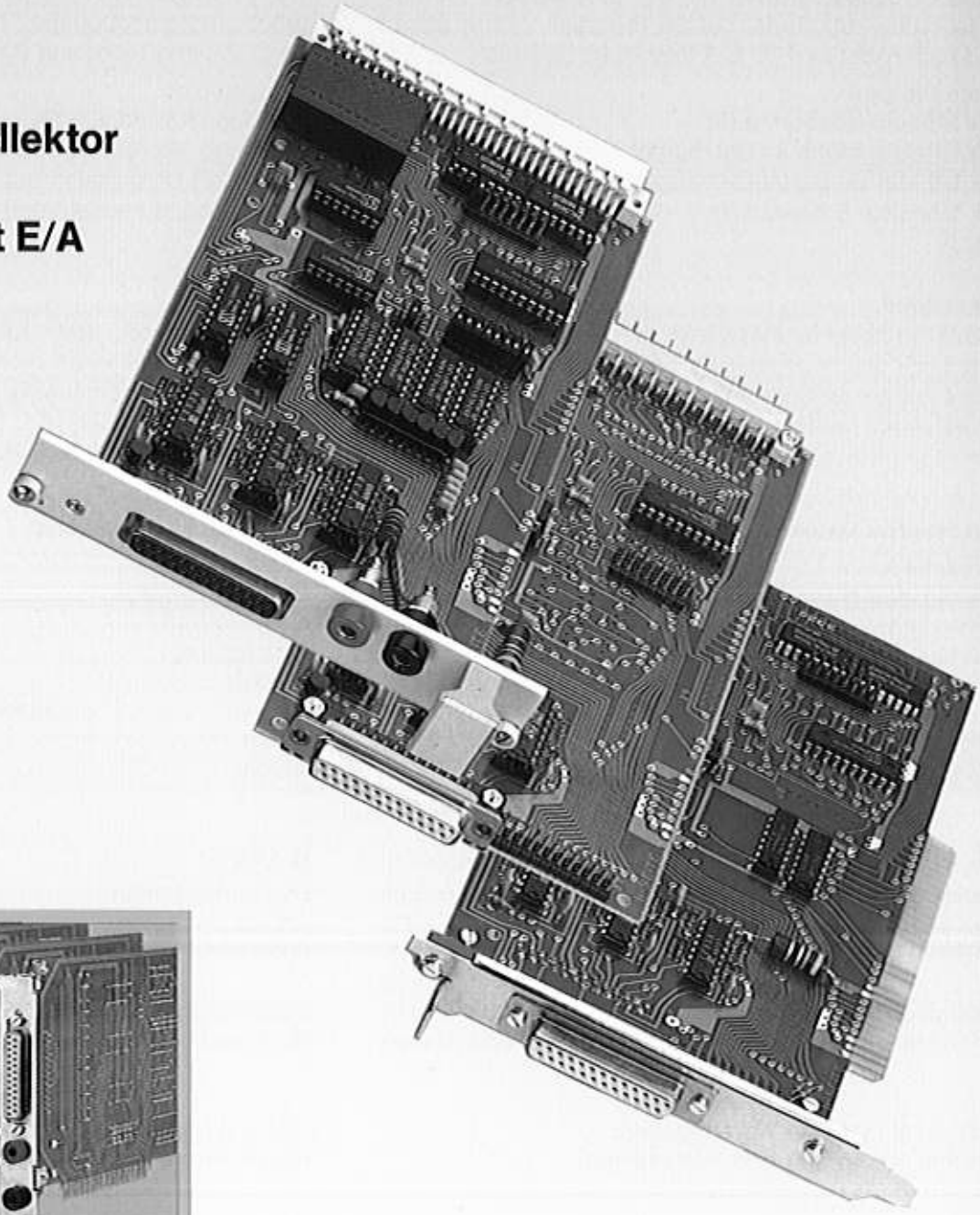
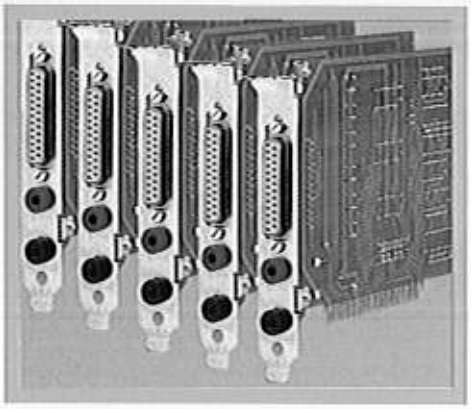
Kalms GmbH
teach-robot®
Made in W. Germany

# Ui 1 / robot

**Steuerkarte für teach-robot, zur Ansteuerung von Servomotoren mit Rückmeldung (inkrementale Geber) oder andere Gleichstromlasten.**
**Für alle XT / AT PC's und kompatible**
**Sonderausführung für das MFA Microprozessor-Entwicklungssystem (Ui 1 / MFA)**

**Ui1/Robot**
**Ui1/Open Kollektor**
**Ui1/Relais**
**Ui1/8 (32) Bit E/A**
**Ui1/Analog**





**microelectronic** Kalms GmbH • Weinstraße 47 A • 77815 Bühl-Eisental • Tel. 0 72 23 / 2 60 93 • Fax: 0 72 23 / 2 04 71

# Ui 1/MFA/PiF/IBM
# Universal-Interface

Über dieses Interface ist TEACH-ROBOT an verschiedene Microprozessor- und Computersysteme anschließbar.

Es enthält alle Komponenten und Schaltungen, die zur Steuerung des Roboters erforderlich sind.

Alle Interfaces werden MIT SOFTWARE geliefert!

Das Ui 1/robot arbeitet mit der universellen Roboter-Steuerung robot-sps. Dieses Programm kann bis zu 8 teach-robot und 32 E/A Interfaces bedienen.

zum Beispiel:
Ui 1/Open Kollektor, 8 Bit
Ui 1/Relais, 4fach 4 x UM, 500 W
Ui 1/8 (32) Bit digital Ein/Ausgang
Ui 1/Analog, 8 Kanal E/A, 0 – 10 Volt
(end)

**Ui 1/robot**

Direkt steckbar für IBM XT/AT oder Kompatible. Die Programmierung ist einfach und über eine Bildschirmmaske mit Anzeige der Soll/Ist Positionen leicht zu handhaben. Programme lauffähig ab MS-DOS 2.0.

**Technische Daten:**

| Allgemein: | (für alle Typen gleich) |
|---|---|
| Betriebsspannung | 5 Volt |
| Nennstrom ca | 0,3 Amp. |
| Format | 100 x 160, Europakarte |
| Ausgänge | Stecker Teach-Robot (25 pol. D-SUB-Stecker, Kabel 1 :1) |
| Eingänge | 6 x TTL (Puls) |
| Gewicht | 160 Gramm |

| **Ui 1/MFA:** | (zusätzlich) |
|---|---|
| Busstruktur | MFA-System-Bus |
| Adressdecodierung | 01 – FF (hex) |
| Pulserfassung | 8 Bit Zähler (od. parallel direkt über MFA-Bus) |
| Stromversorgung f. Robot | 12 – 20 V, extern |
| Frontplatte | Alu, 25 mm breit |

IBM ist ein eingetr. Warenzeichen.
Rechte, Irrtum und Änd. vorbehalten.

# Ui/MFA/PiF/IBM
# Universal-interface 1

TEACH-ROBOT can be connected to various microprocessor developement systems and personal computers.

It contains all circuits required for moving and controlling this robot.

All interfaces will be delivered including software!

The Ui 1/robot works with together with the universal robot-program robot-sps. This programm is able to drive up to 8 teach-robot and 32 I/O interfaces

for example:
Ui 1/open kollektor, 8 Bit
Ui 1/relais, 4 x 4pole switch, 500 W
Ui 1/8 (32) Bit digital in/out
Ui 1/analog, 8 chanel i/o, 0 – 10 volts
(end)

**Ui 1/robot**

Plug-in for all IBM XT/AT or compatibles. An <INFORMATION MASK> on the screen guides the user through easy programming. Actual and destination positions are also displayed.
Program works with MS-DOS 2.0 or later.

**Technical specifications:**

| Common: | (all models) |
|---|---|
| Operating voltage | 5 volt |
| Nom. current | 0,3 amp. |
| Dimensions | 3,94 x 6,30 inch |
| Outputs | connector TEACH-ROBOT (25 pol. D-SUB-female, lead 1: |
| Inputs | 6 x TTL (pulse) |
| Weight | 160 gramm |

| **Ui 1/MFA:** | (additional) |
|---|---|
| Bus configuration | MFA-System |
| Adress decoding | 01 – FF (hex) |
| Pulse recording | 8 bit counter (or parallel direct via MFA-bus) |
| Power supply for robot | 12 – 20 V, external |
| Rear panel | Alu, width 1 inch |

IBM is a registered trade mark.
Rights, errors and changes reserved.

**microelectronic** Kalms GmbH • Weinstraße 47 A • 77815 Bühl-Eisental • Tel. 0 72 23 / 2 60 93 • Fax: 0 72 23 / 2 04 71

## 2.2.1 Ui1/Robot, Interface für TEACH-ROBOT 2.2.1

Steuerkarte für TEACH-ROBOT.
Eingänge: 6 Puls- plus 2 Sensorsignale.
Ausgänge: 6 Vollbrücken für Gleistrom-Motoren. (je 7 Watt)

Funktionstabelle: (hexadezimal)

| Motor | Funktion | links | rechts | stop | Pulse/B-Bus |
|---|---|---|---|---|---|
| 1 | Finger | C1 | C8 | 89 | d0 |
| 2 | Hand-Dreh | C2 | D0 | 92 | d1 |
| 3 | Hd-Schwk | C3 | D8 | 9B | d2 |
| 4 | Arm | C4 | E0 | A4 | d3 |
| 5 | Schulter | C5 | E8 | AD | d4 |
| 6 | Körper | C6 | F0 | B6 | d5 |
| 7 | nicht belegt | | | | Sens 1 d6 |
| | | | | | Sens 2 d7 |

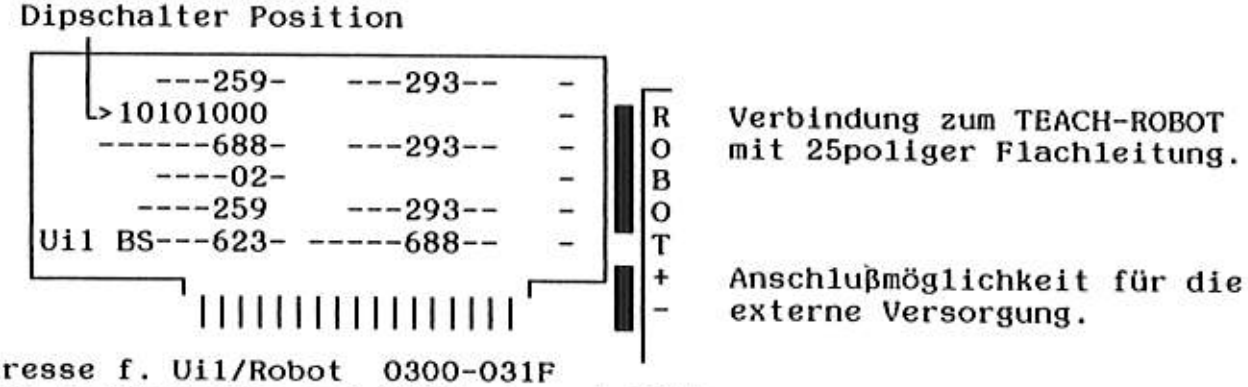
Diese Hex-Konfigurationen auf dem Datenbus rufen die verschiedenen Roboterbewegungen hervor.

Die Eingänge Sens 1 und Sens 2 sind NUR für TTL-Signale geeignet.

Das Interface hat einen Eingang für die externe Stromversorgung!

Abmessungen:
Platine, 100x160mm, direkt steckbar, oder ohne Kamm für MFA-Bus.

KARTENADRESSE:




Adresse f. Ui1/Robot 0300-031F
(bei Auslieferung auf 0315 eingestellt)

HINWEIS:
Gesetzt werden nur A0 bis A7 (03 ist fest verdrahtet!)

| Adresse: | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
|---|---|---|---|---|---|---|---|---|
| Schalter: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Zustand: | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | off | on | off | on | off | on | on | on |

Versorgungsspannung für den inkrementalen Geber

Bürstenanschluß plus

Bürstenanschluß minus

Ground, Masse, 0 Volt

TTL-Pegel, 3,3 Volt, 5 mA mit 220 Ohm entkoppelt von der Antriebsseite nicht belegt, frei.

Motorstecker 1-6:

| | |
|---|---|
| 6 | + 5 Volt |
| 5 | Motoranschluß + |
| 4 | Motoranschluß - |
| 3 | Masse (0 Volt) |
| 2 | Pulsausgang |
| 1 | frei für Optionen |

ACHTUNG:

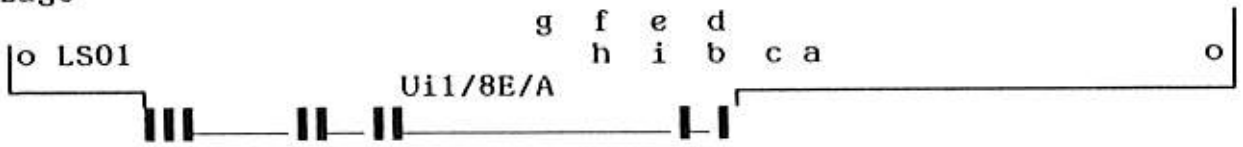
Sens 1+2 (TTL) ist beim Motorstecker 1+2 auf Pin 1 geschaltet. Hier stehen also 2 zusätzliche TTL-Eingänge zur Abfrage von Sensorsignalen zur Verfügung.

Die Anschlußkonfiguration: (für TEACH-ROBOT)

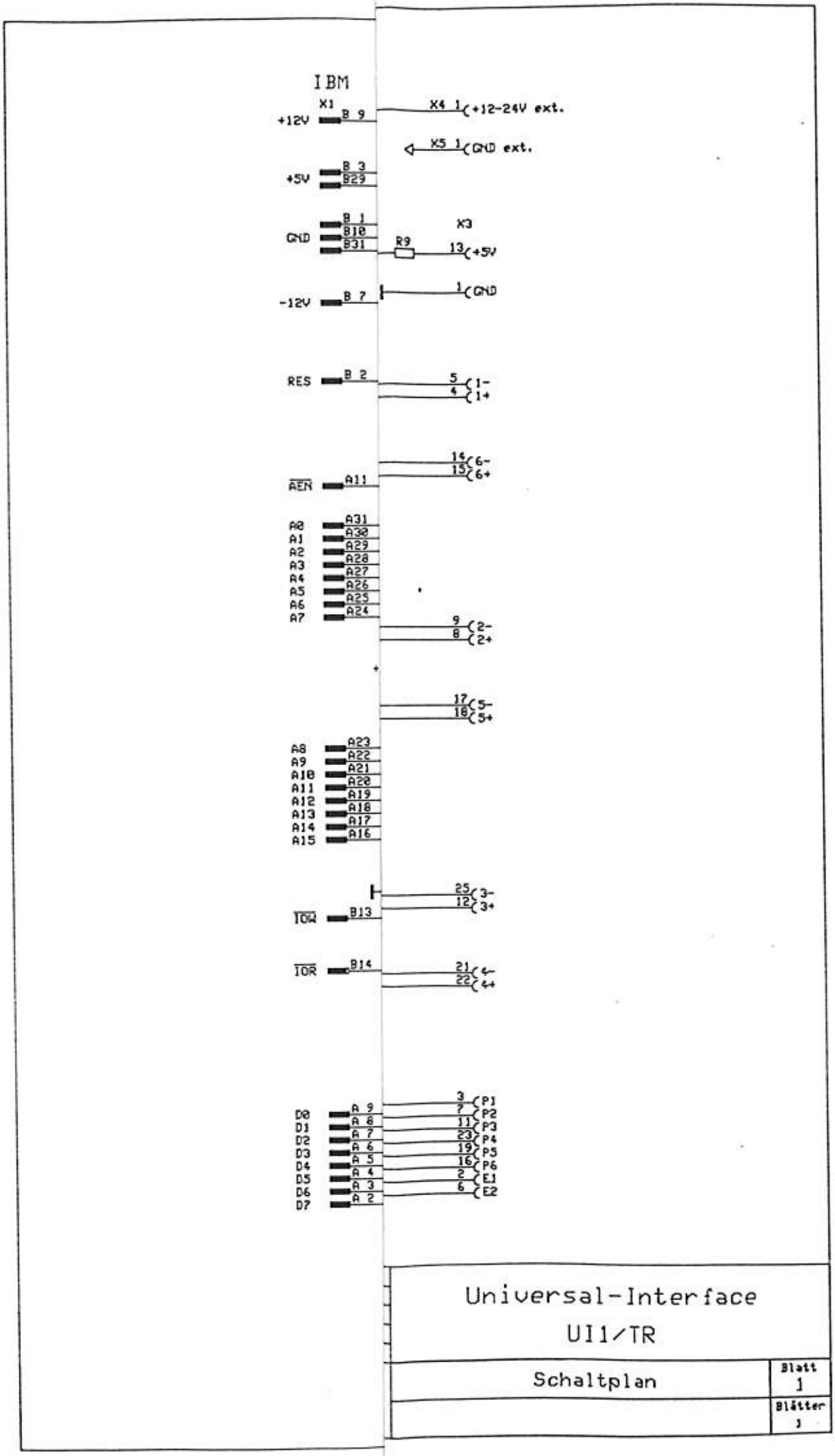
D-SUB 25 polig

| | | | |
|---|---|---|---|
| + 5 Volt | 13 | | |
| + Motor 3 | 12 | 25 | - Motor 3 |
| Puls 3 | 11 | 24 | frei bei M 4 |
| frei bei M 3 | 10 | 23 | Puls 4 |
| - Motor 2 | 9 | 22 | + Motor 4 |
| + Motor 2 | 8 | 21 | - Motor 4 |
| Puls 2 | 7 | 20 | frei bei M 5 |
| Sens 2, TTL in, d7 | 6 | 19 | Puls 5 |
| - Motor 1 | 5 | 18 | + Motor 5 |
| + Motor 1 | 4 | 17 | - Motor 5 |
| Puls 1 | 3 | 16 | Puls 6 |
| Sens 1, TTL in, d6 | 2 | 15 | + Motor 6 |
| Masse | 1 | 14 | - Motor 6 |

