 15 können **frei definiert** werden.
- Als **Standard-TCP (0)** ist die Mitte der Kugelumlaufspindel fest definiert.



TOOL- & LOCAL-KOORDINATENSYSTEME

Tool-Koordinatensystem

- Teachen aller Positionen mit nur **einem TCP**. Anschließend können alle Positionen mit allen TCPs angefahren werden.
- Es muss beim Teachen von Paletten nicht auf die Drehlage des Greifers geachtet werden, um das korrekte Raster zu bestimmen. (U-Orientierung wird nur vom **ersten Palettenpunkt**, dem Ursprungspunkt verwendet).
- Änderung des Werkzeuganstellwinkels **ohne** die Position zu verlassen.



TOOL- & LOCAL-KOORDINATENSYSTEME

Tool-Koordinatensystem

Über den Tool-Assistenten kann auf einfache Weise jedes einzelne Tool definiert werden. Schritt für Schritt erfolgt die Konfiguration des verwendeten Tools.

Schaltpult
Einrichten
Punkte
Arch
Locals
Tools
Arms
ECP
Boxen
Planes
Weight
Inertia
XYZ- Limits
Range
Home-Konfig

Roboter: 1, robot1, H8-551S

Tools
Koordinatensystem des Greifers definieren.

Tools mit dem Assistenten definieren
Tool-Assistent...

Tools manuell definieren

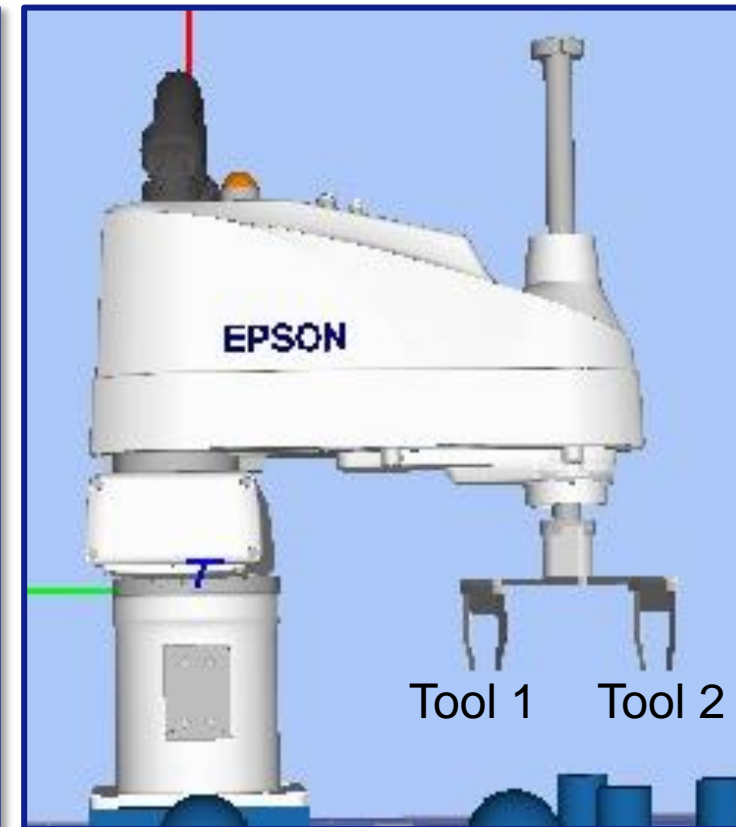
Tool	X	Y	Z	U	V	W
1						
2						
3						
4						
5						
6						
7						
8						
9						

Übernehmen

Wiederherstellen

Voreinstellungen

Löschen



`Position entladen mit Tool 1 (Greifer 1)

```
Tool 1  
Jump pAblage C0  
On outGreifer1  
Wait 0.1
```

`Position beladen mit Tool 2 (Greifer 2)

```
Tool 2  
Jump pAblage C0  
Off outGreifer2  
Wait 0.1
```

VARIABLE ZUWEISUNG:

Die Umschaltung des aktiven Tools kann auch mit einer Variablen erfolgen:

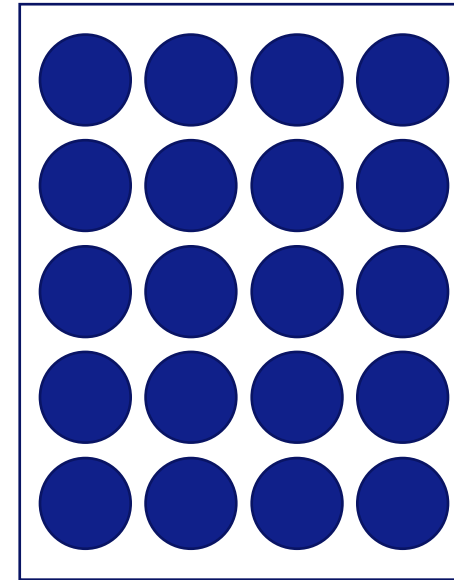
```
Integer iToolID  
iToolID = 2  
Tool iToolID
```



ACHTUNG:

Das zuletzt aktiv verwendete Tool wird auch bei erneutem Programmstart weiterverwendet! Ggf. also in der **Initialisierungs-routine** ein Standardtool beim Programmstart aktivieren.

- Richten Sie über den Tool-Assistenten **beide** Tools des Greifers ein
- **Gemeinsam** passen wir dann die **U-Orientierung** für die Tools an
- Teachen Sie Ihre Palette 1 mit Tool 1
- Passen Sie Ihre **Initialisierungsroutine** entsprechend an
- Schreiben Sie ein neues Ablaufprogramm um mit beiden Tools ein Produkt „simuliert“ an der Rutsche abzuholen und anschließend „simuliert“ in die Palette 1 abzulegen.



Befehlsübersicht

Fahrbefehle, Definitionen & Initialisierung

Syntax:	Hinweis:	Gruppe:
Motor On	Schaltet die Motoren ein.	Systeminitialisierung
Power Low	Versetzt den Manipulator in den LOW-Power Modus.	
Go P_End	Führt einen „Go“ zum Punkt „P_End“ aus.	PTP-Fahrbefehl
Jump P_End	Führt einen „Jump“ zum Punkt „P_End“ aus.	
Move P_End	Führt einen „Move“ zum Punkt „P_End“ aus.	CP-Fahrbefehl
Pallet ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR	Definition einer Palette Syntax zum Aufruf: Pallet (ID, NestID)	Definitionen
Pallet Outside, ID, Ursprung, Endpunkt HR, Endpunkt NR, Nest HR, Nest NR	Definition einer Outside-Palette Syntax zum Aufruf: Pallet (ID, NestID in HR, NestID in NR)	
Xqt TaskNr, Funktionsname	Aufruf eines Paralleltasks	Paralleltask



Befehlsübersicht

Anweisungen & Schleifen

Syntax:	Hinweis:	Gruppe:
If Bedingung Then ... Elseif Bedingung Then ... Else ... EndIf	Selektion durch eine Bedingung	Anweisungen
Select Auswahlkriterium Case Kriterium1 ... Case Kriterium2 ... Default ... Send	Selektion durch ein Auswahlkriterium	

Syntax:	Hinweis:	Gruppe:
For ZählerVar = Start To End Next	Aufbau eines Zählers	Schleife
Do ... Loop	Do-Loop Schleife ohne Abbruchbedingung	
Do While Bedingung ... Loop	Kopfgesteuerte Do-Loop Schleife	
Do ... Loop Until Bedingung	Fußgesteuerte Do-Loop Schleife	



Befehlsübersicht

Local, Offset, Boxen & Planes

Syntax:	Hinweis:	Gruppe:
Go Punkt + U (Wert)	Verändert den „U-Wert“ Merke: Die Punktinformationen bleiben unberührt!	Offset
Go Punkt - Z (Wert)	Verändert den „Z-Wert“ Merke: Die Punktinformationen bleiben unberührt!	
Go Punkt : Z (Wert)	Für die Zeile in der dieser Befehl steht, wird dieser angegebene „Z-Wert“ für den gewünschten Punkt verwendet. Merke: Die Punktinformationen bleiben unberührt!	
Punkt = XY (X_Wert, Y-Wert, Z_Wert, U_Wert)	Die im Punkt vorhanden Werte werden mit Hilfe der Punktmanipulation überschrieben.	Punktmanipulation
Box BoxID, RobotNr, Xmin, Xmax, Ymin, Ymax, Zmin, Zmax, Logik	Definition einer Box. Mit Logik ist „Inside“ oder „Outside“ gemeint.	Boxen
Plane PlaneID, RobotNr, XY (X, Y, Z, U, V, W)	Erzeugt eine Plane an dem definierten Punkt	Plane
Plane PlaneID, RobotNr, PlanePunkt		