LÖTBRÜCKEN:
Die Lötbrücken b-i unter dem IC1 (74688) müssen für PC-Betrieb (IBM) geöffnet sein! Dies ist bei Auslieferung sichergestellt.
Lage




HINWEIS:
2 freie Ausgangssignale an den beiden 8 Bit-Latches (74LS259), jeweils QH, sind in der Standardversion ungenutzt und könnten mit Drahtbrücken an die freien Pins (frei bei M 3-5) gelegt werden. Wenn Sie mit einem der Signale einen magnetischen Parallelgreifer ansteuern würden (über eine kleine externe Leistungsstufe), dann wäre der Antrieb 1 (bisher Finger), für eine mögliche 6. Achse frei.


IBM
X1
+12V
B 9
X4 1 (+12-24V ext.
X5 1 (GND ext.
+5V
B 3
B29
GND
B 1
B18
B31
X3
R9
13 (+5V
-12V
B 7
1 (GND
RES
B 2
5 (1-
4 (1+
14 (6-
15 (6+
AEN
A11
A0
A1
A2
A3
A4
A5
A6
A7
A31
A30
A29
A28
A27
A26
A25
A24
9 (2-
8 (2+
17 (5-
18 (5+
A8
A9
A10
A11
A12
A13
A14
A15
A23
A22
A21
A20
A19
A18
A17
A16
25 (3-
12 (3+
IOW
B13
IOR
B14
21 (4-
22 (4+
D0
D1
D2
D3
D4
D5
D6
D7
A 9
A 8
A 7
A 6
A 5
A 4
A 3
A 2
3 (P1
7 (P2
11 (P3
23 (P4
19 (P5
16 (P6
2 (E1
6 (E2
Universal-Interface
UI1/TR
Schaltplan
Blatt
1
Blätter
1

## 2.2.2 Ui1/OC, Interface Ui1/Open Kollektor 2.2.2

8 Leistungsausgänge Open Kollektor mit einer Schaltleistung von max. 30 W je Ausgang.

### Technische Daten

Stromversorgung:

```
Logic:  UB     =  5 Volt (wird vom PC gespeist)
        Imax   = 70 mA   (C-Mos Logic)
Optokoppler:  UB     = 12 Volt (wird vom PC gespeist)
              Imax   = 80 mA   (alle 8 Transistoren angesteuert)
```

Ausgänge Open Kollektor:

UB max = 48 Volt DC, Ic max = 2 Amp, Pmax = 30 W (P = UxI)
Je 4 Emitter sind auf eine gemeinsame Masse geführt.
(GND OC1-4 Pin 11, und GND OC 5-8, Pin 12)

Schutzmaßnahmen:

OPTOKOPPLER schützen die Logic und den PC vor Rückwirkungen.
TRANSILDIODEN schützen die Endstufen vor Spannungen > 51 Volt.
FREILAUFDIODEN ermöglichen auch den Anschluß von induktiven Lasten. Jeweils 4 dieser Freilaufdioden werden zusammengefaßt auf die Anschlüsse COM 1(Pin 1) und COM 2(Pin 6).

Meßpunkte:

8 Widerstände, gespeist von Uext (13), am Ausgang der Open-Kollektor-Ausgänge, durch Dioden entkoppelt, ermöglichen die Ermittlung der logischen Zustände der Endstufen.

Programmierung:

Die Obergabe von Daten erfolgt mit den Befehlen STP und STB.

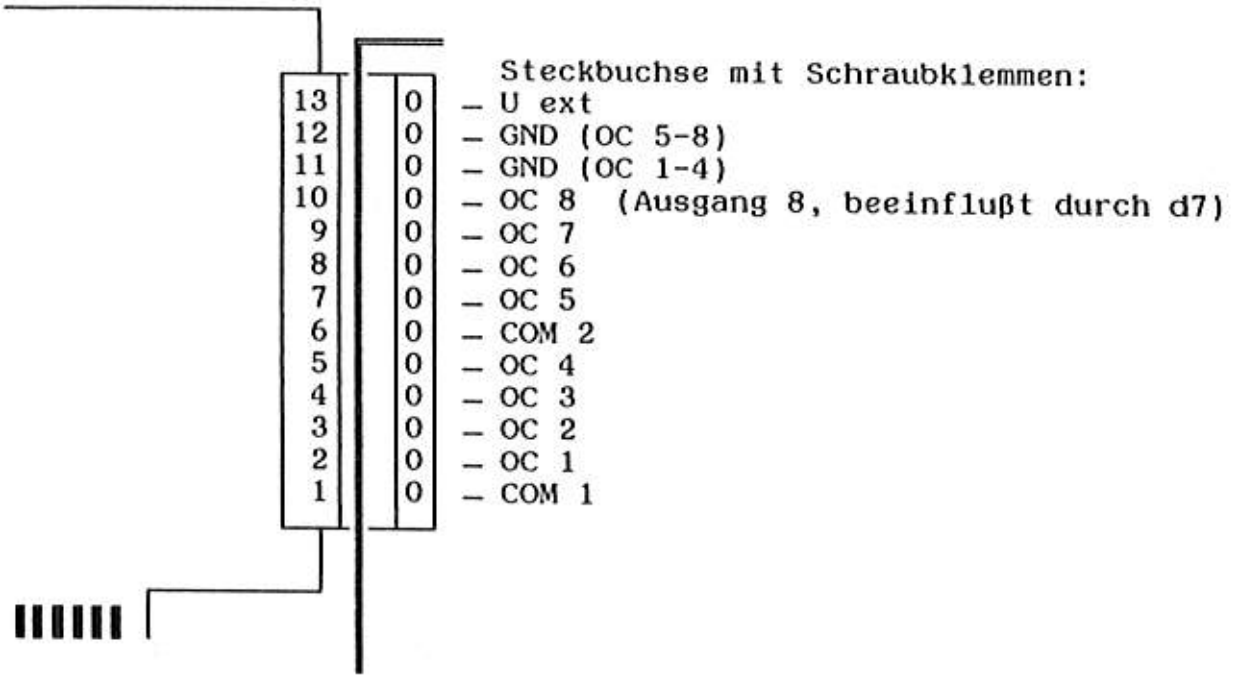
```
Beispiel 1:      Ausgabe von hex 55
     ld $55     ;lade Akku mit hex 55
    stp 1       ;schreibe Akku nach Port 1
;die Zustände an den OC-Ausgängen = 01010101
;                                  oc8....oc1

Beispiel 2:      Ausgabe von Bit 1 (d0) des Akkuinhalts
     ld 255     ;Akku mit bin 11111111 laden
     stb 1,1    ;Bit 1 des Akkus nach Port 1 setzen
;die Zustände an den OC-Ausgängen  = 00000001 (vorher 00000000)
```

ACHTUNG:

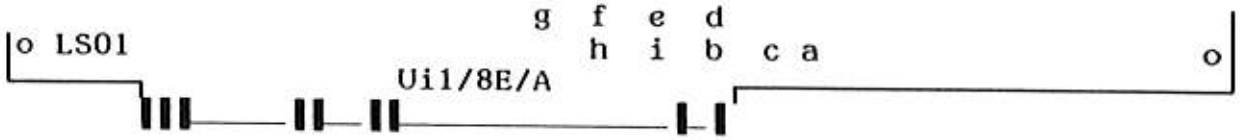
Für 8 Bit Parallel-Ausgabe muß die Port-Adresse 83xx sein!!!

## 2.2.2.1 Schaltbild, Ui1/Open Kollektor 2.2.2.1

Die Anschlußkonfiguration:

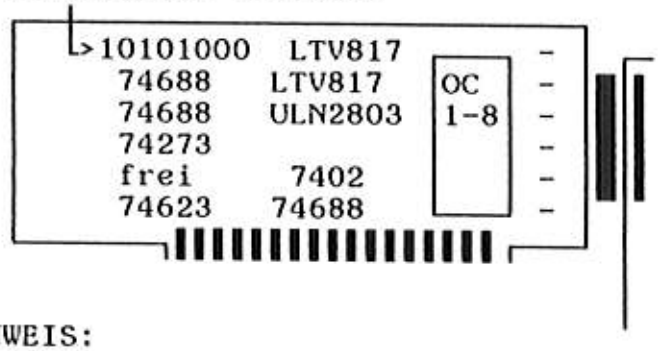



LÖTBRÖCKEN:
Die Lötbrücken b-i unter dem IC1 (74688) müssen für PC-Betrieb (IBM) geöffnet sein! Dies ist bei Auslieferung sichergestellt.
Lage




KARTENADRESSE:

Dipschalter Position




HINWEIS: Die Kartenadresse ist bei der Auslieferung auf hex 0315 eingestellt!

HINWEIS:
Gesetzt werden nur A0 bis A7 (03xx ist fest verdrahtet!)

| Adresse: | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
|---|---|---|---|---|---|---|---|---|
| Schalter: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Zustand: | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | off | on | off | on | off | on | on | on |

## 2.2.3 Ui1/REL, Interface Ui1/Relais 2.2.3

### Technische Daten

Stromversorgung:

```
Logic:  UB    =  5 Volt (wird vom PC gespeist)
        Imax  = 70 mA   (C-Mos Logic)
Relais: UB    = 12 Volt (wird vom PC gespeist)
        Imax  = 65 mA   (alle Relais aktiv)
```

Ausgänge, Relaiskontakte:

```
Belastbarkeit,  max 220 Volt, 50 Hz, 2 Amp
                max  40 Volt, DC, 2,5 Amp
Anzugs- und Abfallzeiten ca.  5 ms
```

Schutzmaßnahmen:

Durch die galvanische Trennung der Arbeitskontakte von der Erregerseite, sind weiter gehende Schutzmaßnahmen nicht erforderlich.

Programmierung:

Die Obergabe von Daten erfolgt mit den Befehlen STP und STB.

```
Beispiel 1:      Ausgabe; Schließe die Kontakte 2/3 bei Relais 1
     ld 01      ;lade Akku mit 01
     stp 1       ;schreibe Akku nach Port 1
;das Relais 1 hat geschaltet. Die Zustände an den Ausgängen sind:
;Kontakte 1/2 geöffnet, Kontakte 2/3 geschlossen.
```

Ladebefehle für 1 - 4:

```
     ld 1       ;Relais 1       (0000 0001)
     ld 2       ;Relais 2       (0000 0010)
     ld 4       ;Relais 3       (0000 0100)
     ld 8       ;Relais 4       (0000 1000)
```

HINWEIS:
Aus Gründen der Sicherheit wird, beim Einschalten der Versorgungsspannung (dies gilt für alle Interfacekarten), ein Zwangsreset ausgelöst. Die Ausgänge werden inaktiv.

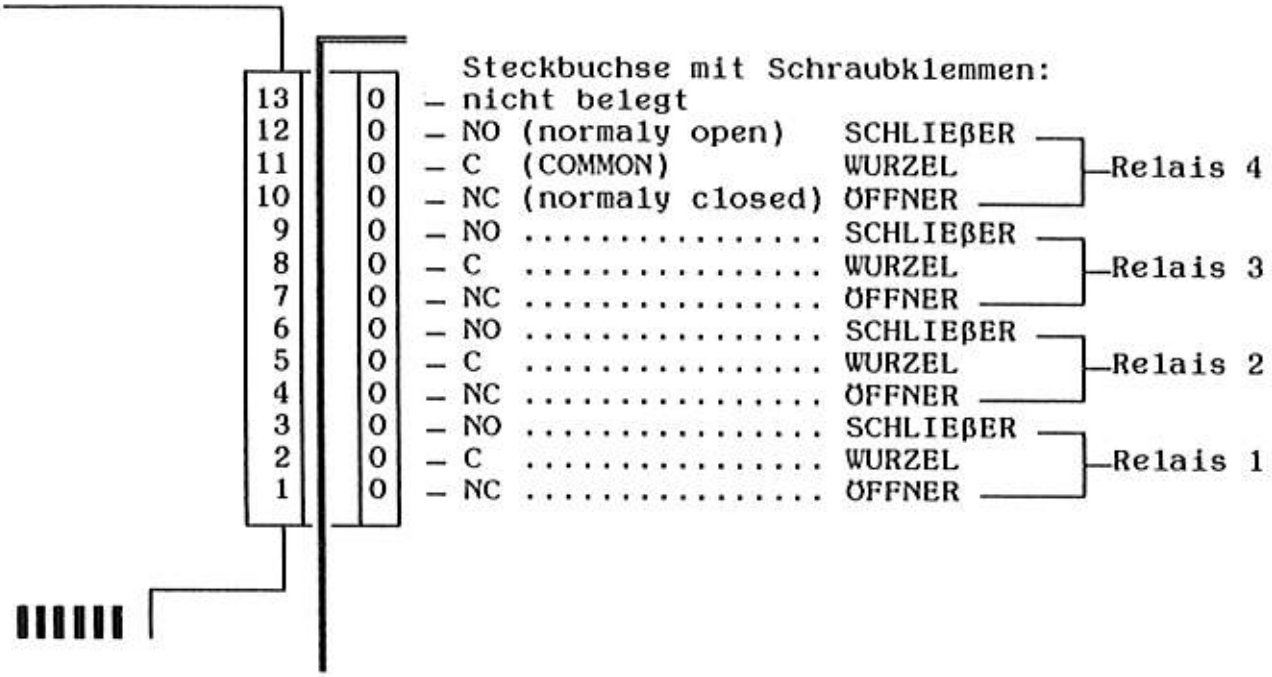
Es empfiehlt sich dennoch im Programmteil INTIALISIERUNG eine Löschroutine zu schreiben (auzurufen).

ACHTUNG:

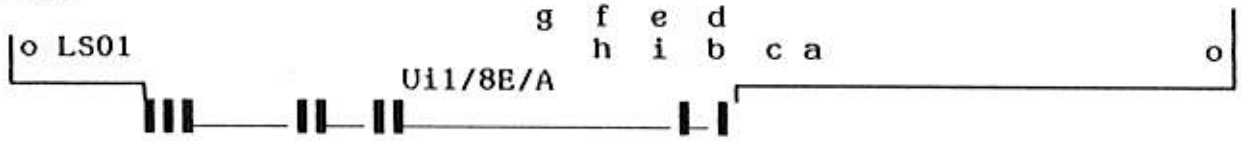
| Für Ansteuerung der Ui1/REL muß die Port-Adresse 83(15) sein! |
|---|

## 2.2.3.1 Schaltbild, Ui1/Relais 2.2.3.1

Die Anschlußkonfiguration:

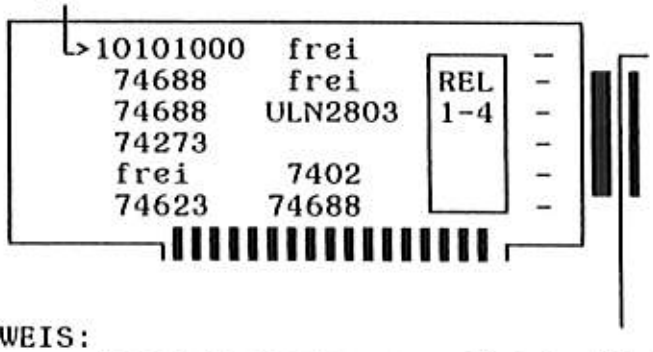



JUMPER/LÖTBRÖCKEN:
Die Lötbrücken b-i unter dem IC1 (74688) müssen für PC-Betrieb (IBM) geöffnet sein! Dies ist bei Auslieferung sichergestellt.
Lage




KARTENADRESSE:

Dipschalter Position




HINWEIS: Die Kartenadresse ist bei der Auslieferung auf hex 0315 eingestellt!

HINWEIS:
Gesetzt werden nur A0 bis A7 (03xx ist fest verdrahtet!)

| Adresse: | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
|---|---|---|---|---|---|---|---|---|
| Schalter: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Zustand: | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | off | on | off | on | off | on | on | on |

2.2.4 Ui1/8 E/A, Interface Ui1/8 Bit Eingang und 8 Bit Ausgang 2.2.4

## Technische Daten

Stromversorgung:

Logic: UB = 5 Volt (wird vom PC gespeist)
Imax = 70 mA (C-Mos Logic)
Optokoppler: UB = 12 Volt (wird vom PC gespeist)
Imax = 80 mA (alle 8 Transistoren angesteuert)

ACHTUNG:

Wegen der galvanischen Trennung durch Optokoppler müssen GND und +5 Volt extern zugeführt werden!

Ausgänge: TTL- Pegel Eingänge: TTL-Pegel

Schutzmaßnahmen:

OPTOKOPPLER schützen die Logic und den PC vor Rückwirkungen.

HINWEIS:
Die Anschlußkonfiguration entspricht im Wesentlichen der einer Centronics-Schnittstelle (Druckeranschluß).

Programmierung:

Die Ausgabe von Daten erfolgt mit den Befehlen STP und STB.

```
Beispiel 1:      Ausgabe von hex 55
     ld $55     ;lade Akku mit hex 55
     stp 1      ;schreibe Akku nach Port 1
;die Zustände an den Ausgängen = 01010101
;                                d7.....d0
```

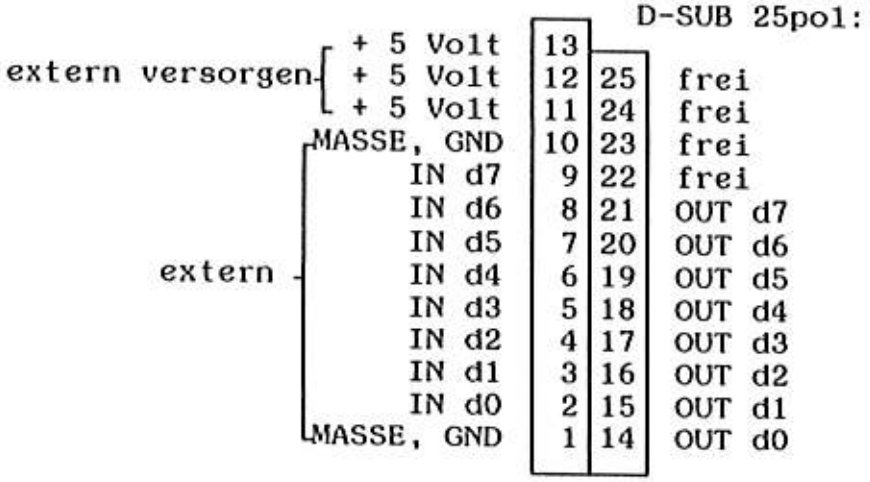
Das Einlesen erfolgt mit den Befehlen LDP und LDB.

```
Beispiel 2:      Einlesen des 8 Bit Ports
     ldp 1      ;Zustand bei Port 1 (Adresse) einlesen


Beispiel 3:      Einlsen von Bit 2 des Ports
     ldb 1,1    ;einlesen NUR von Bit 1 des Port 1 (1-64)
```

Bitbezeichnugen: DATEN d7 d6 d5 d4 d3 d2 d1 d0
BIT 8 7 6 5 4 3 2 1

## 2.2.4.1 Schaltbild Ui1/8 Bit Ein-/Ausgang 2.2.4.1

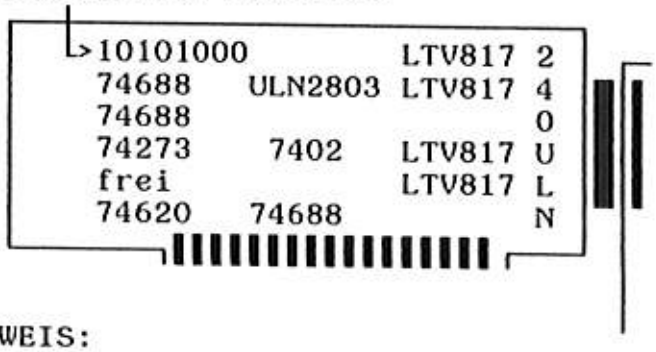
Die Anschlußkonfiguration:




LÖTBRÖCKEN:
Die Lötbrücken b-i unter dem IC1 (74688) müssen für PC-Betrieb (IBM) geöffnet sein! Dies ist bei Auslieferung sichergestellt.
Lage


KARTENADRESSE:

Dipschalter Position




HINWEIS: Die Kartenadresse ist bei der Auslieferung auf hex 0315 eingestellt!

HINWEIS:
Gesetzt werden nur A0 bis A7 (03xx ist fest verdrahtet!)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adresse: | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| Schalter: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Zustand: | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | off | on | off | on | off | on | on | on |

ACHTUNG:

Für 8 Bit Parallel-Ausgabe muß die Port-Adresse 8315 sein!!!

2.2.5 Ui1/32A, Interface Ui1/32 Bit Ausgang
Ui1/32E, Interface Ui1/32 Bit Eingang 2.2.5

## Technische Daten

Stromversorgung:
Logic: UB = 5 Volt (wird vom PC gespeist)
Imax = 140 mA (C-Mos Logic)
Ausgangstreiber: UB = 5 oder 12 Volt (wird vom PC gespeist)
(je nach Wahl der Lötbrücken o oder p)

Ausgänge:
Pegel = 5 Volt (Brücke "o" geschlossen, "p" geöffnet!)
Pegel = 12 Volt (Brücke "p" geschlossen, "o" geöffnet!)
Imax = 100 mAmp, maximale Belastung bei 5 Volt-Pegel.

Die Versorgung der einzelnen Treiberstufen erfolgt über einen 10 Ohm Schutzwiderstand. Dieser müßte gegebenenfalls anderen Lastbedingungen angepaßt werden.

Schutzmaßnahmen:
Um die Quellschaltung vor schädlichen Rückwirkungen (Spannungsspitzen, Oberspannung am Ausgang) zu schützen, sind als Ausgangstreiber die Schaltungen UDN 2981 eingesetzt.

ACHTUNG:

Die Ausgänge sind nicht optoentkoppelt!

Programmierung:

Die Ausgabe von Daten erfolgt mit den Befehlen STP und STB.

```
Beispiel 1:     Ausgabe von logisch 1 (H) bei Bit 24
     ld 128          ;lade Akku mit Bit 8 (dez=128, hex=8) (Tabelle)
     stp 1           ;Akku nach Port 1 (1-64) Adresse 1, gerade
     ld 4            ;lade Wert für Fenster 3 (-> Tabelle)
     stp 2           ;schreibe den zwischengespeicherten Wert nach
                     ;Fenster 3 = Bit 24. Adresse 2 ungerade
```

Das Einlesen von Daten erfolgt mit dem Befehl LDP und LDB

```
Beispiel 2:     Einlesen (8 Bit) von Fenster 2, TTL-Pegel
     ld 02           ;Fenster 2 wählen
     stp 1           ;Fenster setzen
     ldp 1           ;lese den Inhalt von Port 2, Fenster 2 i.d.Akku
```

Bit-Tabelle (dezimal):

| Bit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Wert | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |

Fenster-Tabelle (dezimal):

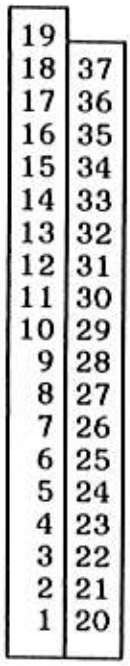
Bit 1-8 = Fenster 1 hex 01
Bit 9-16 = Fenster 2 hex 02
Bit 17-24 = Fenster 3 hex 04
Bit 25-32 = Fenster 4 hex 08

2.2.5.1 Schaltbild, Ui1/32 Bit Ausgang
Ui1/32 Bit Eingang

2.2.5.1

Die Anschlußkonfiguration:

| Ausgang | Bit | Fenster |
|---|---|---|
| 19 | 3 | 3 |
| 18 | 2 | 3 |
| 17 | 1 | 3 |
| 16 | 8 | 2 |
| 15 | 7 | 2 |
| 14 | 6 | 2 |
| 13 | 5 | 2 |
| 12 | 4 | 2 |
| 11 | 3 | 2 |
| 10 | 2 | 2 |
| 9 | 1 | 2 |
| 8 | 8 | 1 |
| 7 | 7 | 1 |
| 6 | 6 | 1 |
| 5 | 5 | 1 |
| 4 | 4 | 1 |
| 3 | 3 | 1 |
| 2 | 2 | 1 |
| 1 | 1 | 1 |



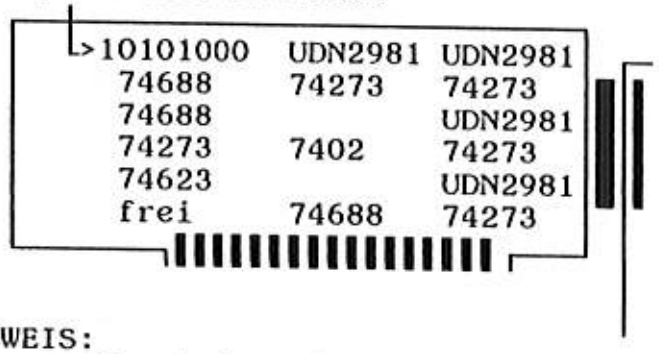

D-SUB 37 polig

Masse
Masse
Masse
Masse

| ;Ausgang | Bit | Fenster |
|---|---|---|
| 32 | 8 | 4 |
| 31 | 7 | 4 |
| 30 | 6 | 4 |
| 29 | 5 | 4 |
| 28 | 4 | 4 |
| 27 | 3 | 4 |
| 26 | 2 | 4 |
| 25 | 1 | 4 |
| 24 | 8 | 3 |
| 23 | 7 | 3 |
| 22 | 6 | 3 |
| 21 | 5 | 3 |
| 20 | 4 | 3 |

KARTENADRESSE:

Dipschalter Position




HINWEIS: Die Kartenadresse ist bei der Auslieferung auf hex 0315 eingestellt!

HINWEIS:
Gesetzt werden nur A0 bis A7 (03xx ist fest verdrahtet!)

| Adresse: | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
|---|---|---|---|---|---|---|---|---|
| Schalter: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Zustand: | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | off | on | off | on | off | on | on | on |

ACHTUNG:

Für 8 Bit Parallel-Ausgabe muß die Port-Adresse 83xx sein!!!
Ui1/32A belegt 2 Adressen!
Die 1. Adr.(gerade) übernimmt die Daten, die 2. das Fenster.

Für Ihre Notizen

# 3.0 ROBOT-SPS, Einleitung 3.0

Die ROBOT-SPS dient zur Steuerung, Überwachung und Beeinflussung von maschinellen und prozeßtechnischen Abläufen.

Die Bezeichnung ROBOT-SPS haben wir gewählt, weil diese SPS in der Lage ist, auf die spezifischen Erfordernisse der Programmierung von Industrierobotern einzugehen.

Die Forderungen, die wir an eine solche Steuerung stellen, insbesondere dann, wenn sie auch in der Ausbildung eingestzt werden soll, sind:

Transparenz : übersichtliche Programmstruktur, einfache, logische Programmiersprache mit Unterprogrammstrukturen, ausführliche Dokumentation im Arbeitsprogramm. (dies ist besonders wichtig für spätere Änderungen!)

Befehlssatz : der Befehlssatz soll auf die wesentlichen Erfordernisse beschränkt sein, ohne die Kreativität des Anwenders in irgendeiner Weise zu begrenzen.
Die Befehle müssen sich logisch aus ihrer Funktion ableiten lassen.
Die Befehle sollen kurz sein.
Bei logischen und arithmetischen Befehlen soll die Anlehnung an den Assembler für Intel-Prozessoren erkennbar sein.

Darstellung: alle Abläufe, Veränderungen, Fehlermeldungen, Meldetexte und sonstige Situationen während eines Programmlaufes müssen übersichtlich, ohne Ausblendung von Teilen des Bildschirms, dargestellt werden können.
Die Vorgänge und Zustände, insbesondere im AKKU und bei den PORTS sollen in dezimaler, hexadezimaler und binärer Form dargestellt sein.
Die SOLL/IST Positionen müssen zu jeder Zeit vom Schirm ablesbar sein.

Das Programm muß in der Lage sein, in einem speziellen Hilfsfenster (reserviert nur für solche Manipulationen) Texte und einfache grafische Figuren darstellen zu können.

Interfaces: die Hardware muß funktionell gut gegliedert sein.
Die einzelnen Karten müssen sich auf eine Funktionsart beschränken und dafür einfach, übersichtlich und unanfällig für Störungen sein.
Alle Signale der Interfacekarten sollten optisch entkoppelt sein, um den Systembus des PC zu schützen.

## 3.1 Gundfunktionen 3.1

Die ROBOT-SPS gestattet folgende Grundfunktionen:

1. Laden und sichern von Arbeitsprogrammen.

2. Anzeige von Diskettenverzeichnissen.
   Im Editor können Sie (z.B.) ein Verzeichnis der bereits vorhandenen Arbeitsprogramme aufrufen.

3. Anzeige einer Label-Liste.
   Im Editor können Sie eine Liste aller bisher erstellten Labels aufrufen.

4. Ausdruck von Arbeitsprogrammen.
   Das Ausdrucken von Arbeitsprogrammen, insbesondere bei langen Kommemtarzeilen, ist sehr hilfreich.

5. Einstellung von Parametern.
   Alle Grundwerte werden vorab im Prameterfeld eingestellt.
   Das Programm bezieht sich bei allen Abläufen auf diese Werte.

6. Editor mit kpl. Blockfunktionen zur Programmerstellung.
   Umfangreiches Instrument zur Bearbeitung von Arbeitsprogrammen.

7. Dukumentierung jeder einzelnen Programmzeile.
   Sie haben die Möglichkeit, jede Programmzeile mit einem ausführlichem Kommentar zu versehen.

8. Zeichenausgaben während des Programmlaufes.
   Über den Drucker UND im Hilfsfenster möglich.

9. Abfrage der Tastatur während des Programmlaufes.

10. Unterprogrammaufruf über Labels.
    Obersichtliche Programmstrukturierung durch Labels mit bis zu 30 Zeichen!

11. Direkte Manipulation von 8 TEACH-ROBOT.

12. Direkte Manipulation von 2 PROBOT (Kleinroboter mit Schrittmot.)

13. Setzen/lesen und logisch/arithmetische Verknüpfung von Ein-/Ausgängen

14. Programmierung von Pausen. (0,1 - 999,9 Sekunden)

15. Verknüpfung mit Hochsprachen wie Basic, Pascal und C durch Softwareschnittstelle.

16. Fehlermeldungen.

17. Hilfsmenü für alle Situationen!

```
R O B O T  -  S P S          I N F O    (C) 1991 Microelectronic Kalms GmbH

steuert bis zu 8 Roboter:    6(7)-achsige TEACH-ROBOT mit Servo-Motoren
                             oder 6(7)-achsige PROBOT mit Schritt-Motoren
und bis zu 64 I/O-Karten:    8 Leistungsausgänge, open collector, optoentk.
                             4 Relais-Leistungsschalter, 1 x UM, 220V/2,5A
  V 2.03  ab DOS 2.01        8 (32) Bit Eing-/Ausgang, TTL, optoentkoppelt
  PROBOT Adr. 8378/8278      8 Bit Analog Ein-/Ausgang, 0-10 V, optoentk.
  IBM PC/Komp.512 K RAM      Als Kartenadressen bitte 0300-031F verwenden!
» » » » » » SOFTWARE-SCHNITTSTELLE für BASIC, PASCAL und C. « « « « « «

Das Help-Menü (Aufruf mit -F1-) informiert ausführlich über die lieferbare
HARDWARE: |Uil/Robot|Uil/Open Kollektor|Uil/Relais|Uil/8/32 Bit Ein-/Ausg.|
          |Uil/Analog|Prüfadapter|Erweiterungs-Bus|Sonder-Karten|

WARNUNG: Im nachfolgenden PARAMETER-FELD müssen alle Adressfelder mit einer
         gültigen Adresse oder mit 0000 gekennzeichnet sein.
         Doppeladressen führen zu Störungen! Prüfen Sie alle Adressierungen
         vor Inbetriebnahme! Wir übernehmen keinerlei Haftung bei falscher
         Handhabung, unsachgemäßer Inbetriebnahme, und bei Fremdeingriffen.

Mit beliebiger Taste weiter zum          | Lizenznehmer/Lizenznummer:
PARAMETER-FELD für die Grundeinstellungen|       (Demoversion)
```

```
R O B O T  -  S P S          P A R A M E T E R - F E L D    (C) Kalms GmbH

Robot/Karte Nr.:   1       2       3       4       5       6       7       8
Kartenadresse    0315    0000    0000    0000    0000    0000    0000    0000
Motor/Achse   7 ... 1 7 ... 1 7 ... 1 7 ... 1 7 ... 1 7 ... 1 7 ... 1 7 ... 1
Motor aktiv   0111111 0111111 0111111 0111111 0111111 0111111 0111111 0111111
Endschalter   0111111 0111111 0111111 0111111 0111111 0111111 0111111 0111111
Referenz      0111111 0111111 0111111 0111111 0111111 0111111 0111111 0111111
Time out   :  (0800)    0800 | Nur installierte Motoren und | K-Adr 0300-0315
Rampe/Takt:   (0400)    0100 | aktive Funktionen mit 1 kennz. | Probot-Adr-8378

Port Nr.:    1       2       3       4       5       6       7       8
Adresse   0000    0000    0000    0000    0000    0000    0000    0000
aktivllllllll llllllll llllllll llllllll llllllll llllllll llllllll llllllll
Bit  7......0 7......0 7......0 7......0 7......0 7......0 7......0 7......0
Port Nr.:    9      10      11      12      13      14      15      16
Adresse   0000    0000    0000    0000    0000    0000    0000    0000
aktivllllllll llllllll llllllll llllllll llllllll llllllll llllllll llllllll
1 = aktiv | Das Programm verwendet NUR die hier abgelegten Einstellungen !

ACHTUNG: Bereits vorhandene Parameter werden  Ü B E R S C H R I E B E N !
Wollen Sie alle Parameter so nach "ROBOT.INF" übernehmen ? Sind Sie sicher ?
Wenn ja, dann drücken Sie  ■ CR (Return) ■, wenn nicht  ■ ESC ■ !
```

```
R O B O T  -  S P S              H A U P T M E N Ü              (C) 1992 Kalms GmbH

ROBOT 1      M1      M2      M3      M4      M5      M6      M7  | PORT 01    HEX DEZ
Pos AKT       0       0       0       0       0       0       0  |in 00000000 00 000
Pos NOM       0       0       0       0       0       0       0 out 00000000 00 000
Zyklus
                PROGRAMM                ABLAUF (TRACE)          | AKKU  00000 $0000 %0000000000000000
ZEILE                                                           |                  15      7      0
0022  ;*** Init Progr                                           |                              START-MENÜ
0023        ld o  ;We                                           | -> START TRACE
0024        st a                                                | G      Start Programm bei Cursor
0025        st b                                                | S      Einzelschritt (single step)
0026        st c ;u.s                                           | HOME   Start vom Anfang (Zeile 0)
0027  trace 1     ;al                                           | Esc    Abbruch, zum Hauptmenü
0028        ld 0                                                |
0029        st h                                                | -Während des Programmlaufes-
0030                                LD    0                     |
0031  ;*** Hauptprogr               ST    A                     | Eine mögliche Fehlermeldung er-
0032                                ST    B                     | scheint in der letzten Zeile des
0033  ANFANG:                       ST    C                     | Hilfsfeldes.
0034        trace 1                 TRACE 1                     | NOTAUS mit Taste BREAK/Scroll/Roll
0035  robot 1                     LD    0                       |
0036        home                                                | F1     Anzeige aktueller Roboter
                                                                |                          F1=Hilfe
```

```
R O B O T  -  S P S              H A U P T M E N Ü              (C) 1992 Kalms GmbH

ROBOT 1      M1      M2      M3      M4      M5      M6      M7  | PORT 01    HEX DEZ
Pos AKT       0       0       0       0       0       0       0  |in 00000000 00 000
Pos NOM       0       0       0       0       0       0       0 out 00000000 00 000
Zyklus
                PROGRAMM                                        | AKKU  00000 $0000 %0000000000000000
ZEILE                                                           |                  15      7      0
0022  ;*** Init Programm ***                                    |                              EDITOR
0023        ld o  ;Wert in Variable lad                         | -> Dir-D- Laden-L- Sichern-S-
0024        st a  ;Kommentar -> -> -> -                         | -> Block-Manipulation
0025        st b                                                | -> Syntax, Labels Fehlermeldungen
0026        st c ;u.s.w.                                        | Kommandos:
0027  trace 1     ;alle Werte und Verän                         | Ctrl L...Programm laden
0028        ld 0                                                | Ctrl S...Programm sichern
0029        st h          ;Home-Zähler, b                       | Ctrl N...Zeile einfügen
0030                                                            | Ctrl Y...Zeile löschen
0031  ;*** Hauptprogramm ***                                    | Alt  Y...alter Zeileninhalt zurück
0032                                                            | Ins.....Leer-Zeichen einfügen
0033  ANFANG:                                                   | Del.....Zeichen löschen
0034        trace 1                                             | Ctrl G...goto: Label od.Zeilen-Nr.
0035  robot 1                                                   | Ctrl F...suchen, find
0036        home                                                | Ctrl R...ersetzen, replace
                                                                |                          F1=Hilfe
```

## 3.1.2 Aufruf unter BASIC und PASCAL 3.1.2

Grundsätzlich lassen sich die ROBOT-SPS und die dazugehörigen Arbeitsprogramme auch unter Basic und Pascal aufrufen.

Dazu ist es erforderlich, die ROBOT-SPS resident zu laden.

Eingabe:
A>ROBOT RESIDENT 3F (Nummer des zul. Sotware-Interrupts)

Das Arbeitsprogramm kann jetzt durch Basic oder Pascal erstellt werden oder ein Bestehendes geladen werden.

Sie können jetzt die ROBOT-SPS wie ein Unterprogramm aufrufen.
(Ein entsprechendes Maschinenprogramm wird mitgeliefert.)

ACHTUNG:

> Am Ende eines Arbeitsprogramms muß jetzt der Befehl RET stehen. Dies ist der Rücksprungbefehl zur Hochsprache!

Möglichkeiten der Ablage von Parametern für das aufgerufene Arbeitsprogramm:

| | |
|---|---|
| Variable | A bis Z, alle Buchstaben des Alphabets, Groß- oder Kleinschreibung. |
| Indirekt | (A) >0-65535<, der Inhalt einer Variablen darf hierbei zwischen 0 und 65535 liegen. Die Variable ist in Klammern zu setzen. |

ANMERKUNG:
Bei dieser Adressierungsart können, zusätzlich zu den normalen Variablen, weitere Speicherplätze angesprochen werden.
Die ROBOT-SPS organisiert diese Plätze intern, d.h. der Inhalt der Variablen ist NICHT die Adresse eines Speicherplatzes im PC, sondern nur die Nummer der Zeile des internen Speichers, den das Programm für diese Operationen reserviert hat.

Beispiel: (Arbeitsprogramm in ROBOT-SPS)

```
ld 65500        ;lade den dezimalen Wert 65500 in den Akku.
st a            ;lege den Inhalt des Akkus in der Variablen
                ;a ab.
ld $FF          ;lade den Akku mit dem (hex) Wert FF.
st (a)          ;speichere den Inhalt des Akkus an die Stelle
                ;des internen Speichers den die Variable
                ;bezeichnet. (Hier: Zeile 65500 des STSP)
```

HINWEIS:
Ihr PC sollte für diese Operationsmöglichkeit einen Arbeitsspeicher von mindestens 640 KByte haben. Ist der Arbeitsspeicher kleiner, so ist die Obergrenze der indirekten Adressierung von 65535 auf einen entsprechend niedrigeren Wert zu korrigieren!

## 3.1.4 Aufruf unter C

3.1.4

Grundsätzlich lassen sich die ROBOT-SPS und die dazugehörigen Arbeitsprogramme auch unter C aufrufen.

Dazu ist es erforderlich, die ROBOT-SPS resident zu laden.

Eingabe:
A>ROBOT RESIDENT 3F (Nummer des zul. Sotware-Interrupts)

Das Arbeitsprogramm kann jetzt durch C erstellt werden oder ein Bestehendes geladen werden.

Sie können jetzt die ROBOT-SPS wie ein Unterprogramm aufrufen. (Ein entsprechendes Maschinenprogramm wird mitgeliefert.)

ACHTUNG:

Am Ende eines Arbeitsprogramms muß jetzt der Befehl RET stehen. Dies ist der Rücksprungbefehl zu C !

Möglichkeiten der Ablage von Parametern für das aufgerufene Arbeitsprogramm:

| | |
|---|---|
| Variable | A bis Z, alle Buchstaben des Alphabets, Groß- oder Kleinschreibung. |
| Indirekt | (A) >0-65535<, der Inhalt einer Variablen darf hierbei zwischen 0 und 65535 liegen. Die Variable ist in Klammern zu setzen. |

ANMERKUNG:
Bei dieser Adressierungsart können, zusätzlich zu den normalen Variablen, weitere Speicherplätze angesprochen werden. Die ROBOT-SPS organisiert diese Plätze intern, d.h. der Inhalt der Variablen ist NICHT die Adresse eines Speicherplatzes im PC, sondern nur die Nummer der Zeile des internen Speichers, den das Programm für diese Operationen reserviert hat.

Beispiel: (Arbeitsprogramm in ROBOT-SPS)

```
ld 65500        ;lade den dezimalen Wert 65500 in den Akku.
st a            ;lege den Inhalt des Akkus in der Variablen
                ;a ab.
ld $FF          ;lade den Akku mit dem (hex) Wert FF.
st (a)          ;speichere den Inhalt des Akkus an die Stelle
                ;des internen Speichers den die Variable
                ;bezeichnet. (Hier: Zeile 65500 des STSP)
```

HINWEIS:
Ihr PC sollte für diese Operationsmöglichkeit einen Arbeitsspeicher von mindestens 640 KByte haben. Ist der Arbeitsspeicher kleiner, so ist die Obergrenze der indirekten Adressierung von 65535 auf einen entsprechend niedrigeren Wert zu korrigieren!

3.2 Hilfsmenü 3.2

Hardware:
-> Garantie, Hardware-Konfiguration
Querverweise: -> Teach-Robot, Probot, Übersicht Ui1-IBM
Ui1/Robot, Ui1/Open Kollektor, Ui1/Relais
Ui1/8 Bit E&A, Ui1/32 Bit E/A, Ui1/Analog
Prüfadapter, Stromversorgung, Kartenadresse, Init-?-
Parameter-Feld (Basis-Daten)
-> Parameter-Feld (Basis-Daten)
Querverweise: -> RAMP, Time Out

Bedienung:
-> Hauptmenü, -> Init-?-
Querverweise: -> Referenz, HOME, Parameter-Feld (Basis-Daten)
-> MS-DOS, -> Editor-Esc-
Querverweise: -> Dir-D-, Laden-L-, Sichern-S-, Block-Manipulation
-> Syntax, Labels, Fehlermeldungen
-> Startmenü-G-
Querverweise: -> START, TRACE
-> Notaus
Querverweise: -> Init-?-
-> Dir-D-, -> Laden-L-, -> Sichern-S-
Querverweise: -> Dir-D-, Blockmanipulation

Programmierung:
-> Punktsteuerung
Querverweise: -> Teach-in, Motordaten
-> Bahnsteuerung, -> Teach-in
Querverweise: -> Action-Box
-> Binär Ein-/Ausgang

Programmiersprache:
-> SPS, -> Befehle
Querverweise: -> ROBOT, RAMP, GRIP, TRACE, NOP, STOP
-> M1+, M1-, M1=, START, LEFT, RIGHT
-> HOME, HALT, PROTOCOL, PAUSE
-> JUMP, CALL, RET, PUSH, POP, LD, ST
-> LDB, STB, LDP, STP, XCHG, XCHGC, ADD
-> SUB, ADC, SBC, CMP, OR, XOR, AND, CLC
NOT, NEG, ABS, MUL(S), DIV(S), ROL, ROR
J(N)E, J(N)C, JPL, JMI, J(N)Z, SHL, SHR
-> AKT/NOM, Color, Print, Input, CLS
-> Syntax, Labels
Querverweise: -> Adressierungen
-> Basic, -> Pascal, -> C
Querverweise: -> Softw-Schnittstelle, Pasro
-> Software-Schnittst., -> Beispiele, ->Fehlermeldungen

HINWEIS:
Der letzte Inhalt des Help-Fensters bleibt nach der Rückkehr (ESC) zum Editor erhalten.

# 3.3 Initialisierung, Aufruf unter DOS 3.3

Eine gültige Initialisierung ist die Voraussetzung für die störungsfreie Funktion aller beteiligten Baugruppen.

Erforderlich:
1. Vor dem ersten Start eines Ablaufprogramms.
   Bevor ein Arbeitsprogramm gestartet wird, muß der Roboter in einer definierten Position stehen. (Referenz)
   Damit dies gewährleistet ist, setzt man als ersten Bewegungs-Aufruf den Befehl -HOME-.

2. Nach einem erfolgten -NOTAUS-.
   NOTAUS ist eine willkürliche Unterbrechung des Programms.
   Nach einem erfolgten Notaus 'weiß' das Programm nicht mehr, wo der Roboter zuletzt war.
   Mit -?- gehen Sie zunächst zurück ins Parameterfeld und von da aus mit -Esc- wieder in das Hauptmenü.
   Alle Roboter haben nun wieder ihre gültige Referenz.

3. Eventuell beim Zugriff von einer Hochsprache aus.

AUFRUF UNTER DOS

A>ROBOTSPS.EXE

Minimalkonfiguration:
512 K RAM
1 Laufwerk
Herkules (VGA) Grafik

AUTOSTART:
Wenn Sie die ROBOT-SPS mit einem bestimmten Arbeitsprogramm automatisch starten wollen, dann geben Sie ein,

A>ROBOTSPS.EXE AUTO xxxx (xxxx = Name des Arbeitsprogramms)

Oder Sie machen sich eine Batch-Datei nach folgendem Muster.

| Eingabe: | Kommentar: |
|---|---|
| copy con:A.bat | Ihre batch-Datei ist aufrufbar mit A |
| cls | clear screen |
| robotsps.exe auto xxxx | (xxxx = Name des Arbeitsprogramms) |
| Ctrl Z | alle Eingaben mit RETURN abschließen |

Jetzt können Sie mit A die Robot-SPS aufrufen, wobei das gewünschte Arbeitsprogramm automatisch geladen und gestartet wird.

Im Zusammenwirken mit MS-DOS können folgende Fehlermeldungen kommen:

| | |
|---|---|
| Datei zu kurz | Speicherbereich zu nicht ausr. |
| Laufwerk voll | Ungültige Speicherblockadresse |
| Ungültige Funktion | Falsche Umgebungsparameter |
| Datei nicht gefunden | Falsches Format |
| Verzeichnis nicht gefunden | Zugriffsart nicht möglich |
| Zu viele Dateien offen | Ungültige Dateien |
| Zugriff nicht möglich | Falsches Laufwerk |
| Falsche Dateinummer | Akt. Verzeichnis nicht löschbar |
| Speicherbereich nicht zus.hängend | Kein weiterer Eintrag |

## 3.3.1 Parameterfeld, Kartenadresse 3.3.1

Das Programm richtet ausschließlich nach diesen Einstellungen!

ROBOTER-BEREICH

| | | |
|---|---|---|
| Robot/Karte Nr. | 1 | (1-8) Benennung der einzelnen Roboter. |
| Kartenadresse | 0300 | 0300-031F, Adresse, unter welcher die Interface-Karten angesprochen werden. |

ACHTUNG:

> Wollen Sie mehr als 32 Karten ansprechen, so müssen Sie, die, in Ihrem System noch freien Adressen, individuell ermitteln!

| | | |
|---|---|---|
| Motor/Achse | 7 ... 1 | Benennnug der Motoren. |
| Motor aktiv | 01111111 | 0111111 \| 1 = Mot angemeldet<br>└Finger \| 0 = kein Mot,inaktiv<br>└Hand-Dreh<br>└Hand-Schwenk<br>└Arm<br>Schulter<br>Körper<br>frei (Option) |

ACHTUNG:

> Ist ein Motor angemeldet, aber nicht vorhanden, so kann das Programm -HÄNGEN- bleiben !

| | | |
|---|---|---|
| Endschalter | 01111111 | 1 gesetzt = Antrieb besitzt Endsch.<br>0 gesetzt = Antrieb hat keine Endsch.<br>(gibt an, ob es einen mechanischen oder elektronischen Stopper gibt, der als Referenz verwendet werden kann.) |

HINWEIS:
Ist die 0 gesetzt, so wird die Referenzposition von der Software angesteuert; es handelt sich jedoch nicht um eine echte Referenz, da diese errechnete Position sich im Laufe der Zeit verstellt!

| | | |
|---|---|---|
| Referenz | 01111111 | 1 gesetzt = dieser Motor wird beim HOME Befehl i.d. Referenzposition bewegt. 0 gesetzt = der HOME Befehl ignoriert diesen Motor. |

PORT-BEREICH

| | | |
|---|---|---|
| Port Nr. | 1 | (1-32) Benennung der verschiedenen Eing-/Ausgangs-Interfaces. |
| Adresse | 8300 | (8300-831F) Die 8 steht für Byte-weise Ausgabe direkt, maskiert od. unmask. |
| aktiv | 11111111 | 1 gesetzt = das entsprechende Bit wird berücksichtigt.<br>0 gesetzt = das Bit wird ignoriert |
| Bit | 7.....0 | |

Mit F1 verlassen Sie das Parameterfeld und sichern alle Daten in der internen Datei Robot.inf.
Mit Esc verlassen Sie das Parameterfeld ebenfalls, die Daten bleiben aber UNGESICHERT.

## 3.3.2 Time out

Time-out ist mit seinem Wert ein interner Multiplikator für die Zeit, die das Programm wartet, bevor es die Entscheidung trifft: -es erfolgt kein Flankenwechsel mehr-, der Antrieb ist in den Endschalter gefahren oder blockiert, er wird abgeschaltet.

Standardwert: 0800
für 8 MHz/286 µP. Für höhere Verarbeitungsgeschwindigkeiten kann dieser Wert nach unten korrigiert werden.



## 3.3.3 Rampe

Die Rampe beeinflußt das Anlauf- und Bremsverhalten eines Antriebes.

Sie soll abrupte Massenbewegungen und damit ein zu großes Überschwingen des Roboterarmes verhindern.

Für höhere Clock-Frequenzen kann dieser Wert nach oben korrigiert werden.
Dieser Wert wird ebenfalls nach oben korrigiert, wenn die Betriebsspannung für den Roboter erhöht wird. (Für TEACH-ROBOT max. 19 Volt)

Syntax:

```
RAMP 0
|    └──────Rampe aus
└───────────Befehlsname
```

oder

```
RAMP x
|    └Rampe einstellen, prozentualer Wert zwischen
|     1 und 1000.
└─────Befehlsname
```

ERLÄUTERUNG:
RAMP 100 = Die im Parameterfeld eingestellte Rampe ist zu 100 % wirksam.
RAMP 55 = ... ist zu 55% wirksam, d.h. die Steilheit der Rampe ist entsprechend größer. (steiler)
RAMP 300 = ... ist zu 300% wirksam, d.h. die Steilheit der Rampe ist entsprechend kleiner. (flacher)
Werte sind zwischen 0 und 1000 möglich.
Standardwert 0100 für 8MHz/286 µP.

## 3.3.4 Notaus

3.3.4

Eine wichtige Forderung beim Betrieb von Industrierobotern ist die Sicherheit.
Ein Teil dieser Sicherheit ist der Schalter NOTAUS.

Diese Funktion wird beim Schulungsroboter TEACH-ROBOT und dem Kleinroboter PROBOT durch drücken der Taste -Break- (Scroll/ROLL) realisiert.

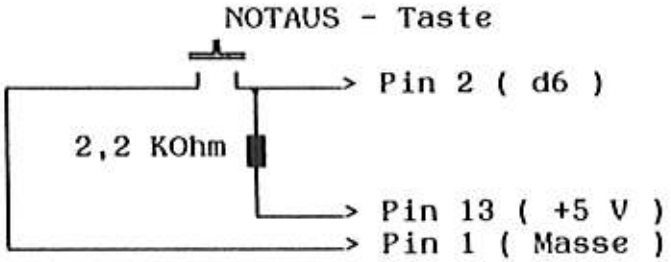
Taste: Break (Scroll/ROLL) ->NOTAUS
Alle Antriebe werden spontan gestoppt.
Der Programmablauf wird gestoppt.

ACHTUNG:

Bei Notaus wird das Programm willkürlich unterbrochen !!!
Es können dabei Daten und Positionen verlorengehen.
Vor der Fortsetzung des Programms muß neu INITIALISIERT werden!

Sie können natürlich auch einen der beiden freien Eingänge des Interfaces Uil/Robot verwenden.

Beschaltung: Buchse 25pol, TEACH-ROBOT




Im Parameterfeld muß die Adresse der Interface-Karte Uil/Robot, bei der Sie die Taste NOTAUS beschaltet haben, zusätzlich bei Port 1 (1-16) eingetragen werden.

Unterroutine Notaus:

```
NOTAUS:              ;Label, Name
     LDB 6,1         ;lade Bit 6 von Port 1
     cmp %01000000   ;Vergleiche d6 mit dem Inhalt des Akkus
     jne Motaus      ;springe zur Routine Motaus (alle Motoren aus)
                     ;wenn d0 zu 0 geworden ist.
     ret             ;Rückkehrbefehl zur Ausgangsroutine
                     ;(return from subroutine)
MAUS:                ;Unterroutine: alle Motoren abschalten
u.s.w.               ;stop1 stop2 ... ret
```

noch nicht belegt

Nachdem alle Werte in das Parameterfeld eingetragen sind und mit F1 abgespeichert wurden, meldet sich die ROBOT-SPS mit der Maske des Hauptmenüs.
Von dieser Maske aus lassen sich alle Funktionen des Programms aufrufen.
Im Helpfeld sind die verschiedenen Möglichkeiten aufgezeigt.

ÜBERSICHT:

Funktionen: (Kennz.1010592) Vers. 1 vom 1.5.92

| | |
|---|---|
| ? = zurück zum Parameterfeld | z.B. nach Notaus oder zur Änderung von Parametern. (z.B. Rampe, Time out) |
| L = Arbeitsprogramm laden | Namen des Arbeitsprogramms in die invertierte Zeile schreiben und mit CR holen. |
| S = Arbeitsprogramm sichern | Namen des Arbeitsprogramms in die invertierte Zeile schreiben und mit CR sich. |
| D = DIR | Verzeichnis der Arbeitsprogramme *.ROB im Helpfenster darstellen. Reicht das Fenster nicht aus, so können Sie mit der Cursortaste das Listing rollen. |
| G = Start-Menü aufrufen<br>Hinweis: Die inv. Programmzeile kann mit den Cursortasten bewegt wegt werden.<br>Mit ESC mach Sie diese Eingabe rückgängig. | G (2.Eingabe) startet ein Arbeitsprogramm bei der invertierten Programmzeile.<br>S startet einen Programmschritt bei der inv. Zeile.<br>HOME (7 im Ziffernblock, Numlock off) startet das Arbeitsprogramm von Zeile 0000. |

HINWEIS:
Ein laufendes Arbeitsprogramm können Sie mit der Taste BREAK unterbrechen (->NOTAUS).
Mit G können Sie das Programm fortsetzen oder mit ESC in das Hauptmenü zurückkehren. (->INITIALISIERUNG)

| | |
|---|---|
| Esc = Editor für Arbeitsprogramms | Mit den Cursortasten können Sie die zu bearbeitende Zeile aufsuchen. |

X = zurück zu DOS; ROBOT-SPS Programm beenden.

Hilfs-Menü:

| | |
|---|---|
| F1 | Hilfs-Menü aufrufen; Im Helpfenster erscheint eine Übersicht über die Teilgebiete der Hilfe. |
| <- \| -> | Begriff auswählen; Mit den Cursortasten springen Sie zum gewünschten Begriff. Dieser blinkt dann. |
| RETURN (CR) | Begriff aufrufen; Im Helpfenster erscheint nun der ausgewählte Begriff. Weitere Querverweise können Sie, soweit angeboten, mit den Cursortasten <--/--> (li/re) auswählen. Mit der Taste <-- (⊢—, Backspace) kehren Sie in die jeweils vorausgegangene Hilfsebene zurück. Mit Esc verlassen Sie das Hilfsmenü, dabei bleibt jedoch der zuletzt ausgewählte Teilbereich (Begriff) im Helpfenster erhalten. |

Erklärung der Begriffe im H A U P T M E N Ü

| | |
|---|---|
| ROBOT 1 (1-8) | aktueller Roboter; Hier wird angezeigt, während eines Arbeitsprogrammlaufes möglicherweise wechselnd, welcher Roboter gerade angesprochen wird. |
| Pos AKT | Ist-Position: Die Position, auf die der Roboter tatsächlich gelaufen ist. (Soll + Schlupf) |
| Pos NOM | Soll-Position: Die Position, auf die sich der Roboter hin bewegen soll. |
| PORT 01 (1-64) | aktuelles Port; Nummer des gerade angesprochenen Ports. |
| in | Wort am Eingang; Bitkonfiguration des ausgewählten Ports. |
| out | Wort für Ausgang; Bitkonfiguration für das ausgewählte Port. |
| Zyklus | Zahl d.Durchläufe; Wird mit dem Befehl HOME um 1 erhöht. |
| ZEILE | Programmzeilen-Nummer. Aktive Zeile ist invertiert. |
| PROGRAMM | Programm-Liste; (Kommentar ist teilweise verdeckt) |
| ABLAUF (TRACE) | tatsächlicher Programmverlauf.(->bedingte Sprünge) |
| AKKU | Inhalt des Akkus. |

## 3.5 Editor, Grundfunktionen 3.5

Mit dem Editor können Sie alle Arbeitsprogramme erstellen und bearbeiten. --> Aufruf mit -Esc-.

Mit den Cursor Tasten können sie sich in einem Bereich von
--> 10 000 Zeilen x 253 Zeichen <-- frei bewegen !

Zur Übersichtlichen Programmdarstellung/-eingabe benutzen Sie bitte den TABulator !
Mit -Ctrl T- können sie eine Tabulatormarke setzen, oder, wenn bereits vohanden, löschen.

HINWEIS:
Sie können jede Programmzeile mit einem Kommentar versehen.
Dazu stehen Ihnen insgesammt 253 Zeichen (abzüglich des Programmteiles) in jeder Zeile zur Verfügung.
Textzeilen, die länger sind als das Editorfenster breit ist, werden (nach links) gerollt. Optional ist ein Spaltenzähler in der TAB-Zeile vorgesehen.

Durch das Zeichen ; (Semikolon) wird der Kommentar vom Programmteil abgetrennt.

Kommandos:

Ctrl L...Programm laden, Namen des Programms in die invertierte Zeile (max. 30 Zeichen) eingeben und mit -CR- holen.

HINWEIS:
Geladen wird der gesammte Programmbereich, 0-9999.
Ein bereits geladenes Programm wird überschrieben.
Zum Laden von Teilbereichen benutzen Sie bitte die Blockfefehle.
(z.B. Alt L. ->EDITOR, ->BLOCK-MANIPULATION).

Ctrl S...Programm sichern, Namen eingeben und mit -CR- sichern.

Ctrl N...Zeile einfügen. Alle Sprungadressen werden automatisch korrigiert!

Ctrl Y...Zeile löschen.

Alt Y...UNDO, alten Zeileninhalt zurückholen.

INS.....Ein Leerzeichen einfügen.

DEL.....Zeichen löschen.

Ctrl G...goto; springt zum angegebenen Label oder zur Zeile.

Ctrl F...suchen, (find); sucht die angegebene Zeichenkombination.

Ctrl R...ersetzen, (replace); ersetzt die gesuchte Zeichenkombination durch die Neue.

## 3.5.1 Ablaufprogramm laden

3.5.1

-> Dir-D- Block-Manipulation (Querverweise)

-L- Laden eines Arbeitsprogramms.
Im Helpfeld erscheint eine invertierte Zeile.
Hat bereits vorher ein Ladevorgang stattgefunden, so wird der zuletzt benutzte Programmname vorgeschlagen.

Geladen wird der gesammte Programmbereich, 0-9999, beginnend mit der Zeile 0000.

Den Namen des Arbeitsprogramms (max. 30 Zeichen) in die invertierte Zeile schreiben und mit Eingabe von -CR- (Return) den Ladevorgang aktivieren. (ACHTUNG: DOS Format xxxxxxxx.rob)

ACHTUNG:

Ein bereits geladenes Programm wird überschrieben!

HINWEIS:

Eine Liste aller verfügbaren Arbeitsprogramme erhalten Sie mit der Eingabe von -D-.

Kapitel

## 3.5.2 Ablaufprogramm sichern

3.5.2

-> Dir-D- Block-Manipulation (Querverweise)

-S- Sichern eines Arbeitsprogramms.
Im Helpfeld erscheint eine invertierte Zeile.
Hat bereits vorher ein Sicherungsvorgang stattgefunden, so wird der zuletzt benutzte Programmname vorgeschlagen.

Gesichert wird der gesammte Programmbereich, 0-9999, beginnend mit der Zeile 0000.

Den Namen des Arbeitsprogramms (max. 30 Zeichen) in die invertierte Zeile schreiben und mit Eingabe von -CR- (Return) den Sicherungsvorgang aktivieren. (ACHTUNG: DOS Format 8(Zeichen).rob)

ACHTUNG:

Ein bereits benutzter Programmname wird überschrieben!

## 3.5.3 Blockmanipulation 3.5.3

Kommandos:

Ctrl B...Block Start markieren. Stellen Sie vorher den Cursor an den Anfang der ersten Zeile des Blocks.
Die Blockmarkierung beginnt mit dem Blockanfang und endet bei Zeile 9999.

Ctrl K...Block Ende markieren. Stellen Sie vorher den Cursor auf den Anfang der ersten Zeile, die dem Block folgt.
Die Blockmarkierung kennzeichnet nun den gesamten Block.

Alt B...zum Blockstart. Der Cursor springt an den Anfang der ersten Blockzeile.

Alt K...zum Blockende. Der Cursor springt zum Anfang der ersten Zeile nach dem Block.

Alt L...Block laden.
Namen des Blocks in die invertierte Zeile schreiben und mit -CR- holen.

ACHTUNG:

| |
|---|
| Wird der Block nicht gefunden, (falscher Name od Ä.) so erscheint in der untersten Zeile des Schirms eine Fehlermeldung. Aufhebung dieser Fehlermeldung mit -CR-. |

Eingefügt wird der Block oberhalb der Zeile, in der der Cursor steht.
Der gesammte Block wird durch die Blockmarkierung gekennzeichnet.
Alle nachfolgenden Zeilen werden um die Länge des Blocks verschoben.
Die Sprungadressen ändert das Programm selbstständig.

Alt S...Block sichern.
Namen des Blocks in die invertierte Zeile schreiben und mit -CR- sichern.

ACHTUNG:

| |
|---|
| Stellen Sie sicher, daß Sie einen bereits abgelegten Block gleichen Namens nicht überschreiben. |

Ctrl V...Block verschieben. Der Block wird oberhalb der Zeile eingefügt, in der der Cursor steht. Nach der Verschiebung steht der Cursor in der ersten Zeile des so verschobenen Blocks an seiner neuen Stelle.

Ctrl D...Block löschen. Der gesamte, markierte Block wird gelöscht. Er ist damit unwiederbringlich verloren! (-> sichern)

Ctrl I...Block einfügen. Der markierte Block wird oberhalb der Zeile eingefügt, in welcher der Cursor steht.
Der alte Block bleibt erhalten, der neue Block ist eine Kopie. Die Kopie ist nicht markiert.

Ctrl H...Block verstecken. Der Block wird "versteckt", d.h. er wird für die Benutzeroberfläche unsichtbar. Er kann aber jederzeit mit den Blockkommandos wieder aktiviert werden.

Alt P...Block ausdrucken.
Der Ausdruck erfolgt mit Angabe des Programmnamens und der Zeilennummer direkt nach der Eingabe.

## 3.5.4 Programmeingabe 3.5.4

Mit Esc rufen Sie den Editor auf.
Sie können jetzt Arbeitsprogramme erstellen und bearbeiten.

Mit dem Befehl Ctrl T sollten Sie sich vor Beginn der Programmierung einen Tabulator anlegen.
Dies erleichtert das spätere Zurechtfinden in einem "alten" Programm ganz wesentlich.

Wir schlagen Ihnen nachfolgende Aufteilung vor:
(Mit dieser Aufteilung erstellen wir alle Beispielprogramme)

```
0    Tab 1          Tab 2

Labels:
     Befehle        ;Kommentare, Dukumentation u.s.w.
```

Mit Ctrl T können sie eine Tabulatormarke setzen, oder, wenn bereits vohanden, löschen.

HINWEIS:

Sie können jede Programmzeile mit einem Kommentar versehen.
Dazu stehen Ihnen insgesammt 253 Zeichen (abzüglich des Programmteiles) in jeder Zeile zur Verfügung.

Textzeilen, die länger sind als das Editorfenster breit ist, werden (nach links) gerollt. Optional ist ein Spaltenzähler in der TAB-Zeile vorgesehen.

Durch das Zeichen ; (Semikolon) wird der Kommentar vom Programmteil abgetrennt.

Beispiel: (Simultanblock)

```
     ■              ■ (TABzeile)
HOLEN:              ;Label, (Name), Antrieb 3 und 4 zur Position
     M3+250         ;bewege Antrieb 3, rechtsdrehend, 250 Pulse
     M4+300         ;bewege Antrieb 3, rechtsdrehend, 300 Pulse
     start          ;alle Antriebe oberhalb dieses Startbefehls
                    ;simultan starten
```

HINWEIS:

Mit -F1- können Sie, auch während des Editierens die Helpfunktion aufrufen.

Nach erfolgter Auswahl verlassen Sie mit -Esc- das Hilfs-Menü.
Der letzte Inhalt des Help-Fensters bleibt jedoch erhalten und kann so Ihre Eingabe unterstützen.

Frei für Ergänzungen.

## 3.6 Startmenü

3.6

Mit -G- können Sie das Startmenü aufrufen.

START-MENO

-> START TRACE.

G Starte das Programm bei Cursor, d.h. der Programmlauf beginnt mit dem Befehl auf dem der Cursor steht. Der Cursor ist markiert durch die Invertierung der Zeilennummer. Er läßt sich mit den Cursortasten (2 u.8) verschieben.

S Einzelschritt (single step). Bei der Eingabe von -S- arbeitet das Programm nur die Zeile ab, in welcher der Cursor steht. Jede weitere Eingabe von -S- bewirkt einen weiteren Schritt. Nach der Eingabe von -G- wird der Programmlauf kontinuierlich fortgesetzt.

HINWEIS:
Zum Austesten von neuen Programmen ist der Single-Step Betrieb sehr gut geeignet.
Sie haben hierbei die Möglichkeit alle Veränderungen über die Anzeige auf dem Bildschirm zu verfolgen.

HOME Start vom Anfang (Zeile 0). Das Programm wird mit dieser Anweisung grundsätzlich von Zeile 0000 gestartet, egal wo der Cursor steht.

Esc Abbruch, zurück zum Hauptmenü. Sie verlassen das Startmenü.

-- Während des Programmlaufes haben Sie folgende Möglichkeiten --

NOTAUS mit Taste BREAK/Scroll/Roll. Das Programm wird unterbrochen, alle Antriebe werden abgeschaltet.
Mit -Esc- kehren Sie ins Hauptmenü zurück.

ACHTUNG:

Wenn Ihr Programm keine Antriebe mit Rückmeldung steuert, dann ist dies nur eine Unterbrechung. Sie können das Programm mit -S- oder -G- fortsetzen.
In allen anderen Fällen ist dies ein unkontrollierter Abbruch, NOTAUS und zieht die neue Initialisierung nach sich!

F1 Anzeige aktueller Roboter. Während eines Programmlaufes, auch oder gerade wenn TACE = 0 gesetzt ist, können Sie sehen, welchen Roboter das Programm gerade anspricht.

F2 Anzeige letzter Portzugriff. Hier sehen Sie ebenfalls, auf welches Port das Programm zuletzt zugegriffen hat.

F3 Robot Nummer -1. Wenn Sie während eines Programmlaufes einen bestimmten Roboter oder ein bestimmtes Port beobachten wollen, dann können Sie mit F3 - F6 die Auswahl treffen.

F4 Robot Nummer +1

F5 Port Nummer -1

F6 Port Nummer +1

# 3.7 Roboterbewegungen programmieren 3.7

Für die Programmierung von Roboterbewegungen stehen verschiedene Verfahren zur Verfügung.

1. Teach-In
   Eingabe der Wegstrecken durch manuellen Bewegungsaufruf über die Pc-Tastatur oder eine spezielle Teach-Box.

2. PTP, Punktsteuerung
   Eingabe der Fixpunkte zu denen sich die einzelnen Antriebe hin bewegen sollen.

3. Wegsteuerung
   Eingabe einer Anzahl von Pulsen je Antrieb. Die Anzahl der Pulse bestimmt die Länge der Wegstrecken.

4. CP, Bahnsteuerung
   Nach Eingabe verschiedener Parameter berechnet das Programm die Vorgaben für die verschiedenen Antriebe.

Für erste Bewegungsstudien steht die Action Box zur Vefügung.

Mit der Action Box ist es möglich, jeden Antrieb von TEACH-ROBOT über die Sub-Min-Buchse anzusprechen.
Außer der Action Box benötigen Sie noch eine (möglichst einstellbare) Stromquelle. Die höchstzulässige Spannung ist 20 Volt.
Die technische Wirkungsweise der Action Box ist im Kapitel 2.1.8 beschrieben.

Bevor Sie versuchen, eine komplexe Bewegung zu programmieren, versuchen Sie bitte diesen Bewegungsablauf zu analysieren und in Einzelbewegungen zu zerlegen.
Erstellen Sie am Anfang ruhig eine Liste, in welche Sie den Bewegungsablauf chronologisch eintragen.

Erst nachdem Sie sich darüber klar geworden sind, welche Bewegung zu welcher Zeit sinnvoll ist, können Sie daran gehen, einige dieser Einzelbewegungen zu sogenannten Simultanblöcken zusammenzufassen.
Ein Simultanblock ist die Zusammenfassung von Bewegungen verschiedener Antriebe, die aber gleichzeitig gestartet werden.

# 3.7.1 Teach-In 13.7.1

Teach-In ist das gebräuchliste Verfahren zur Programmierung von Bewegungsabläufen durch Industrieroboter.

Mit F3 können Sie während des Editierens das Programmsegment Teach-In aufrufen.

Es erscheint im Help-Feld:

TEACH-IN
-> Action-Box (zurück mit -F3-)
Roboter-Bewegungen:

| | |
|---|---|
| 1-Finger | Q-Finger auf |
| 2-Hand Dreh links | W-Hand D rechts |
| 3-Hand aufwärts | E-Hand abwärts |
| 4-Arm aufwärts | R-Arm abwärts |
| 5-Schulter zurück | T-Schulter vor |
| 6-Körper links | Z-Körper rechts |

Programmieren einer (mehrerer) Wegstrecke(n) durch manuelles Anfahren des Zielpunktes.

Die Antriebe bewegen sich solange, wie die entsprechende Taste gedrückt bleibt.

Mit -CR- werden die so entstandenen Streckenwerte in das Programm übernommen.

Diese Zielpunktwerte werden als Liste in das Arbeitsprogramm übernommen.
Unter die Auflistung setzt das Programm den Befehl START.
Dies bedeutet, daß alle, oberhalb von START stehenden Motoren, simultan gestartet werden.

ÄNDERUNGEN:
Die so entstandenen Werte lassen sich, zu einem späteren Zeitpunkt, sehr einfach mit dem Editor ändern.
Die Korrektur kann, direkt nach Eingabe des geänderten Wertes, mit der Taste F10 vollzogen werden.

HINWEIS:
Die Bewegung wird ausgeführt.
Sie können durch nochmaliges Drücken der Taste F10 diese Bewegung wieder rückgängig machen. Der Roboter fährt dann auf die Position zurück, die er vor der Eingabe hatte.

Nach der letzten Korrektur wird der geänderte Wert mit CR in das Arbeitsprogramm übernommen.

## 3.7.2 Punktsteuerung

3.7.2

Gebräuchliche Bezeichnung: PTP, point to point. (Punkt zu Punkt)

Bei der Punktsteuerung wird den Antrieben vorgegeben, bis zu welchem Punkt sie sich bewegen sollen.

Der angesprochene Antrieb bewegt sich gradlinig, auf dem kürzesten Weg auf diesen Punkt zu.

Stellen Sie sich die Wegstrecken für die sechs Antriebe des TEACH-ROBOT als Zahlenstrahlen vor.

| | |
|---|---|
| Finger | 0——100——200—250 |
| Hand Dreh | 0——100——200——300 |
| Hand Schwenk | 0——100——200——300——400——500——600——700——800 |
| Arm | 0——100——200——300——400——500——600——700——800 |
| Schulter | 0——100——200——300——400——500——600——700—750 |
| Körper | 0——100——200——300——400——500——600——700-//-1400 |

Die Längen der einzelnen Strecken sind je nach Antrieb verschieden.

ANMERKUNG:
Durch Fertigungstoleranzen und die Möglichkeit, den Referenzpunkt nachträglich (bei Bedarf) selbst verstellen zu können, sind die Wegstrecken von Roboter zu Roboter geringfügig unterschiedlich.

Die Eingabe bei der ROBOT-SPS erfolgt mit dem Befehl:

Mx=yyy

Beispiel:

```
M1=150          ;Motor 1 nach Position 150.
                ;die Position 150 ist der Punkt, der 150 Flan-
                ;kenwechsel vom Referenzpunkt (0) entfernt ist.
                ;Weil das Programm die aktuelle Position aller
                ;angemeldeten Motoren kennt, kann es entschei-
                ;den, ob zum Erreichen dieses Punktes der Motor
                ;rechts- oder linksdrehend eingeschaltet werden
                ;muß.
```

HINWEIS:
Bei dieser Art der Eingabe brauchen Sie keine Kenntnis darüber, ob der Restweg des Antriebes ausreicht, den resultierenden Weg auch tatsächlich zurücklegen zu können.

## 3.7.3 Wegsteuerung

3.7.3

Diese Form der Steuerung ist im Prinzip ebenfalls eine Punktsteuerung (PTP).
Wir haben hier nur deshalb eine Unterscheidung vorgenommen, weil die ROBOT-SPS unterschiedliche Befehle für PTP kennt.

Auch hier wird dem Antrieb vorgegeben, bis zu welchem Punkt er sich bewegen soll.
Der angesprochene Antrieb bewegt sich gradlinig, auf dem kürzesten Weg auf diesen Punkt zu.

Die Eingabe bei der ROBOT-SPS erfolgt mit den Befehlen:

Mx-yyy und Mx+yyy

Beispiel:

```
TEST:                ;Label
     HOME            ;Referenzposition anfahren
     M2-100          ;Motor 2, linksdrehend bewegen, bis 100 Pulse
                     ;erfasst sind.
     m3+200          ;Motor 3, rechtsdrehend bewegen, bis 200 Pulse
                     ;erfasst sind.
     start           ;Startbefehl
     jump test       ;Schleife, zum Anfang
```

HINWEIS:
Bei dieser Art der Eingabe brauchen Sie Kenntnis darüber, ob der Restweg des Antriebes ausreicht, den gewünschten Weg auch tatsächlich zurücklegen zu können.

Sie können den noch möglichen Restweg aber leicht ermitteln, indem Sie beim Einzelschrittbetrieb die Anzeige Pos AKT beachten.
Hier wird der aktuelle Stand der einzelnen Antriebe angezeigt.

Es gilt: Gesamtweg - Teilweg = Restweg

Für unser Beispiel bedeutet das:

Motor 2 und 3 sind in die Referenz gefahren (Nullposition). Bei beiden Antrieben steht also der Gesamtweg als Restweg zur Verfügung.

Motor 2: (M2-100) Da der Motor bereits mit HOME linksdrehend in den Endschalter gefahren ist, kann er sich nicht weiter bewegen.
Motor 3: (M3+200) Hier gilt: 800 - 200 = 600 (Restweg)
Der Arm kann sich 200 Pulse lang abwärts bewegen.

| Anzeige bei: | M2 | M3 |
|---|---|---|
| Pos AKT | -1 | +200 |
| Pos NOM | -100 | +200 |

Übung mit kleinen Testprogrammen bringt Ihnen hier Sicherheit.

Gebräuchliche Bezeichnung: CP, continous path.

Programmsegment in Vorbereitung.

Verfügbar, voraussichtlich ab September 1992.

Frei für Ergänzungen.

# 3.8 Befehlssatz der ROBOT-SPS 3.8

Die ROBOT-SPS verfügt über 55 Stammbefehle.

Sie bilden 3 Gruppen:

1. Spezielle Roboterbefehle:

   -> ROBOT RAMP GRIP TRACE NOP STOP

   -> M1+ M1- M1= START LEFT RIGHT

   -> HOME HALT PROTOCOL PAUSE

2. Allgemeine Befehle:

   -> JUMP CALL RET PUSH POP LD ST

   -> LDB STB LDP STP XCHG XCHGC ADD

   -> SUB ADC SBC CMP OR XOR AND CLC

   NOT NEG ABS MUL(S) DIV(S) ROL ROR

   J(N)E J(N)C JPL JMI J(N)Z SHL SHR

3. Kommunikationsbefehle:

   -> AKT/NOM Color Print Input CLS

   HOME setzt den Zykluszähler + 1

ANMERKUNG:
Der Befehl HOME hat die zusätzliche Funktion, den Zykluszähler um 1 zu erhöhen.
Damit haben Sie einen Zähler, der Ihnen mitteilt, wieviele Schleifen Ihr Programm bereits durchlaufen hat.

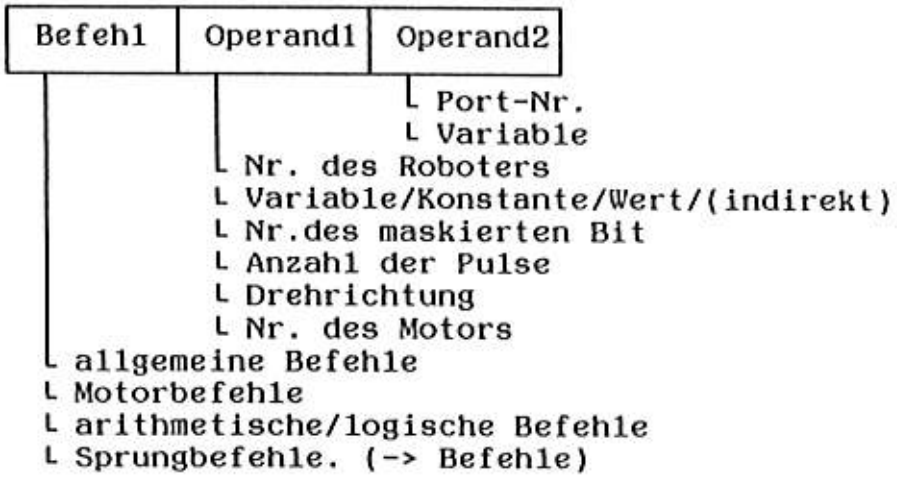
Im Help-Feld können Sie diese Obersicht unter dem Stichwort Befehle darstellen.
Sie haben so, während der Programmerstellung, immer eine hilfreiche Kurzübersicht über alle verfügbaren Befehle.

## 3.8.1 Syntax (Satzbau), Label 3.8.1

SYNTAX

Ein Befehl kann bestehen aus:

Erstens: Befehl
Zweitens: Befehl Operand1
Drittens: Befehl Operand1 Operand2




LABEL

Die ROBOT-SPS kennzeichnet Unterprogramme mit Labels, d.h. jedes Unterprogramm (Subroutine) hat eine Bezeichnung, die aus maximal 30 Buchstaben bestehen kann.

Schreibweise: EINXBELIEBIGERNAME: (Großschreibung empfohlen)
Ein Label wird vom Programm erst dann als solches erkannt, wenn es mit einem DOPPELPUNKT (:) abgeschlossen ist.

ACHTUNG:

> Ziffern am Anfang und Satzzeichen sind für die Kennzeichnug von Labels NICHT zugelassen. Ebenso KEIN Leerzeichen! Das LABEL muß IMMER an den Zeilenanfang!

Eine Liste aller bereits verwendeten Labels können Sie mit F4 aufrufen.

KOMMENTAR

Sie haben die Möglichkeit, jede Programmzeile mit einem Kommentar zu versehen.
Die Trennung vom Befehlsteil erfolgt mit SEMICOLON (;). (->TAB)
Jede Programmzeile darf maximal 253 Zeichen lang sein! (Befehl + Kommentar. Es sind alle Zeichen zugelassen.

## 3.8.2 Adressierungen

3.8.2

Unter Adressierungen verstehen wir hier den Bezug, den die Operanden haben können.

| Adressierungsart: | Erläuterung: |
|---|---|
| Konstante | -32768 bis 65535, ein beliebiger Wert dazwischen. |
| Konstante hex | $0000 bis $FFFF, hexadezimal, ein beliebiger Wert dazwischen. |
| Konstante binär | %00001101, für eine Übersichtliche Darstellung, insbesondere bei maskierten Ein-/Ausgaben, sollten Sie diese Form wählen. |
| Variable | A bis Z, alle Buchstaben des Alphabetes, Groß- oder Kleinschreibung. (A = a) |

ANMERKUNG:
Wenn Sie die ROBOT-SPS von einer Hochsprache aus ansprechen wollen, haben Sie die Möglichkeit, die Parameter für das Arbeitsprogramm in diesen Variablen abzulegen.

| | |
|---|---|
| Indirekt | (A) >0-65535<, der Inhalt einer Variablen darf hierbei zwischen 0 und 65535 liegen. Die Variable ist in Klammern zu setzen. |

ANMERKUNG:
Bei dieser Adressierungsart können, zusätzlich zu den normalen Variablen, weitere Speicherplätze angesprochen werden.
Die ROBOT-SPS organisiert diese Plätze intern, d.h. der Inhalt der Variablen ist NICHT die Adresse eines Speicherplatzes im PC, sondern nur die Nummer der Zeile des internen Speichers, den das Programm für diese Operationen reserviert hat.

Beispiel:

```
ld 65500        ;lade den dezimalen Wert 65500 in den Akku.
st a            ;lege den Inhalt des Akkus in der Variablen
                ;a ab.
ld $FF          ;lade den Akku mit dem (hex) Wert FF.
st (a)          ;speichere den Inhalt des Akkus an die Stelle
                ;des internen Speichers den die Variable a
                ;bezeichnet. (Hier: Zeile 65500 des STSP)
```

HINWEIS:
Ihr PC sollte für diese Operationsmöglichkeit einen Arbeitsspeicher von mindestens 640 KByte haben. Ist der Arbeitsspeicher kleiner, so ist die Obergrenze der indirekten Adressierung von 65535 auf einen entsprechend niedrigeren Wert zu korrigieren!

| | |
|---|---|
| AKTx | Ist-Position, lesen oder überschreiben der Ist-Position. |
| NOMx | Soll-Position, lesen oder überschreiben der Soll-Position. |

## 3.8.3 Fehlermeldungen

3.8.3

Fehlermeldungen werden zum Einen während eines Arbeitsprogrammlaufes und zum Anderen im Editor ausgegeben.

Die Eingabe (in der Editor-Funktion) wird ständig auf Fehler abgeprüft.
Dies soll die Programmierung erleichtern und helfen, Fehleingaben zu vermeiden.
Ein Fehler wird lokalisiert und in der Zeile oberhalb der ersten Programmzeile mit einem Pfeil markiert.

ACHTUNG

Im Editor erhalten Sie solange eine Fehlermeldung, bis das Programm einen gültigen Befehl oder Wert erkennt und akzeptiert. Verlassen Sie die Programmzeile aber, ohne eine Fehlerkorrektur vorgenommen zu haben, so bleibt dieser Fehler erhalten !!!

| FEHLERMELDUNG: | Bedeutung: |
|---|---|
| Zahl fehlt | ->kein Wert angegeben |
| | ->falsches Zeichen |
| falsche Adressierung | ->Schreibweise nicht korrekt |
| Komma fehlt | ->Komma setzen (,) |
| Zahl zu groß | ->zuviele Stellen |
| | ->Wert zu groß |
| Klammer fehlt | ->Klammer ( ) setzen |
| Zeichen nicht zugeordnet | ->zuviele Zeichen |
| | ->; fehlt vor dem Kommentar |
| falsches Zeichen | ->Zeichen ungültig |
| Wort falsch oder fehlt | ->Bezeichnug falsch (zuviele Zeichen) |
| | ->Wort fehlt |
| unbekanntes Kommando | ->Befehl unvollständig |
| | ->Befehl falsch |
| | ->Leerzeichen nach Befehl fehlt |
| unbekannte Sprungmarke | ->Label falsch |
| | ->Label unvollständig |
| | ->Label unbekannt |

Während des Programmlaufes erscheint eine mögliche Fehlermeldung in der letzten Zeile des Programmfeldes.

-> ROBOT RAMP GRIP TRACE NOP STOP
-> HOME HALT PROTOCOL PAUSE
-> Color Print Input CLS

ROBOT

Robot 1
- 1 — Nummer des Robots (1-8)
- Robot — Befehlsname

Wahl des Roboters.
Dieser Befehl schaltet zwischen den Karten Ui1/Robot um !
Er muß grundsätzlich am Anfang einer Programmsequenz für einen bestimmten Roboter stehen. Alle nachfolgenden Motor-Befehle werden auf diesen ausgewählten Roboter bezogen !
Die Wahl eines anderen Roboters bewirkt automatisch den Wechsel des Bezuges.

RAMP

RAMP 0 (100/50)
- 50 — Rampe dopp. Steilh.
- 100 — Rampe Standard-Wert
- 0 — Rampe aus
- RAMP — Befehlsname

Die RAMPE beeinflußt das Anlauf- und Bremsverhalten eine Antriebes.
Sie verhindert abrupte Massenbewegungen.
Standardwert: 0100 für 8MHz/286 µP. Für höhere Clock-Frequenzen kann dieser Wert nach oben korrigiert werden.

GRIP

Grip 0 (1)
- 1 — Greifer wird nur mit M1+/-/= bewegt !
- 0 — Greifer mit jedem Startbefehl (beliebiger Motoren) schließen.
- Grip — Befehlsname

1 = Der Greifer versucht mit jedem nachfolgenden Startbefehl zu schließen, d.h. Motor 1 wird linksdrehend eingeschaltet. Die Abschaltung geschieht, nach Time-Out automatisch.
Motor 1 (Finger) möglichst seperat d.h. nicht zusammen mit anderen Motoren programmieren !

TRACE

Trace 0 (1)
- 1 — Schirm ein
- 0 — Schirm aus
- Trace — Befehlsname

Bildschirm-Aktualisierung während des Programmlaufes.
Trace 1 = Darstellung des Ablaufes sowie aller Änderungen auf dem Bildschirm.
Nachteil: Das Arbeitsprogramm wird langsamer.
Trace 0 = Keine Änderung des Bildschirmes.

NOP

```
NOP
└──────────────Befehlsname
```

No operation (Leerschritt). Bei diesem Befehl führt der Prozessor (PC) keine Operationen durch.
Geeignet zur Bildung von Warteschleifen < 100 ms.

```
Beispiel:       ;Kommentar
TIME:           ;Label
     ld 0       ;Akku = 0
     loop       ;Schleife
     nop        ;hier wird die Schleifenzeit durch die Anzahl
     nop        ;der nop-Schritte fein beeinflußt
     add 1      ;Akku + 1
     cmp 3      ;Akku = 3 ?
     jne loop   ;nein->loop
     ret        ;zurück
```

STOP

```
STOP 1
│    └──────────Nr.d. Motors (1-6)
└──────────────Befehlsname
```

Dieser Befehl schaltet den bezeichneten Motor ab.
Die Drehrichtung ist dabei unerheblich.
ACHTUNG

| Bei diesem Befehl stoppt der Motor NICHT unter Pulskontrolle! |
|---|

HOME

```
HOME
└──────────────Befehlsname
```

Die gesetzten Antriebe fahren in die Referenzposition.
Die Abschaltung erfolgt NUR durch die mechanischen ENDSCHALTER.
Alle Positonswerte (AKT/NOM) werden zu 0 gesetzt.

ANMERKUNG:
Der Befehl HOME hat die zusätzliche Funktion, den Zykluszähler um 1 zu erhöhen.
Damit haben Sie einen Zähler, der Ihnen mitteilt, wieviele Schleifen Ihr Programm bereits durchlaufen hat.

HALT

```
HALT
└──────────────Befehlsname
```

Stopanweisung im Programm. Mit diesem Befehl wird der Programmlauf UNTERBROCHEN. Eine Fortsetzung ist NUR manuell, also durch Bedienung Über die Tastatur möglich. Dieser Befehl kann als - BREAKPOINT - bei der Fehlersuche verwendet werden.

PAUSE

```
PAUSE xxxx
  |     └──────Wert/Variable
  └────────────Befehlsname
```

Unterbrechung des Programms um (1-9999 mal 0,1) Sekunden.
Kleinste Pause 0,1 sec. Größte programmierbare Pause 999,9 Sec.

```
Beispiel:       ;Kommentar
PAUSE10:        ;Label
     pause 50   ;50 x 0,1 = 5 Sekunden Warteschleife.
     ret        ;Rücksprunganweisung
```

PROTOCOL

```
PROTOCOL 0 (1)
  |        |  └──Zeichen zum Drucker-Port
  |        └─────Zeichen nicht zum D-Port
  └──────────────Befehlsname
```

Mit PROTOCOL wird beim Befehl Print, der sonst ausschließlich im Help-Feld wirksam wird, eine Ausgabe auch auf das Druckerport übertragen.

```
Beispiel:             ;Kommentar
DRUCK:                ;Label
     protocol 1       ;die Ausgabe an den Drucker wird aktiviert.
     print "Ausgabe";der Text -Ausgabe- wird im Helpfenster
                      ;dargestellt UND ausgedruckt.
     protocol 0       ;Ausgabe an den Drucker desaktivieren.
     ret              ;Rücksprunganweisung
```

Damit sind Protokolle auch während eines Programmlaufes möglich.
Ausgabe beliebiger Hexkonfigurationen möglich, -> Steuerung über Drucker-Port.

Befehle zur Bestimmung der (ersten) Printposition ->CURSOR<-

```
LINE x
  |   └──────Nummer der Zeile
  └──────────Befehlsname
```

Der Cursor wird in die Zeile x (0-14) des Helpfeldes gestellt.

```
COLUMN x
  |     └────Nummer der Spalte
  └──────────Befehlsname
```

Der Cursor wird in die Spalte x (0-34) des Helpfeldes gestellt.
(Bei Ausgabe an den Drucker bis 79)
Das erste Zeichen wird nun an der Curserposition ausgegeben.

PRINT

```
PRINT "x",
|     |  └──────Cursorposition bleibt in der Zeile
|     └─────────Text
└───────────────Befehlsname
```

Befehl zur Ausgabe von Zeichen innerhalb des Helpfeldes.
Alle Ascii Zeichen, auch die einfachen Grafikzeichen zugelassen.
Damit ist eine freie Gestaltung des Helpfeldes während des Arbeitsprogramm-Ablaufes möglich.

> ACHTUNG: Wenn Sie das Komma weglassen, dann macht der Printbefehl zusätzlich linefeed und carriage return.
> (neue Zeile Anfang)
> Beide Befehle, PRINT und INPUT bieten eine komfortable Möglichkeit während des Programmlaufes mit dem PC zu kommunizieren!
> Die Darstellung von eigenen Texten und einfachen Grafiken bietet die optimale Möglichkeit zur Überwachung von komplexen Abläufen.

Weitere PRINT Variationen:

| | |
|---|---|
| PRINT | neue Zeile Anfang |
| PRINTF, | Darstellung/Druck des Akkuinhaltes dezimal, hexadezimal und binär in 1 Zeile |
| PRINTD x, | Darstellung/Druck von x dezimal |
| PRINTH x, | Darstellung/Druck von x hexadezimal |
| PRINTB x, | Darstellung/Druck von x binär |
| PRINTC x, | Ascii-Zeichen ausgeben |

INPUT

```
INPUTC x
|      └──────Eingabe (Tastenunmmer, scan code) von der
|              PC-Tastatur abholen und bei x ablegen
└──────────────Befehlsname
```

Befehl zur Abholung eines Zeichens von der Tastatur.
Das Programm wartet in dieser internen Routine, bis ein Zeichen von der Tastatur eingetroffen ist.
Die Tastennummern von F1 - F6 können nicht abgeholt werden, da diese während des Programmlaufes die Roboter- und Portanzeige einstellen!
Ebenso entfallen die Tasten Break, Shift, Ctrl und Alt.

```
INPUTQ x
|      └──────Tastatur-Status (Taste im Tastaturpuffer)
|              einlesen (in den Akku)
|               0 = keine Taste gedrückt worden
|              >0 = Taste gedrückt worden; (->scan code)
└──────────────Befehlsname
```

> scan code: Tastennummer, welche die Lage der Taste auf der Tastatur beschreibt.

Weitere INPUT Variationen:

| | |
|---|---|
| INPUTF x | Zahl (max. 4 stellig) in beliebiger Form eingeben. $=hex, %=binär, sonst dezimal |
| INPUTH x | Zahl als Hexformat einlesen |
| INPUTB x | Zahl als Binärformat einlesen |

ACHTUNG:

Das Programm wird dabei für die Eingabe einer Zahl unterbrochen.
Es wartet auf eine Eingabe!
Nach Eingabe einer Zahl und Return (CR / Enter) wird das Programm automatisch fortgesetzt.

CLS

CLS
└──────────Befehlsname

Dieser Befehl löscht das Help-Feld komplett.

AKT/NOM

| | |
|---|---|
| IST-Position | AKT1 - AKT7 |
| SOLL-Position | NOM1 - NOM7 |

AKT und NOM lassen sich mit den Befehlen -LD AKT1- einlesen und mit -ST AKT1- überschreiben.

So können Sie Nullpunktkorrekturen und Bahnbeeinflussungen durchführen.

Für die Programmtechnische Stellfehler-Korrektur ist dies erforderlich.
Sie erfassen hierbei den Wert der Abweichung je Bewegung und kompensieren diesen Stellfehler, indem Sie, direkt oder aufaddiert, den nachfolgenden Ablauf beeinflußen.

## 3.8.3.2 Motor Befehle mit Pulskennung 3.8.3.2

-> M1+ M1- M1=

M1+

M1+xxxx
- Anzahl der Pulse
- Drehrichtung rechts
- Nr.des Motors, 1-6
- Befehlsname

Motor 1 (1-6) rechtsdrehend einschalten, mit PULSKONTROLLE.

Beispiel:
```
Armrunter:          ;Label
     M4+200         ;Motor 4 (Arm) bewegt sich, rechtsdrehend
                    ;abwärts bis 200 Pulse erfasst sind.
     START          ;Startbefehl
```

HINWEIS:
Für die programmierte Pulsmenge muß eine ausreichend große Wegstrecke (hier min.200) zur Verfügung stehen, bevor der Antrieb rechtsdrehend in seinen Endschalter fährt.

M1-

M1-xxxx
- Anzahl der Pulse
- Drehrichtung links
- Nr.des Motors, 1-6
- Befehlsname

Motor 1 (1-6) linksdrehend einschalten, mit PULSKONTROLLE.

HINWEIS:
Der Antrieb kann linksdrehend nur dann eine vorgegebene Strecke fahren, wenn die Ausgangsposition mindestens den Wert der Strecke vom Nullpunkt (REFERENZ) entfernt war.

M1=

M1=xxxx
- Zielpunkt-(Adresse)
- Drehrichtung rechts/links
- Nummer des Motors, 1-6
- Befehlsname

Motor 1 (1-6) in die angegebene Zielposition fahren.

Beispiel:
```
ARMRELI:            ;Label
     M4=200         ;Motor 4 bewegt sich rechts-/ od. linksdrehend,
                    ;je nach Ausgangslage, bis die Position
                    ;(Adresse) 200 erreicht ist. Der Fixpunkt ist
                    ;hier 200 Pulse vom O-Punkt (REFERNZ) entfernt.
                    ;Der Arm bewegt sich aufwärts oder abwärts.
```

HINWEIS:
Bei dieser Art der Positionsangabe ist keine Kenntnis der noch verfügbaren Restwegstrecke erforderlich!

3.8.3.2 Motor Befehle ohne Pulskennung und START 3.8.3.2

-> LEFT RIGHT

LEFT

```
LEFT 1
|    └──────Nummer des MOTORS (1-6)
└───────────Befehlsname
```

Motor 1 (1-6) linksdrehend einschalten, OHNE Pulskontrolle.

ACHTUNG:

Nur der ENDSCHALTER oder ein Hindernis begrenzen hier den Lauf!

Zur Ausführung ist der START Befehl NICHT erforderlich!

Dieser Befehl ist geeignet zum Einschalten von endlos laufenden Motoren bei Förderbändern und Ähnlichem.

RIGHT

```
RIGHT 1
|     └──────Nr.d. MOTORS (1-6)
└────────────Befehlsname
```

Motor 1 (1-6) rechtsdrehend einschalten, OHNE Pulskontrolle.

NUR der Endschalter begrenzt den Lauf.

Zur Ausführung ist der START Befehl NICHT erforderlich!

Dieser Befehl ist geeignet zum Einschalten von endlos laufenden Motoren bei Förderbändern und Ähnlichem.

Der START Befehl.

START

```
START
└──────────Befehlsname
```

(Simultan-) Start Befehl für alle vorausgegangenen M-Befehle. Die Abschaltung (STOP) erfolgt für jeden Antrieb getrennt, unter Programmkontrolle.

## 3.8.3.4 Arithmetische-/logische Befehle 3.8.3.4

-> PUSH POP LD ST
-> LDB STB LDP STP XCHG XCHGC ADD
-> SUB ADC SBC CMP OR XOR AND CLC
-> NOT NEG ABS MUL(S) DIV(S) ROL ROR
-> SHL SHR

PUSH

```
PUSH
└──────────────Befehlsname
```

Rette den Inhalt des Akkus. PUSH legt den Akku auf den Stapel.
Sicherung vor einem Sprung zum Unterprogramm:

```
Push            ;rette Akku
call subroutine;springe zum Unterprogramm
pop             ;fülle Akku mit dem Inhalt vor dem Sprung
```

POP

```
POP
└──────────────Befehlsname
```

Holt den geretteten Inhalt des Akkus vom Stapel.

HINWEIS:

Die Befehle PUSH und POP können in der ROBOT-SPS auch zum Retten von Variablen direkt oder indirekt verwendet werden !

Beispiel:

```
ld 63000        ;lade Konstante zur Bezeichnung des internen Stapel-
                ;speichers über den Akku
st b            ;in die Variable b
ld 55           ;lade einen (dezimalen) Wert in den Akku
push (b)        ;lege den Inhalt des Akkus an dem, mit b bezeichne-
                ;ten Stapelspeicherplatz ab
ld 0            ;lade den Akku mit 0, alter Wert ist verloren !
pop (b)         ;rette alten Akkuinhalt von (b)
```

LD

```
LD x
│  └──────────Konstante/Wert
│  └──────────Variable
│  └──────────Adressierung, indirekt (X)
└─────────────Befehlsname
```

Lade den Akku, mit dem angegebenen Wert, dem Inhalt der Variablen, oder dem Wert unter der angegebenen Adresse.
Der alte Inhalt des Akkus wird überschrieben und geht verloren!
Ist x ein Wert, so kann dieser auf 3 verschiedene Arten angegeben werden:

```
1. Binär        ;%0101 0101
2. Hexadezimal  ;$55
3. Dezimal      ;55
```

3.8.3.4 Arithmetische-/logische Befehle 3.8.3.4

-> PUSH POP LD ST
-> LDB STB LDP STP ...

ST

```
ST x
|  └──────────Variable
|  └──────────Adresse, indirekt (X)
└─────────────Befehlsname
```

Speichere den Inhalt des Akkus nach x.
Der Inhalt des Akkus bleibt erhalten. Der Inhalt von x wird überschrieben!

LDB maskierter Input

```
LDB x,y
|   |  └──────Nummer des Ports
|   └─────────Nummer des maskierten Bits
└─────────────Befehlsname
```

Lade das/die maskierte(n) Bit(s) von Port y in den Akku.
Mit diesem Befehl ist das Einlesen einzelner Bits möglich.

STB maskierter Output

```
STB x,y
|   |  └──────Nummer des Ports
|   └─────────Nr. des mask. Bits
└─────────────Befehlsname
```

Schreibe die maskierten Bits vom Akku nach Port y.
Mit diesem Befehl ist es möglich, einzelne oder mehrere (maskierte) Bits in das Ausgangsport zu schreiben. Der Inhalt des Ausgangsports wird dabei NUR an den Stellen überschrieben, die maskiert sind.

Beispiel:
Aufgabe: Bit 5 des Akkus nach Port 2 schreiben. Akku = 0011 1000

```
                ;                Inhalt v. Port 2 = 1000 1111
    STB 5,2     ;schreibe bit 5 nach port 2
                ;Ergebnis:       Inhalt v. Port 2 = 1010 1111
```

LDP

```
LDP y
|   └──────Nummer des Ports
└─────────Befehlsname
```

Lade den Inhalt von Port y in den Akku.

STP

```
STP y
|   └──────Nummer des Ports
└─────────Befehlsname
```

Schreibe den Inhalt des Akkus nach Port y.

ACHTUNG:

Für 8 Bit Parallel-Ausgabe muß die Port-Adresse 83(15) sein!!!

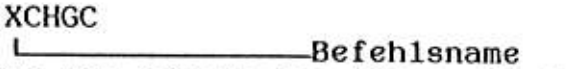
## 3.8.3.4 Arithmetische-/logische Befehle 3.8.3.4

-> PUSH POP LD ST
-> LDB STB LDP STP XCHG XCHGC ADD
-> SUB ADC SBC CMP OR XOR AND CLC
-> NOT NEG ABS MUL(S) DIV(S) ROL ROR
-> SHL SHR

XCHG

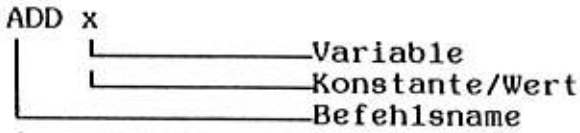

Vertausche den Inhalt von x mit dem Inhalt des Akkus.

XCHGC



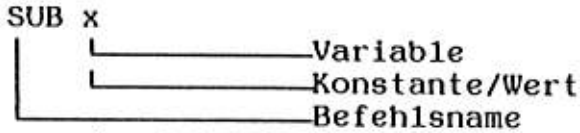

Vertausche den Inhalt des Akkus mit dem Inhalt von CARRY.

ADD



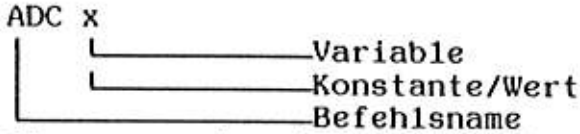

Addiere den Inhalt von x zum Inhalt des Akkus.
Das Ergebnis steht im Akku. Alter Inhalt des Akkus geht verloren!

SUB



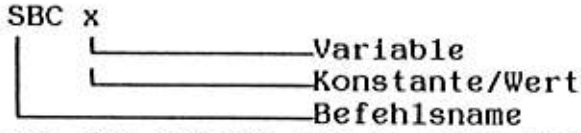

Subtrahiere den Inhalt von x vom Inhalt des Akkus.
Das Ergebnis steht im Akku. Alter Inhalt des Akkus geht verloren!

ADC




Der Inhalt von x, plus das CARRY Bit werden zum Inhalt des Akkus addiert. Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren!

SBC




Subtrahiere den Inhalt von x, plus das CARRY Bit, vom Inhalt des Akkus. Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren!

## 3.8.3.4 Arithmetische-/logische Befehle 3.8.3.4

-> PUSH POP LD ST
-> LDB STB LDP STP XCHG XCHGC ADD
-> SUB ADC SBC CMP OR XOR AND CLC
-> NOT NEG ABS MUL(S) DIV(S) ROL ROR
-> SHL SHR

CMP

```
CMP x
 |  └──────────Variable
 |  └──────────Konstante/Wert
 └─────────────Befehlsname
```

Vergleiche den Inhalt von x mit dem Inhalt des Akkus.
Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren! Der Condition-Code wird beeinflußt.

OR

```
OR x
 |  └──────────Variable
 |  └──────────Konstante/Wert
 └─────────────Befehlsname
```

Logische ODER Verknüpfung von x mit dem Inhalt des Akkus.
Das Ergebnis steht im Akku. Alter Inhalt des Akkus geht verloren!

XOR

```
XOR x
 |  └──────────Variable
 |  └──────────Konstante/Wert
 └─────────────Befehlsname
```

Logische EXKLUSIV-ODER Verknüpfung von x mit dem Inhalt des Akkus.
Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren!

AND

```
AND x
 |  └──────────Variable
 |  └──────────Konstante/Wert
 └─────────────Befehlsname
```

Logische UND Verknüpfung von x mit dem Inhalt des Akkus.
Das Ergebnis steht im Akku. Alter Inhalt des Akkus geht verloren!

CLC

```
CLC              clear carry
 └──────────────Befehlsname
```

Lösche das Carry Bit. Dieser Befehl wird gebraucht bei arithmetischen Befehlen und SCHIFT.

NOT

```
NOT
 └──────────────Befehlsname
```

Invertiere den Inhalt des Akkus.

-> PUSH POP LD ST
-> LDB STB LDP STP XCHG XCHGC ADD
-> SUB ADC SBC CMP OR XOR AND CLC
-> NOT NEG ABS MUL(S) DIV(S) ROL ROR
-> SHL SHR

NEG

```
NEG
└──────────────Befehlsname
```

Negiere den Inhalt des Akkus. Das Ergebnis steht im Akku.

ABS

```
ABS
└──────────────Befehlsname
```

Negiere den Inhalt des Akkus wenn CC (Codition Code) negativ.
Das Ergebnis steht im Akku.

MUL(S)

```
MUL(S) x
│      └──────Variable, auch (indirekt)
│      └──────Konstante/Wert
└─────────────Befehlsname
```

Multipliziere (vorzeichengerecht) den Inhalt von x mit dem Inhalt des Akkus. Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren!

ACHTUNG:

Die Klammern stehen nur zur Andeutung der anderen Möglichkeit.
Entweder heißt dieser Befehl:MUL (normale Multiplikation)
oder:MULS (vorzeichengerechte Multipl.)

DIV(S)

```
DIV(S) x
│      └──────Variable, auch (indirekt)
│      └──────Konstante/Wert
└─────────────Befehlsname
```

Dividiere (vorzeichengerecht) den Inhalt von x mit dem Inhalt des Akkus. Das Ergebnis steht im Akku. Der alte Inhalt des Akkus geht verloren!

ACHTUNG:

Die Klammern stehen nur zur Andeutung der anderen Möglichkeit.
Entweder heißt dieser Befehl:DIV (normale Division)
oder:DIVS (vorzeichengerechte Division)

## 3.8.3.4 Arithmetische-/logische Befehle 3.8.3.4

-> PUSH POP LD ST
-> LDB STB LDP STP XCHG XCHGC ADD
-> SUB ADC SBC CMP OR XOR AND CLC
-> NOT NEG ABS MUL(S) DIV(S) ROL ROR
-> SHL SHR

ROL rotiere den Akku links (rotate accu left)

```
ROL x
 |  └──────────Zahl der Schritte (1-8)
 └─────────────Befehlsname
```

Verschiebe den Inhalt des Akkus um x Stellen nach links.

Beispiel:

```
                ;Inhalt des Akkus = 00000001  (dez 1)
    rol 3       ;verschiebe den Inhalt des Akkus um 3 Stellen
                ;nach links.
                ;Ergebnis im Akku = 00001000  (dez 8)
```

Das CARRY Bit wird NICHT beeinflußt!

ROR rotiere den Akku rechts (rotate accu right)

```
ROR x
 |  └──────────Zahl der Schritte (1-8)
 └─────────────Befehlsname
```

Verschiebe den Inhalt des Akkus um x Stellen nach rechts.
Das CARRY Bit wird NICHT beeinflußt!

SHL

```
SHL x          schiebe den Akku x Stellen nach links
 |  └──────────Zahl (1-8)          (shift accu left)
 └─────────────Befehlsname
```

Schiebe den Inhalt des Akkus um 1 Stelle nach links.
Das CARRY Bit kann beeinflußt werden!

Beispiel:

```
                ;Inhalt des Akkus = 00000001  (dez 1)
    shl 3       ;verschiebe den Inhalt des Akkus um 3 Stellen
                ;nach links.
                ;Ergebnis im Akku = 00001000  (dez 8)
```

SHR

```
SHR x          schiebe den Akku x Stellen nach rechts
 |  └──────────Zahl (1-8)          (shift accu right)
 └─────────────Befehlsname
```

Schiebe den Inhalt des Akkus um x Stellen nach rechts.
Das CARRY Bit kann beeinflußt werden!

## 3.8.3.5 Sprung Befehle

3.8.3.5

-> JUMP CALL RET
-> J(N)E J(N)C JPL JMI J(N)Z

JUMP

JUMP xxxx
| └──────Label/Adresse
└──────────Befehlsname

Sprung, ohne Bedingung zum angegebenen Label (Adresse).
Das Programm wird an der gekennzeichneten Programmzeile fortgesetzt.

CALL

CALL xxxx
| └──────Label
└──────────Befehlsname

Sprung, ohne Bedingungen zum Unterprogramm bei Label xxxx.
Der Rücksprung muß mit RET am Ende des Unterprogramms sichergestellt sein.
Aufbau einer Subroutine:

```
LABEL:              ;Name, (maximal 30 Zeichen), die Groß- oder
                    ;Kleinschreibung spielt hierbei keine Rolle
     xxx            ;beliebiger Inhalt
                    ;des Unterprogramms
     RET            ;Befehl zum Rücksprung
```

Es sind maximal 128 Verschachtelungen möglich!
D.h.: Sie können bis zu 127 mal aus einem Unterprogramm in ein weiteres Unterprogramm springen.

ACHTUNG:

> Einer der Fehler die am häufigsten gemacht werden ist der, Register und Akku-Inhalte versehentlich, meist in einer aufgerufenen Unterroutine, zu überschreiben!
> RETTEN Sie diese Inhalte vorher mit PUSH und POP !

HINWEIS:
Ist das Label falsch angegeben oder fehlt es, so wird das Programm gestoppt und eine Fehlermeldung ausgegeben.

RET

RET
└──────────Befehlsname

Rückkehr vom Unterprogramm. Rücksprung zur Ausgangsadresse +1.

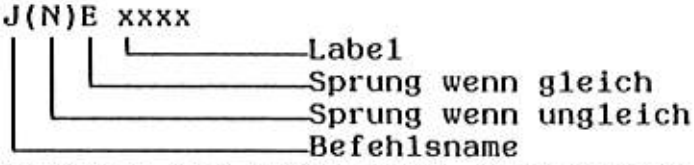
HINWEIS:
Wird die ROBOT-SPS von einer Hochsprache aus (->Softw.Schnittst.) aufgerufen, so ist das Ende einer kpl. Programmsequenz mit -RET- abzuschließen. Dies erst ermöglicht den gültigen Rücksprung.

## 3.8.3.5 Sprung Befehle

3.8.3.5

-> JUMP CALL RET
-> J(N)E J(N)C JPL JMI J(N)Z

J(N)E




Bedingter Sprung zum Label xxxx, wenn gleich (ungleich).
Die Sprungbedingung wird durch eine vorausgegangene Operation geschaffen, die den Condition Code beeinflußt hat.

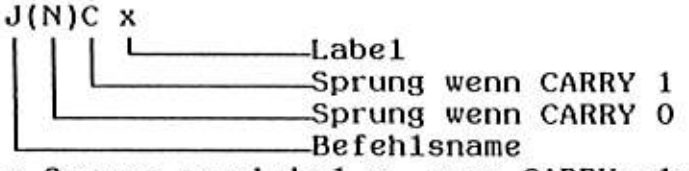
Beispiel:

```
Zähler:             ;Label, Zähler mit Flag bei Oberlauf
     ld 0           ;Akku zu 0 setzen
     st a           ;Variable a zu 0 setzen
loop:               ;Zählschleife
     add 1          ;Addiere 1 zum Inhalt des Akkus
     cmp 255        ;Wert 255 schon erreicht ?
     je flag        ;Sprung zum Oberlaufzeiger wenn ja
     jne loop       ;Sprung zur Schleife (loop) wenn nicht
flag:               ;Oberlaufzeiger
halt                ;stop Programm
```

ACHTUNG:

> Die Klammern stehen nur zur Andeutung der anderen Möglichkeit.
> Entweder heißt dieser Befehl: JE (Sprung wenn gleich)
> oder: JNE (Sprung wenn ungleich)

J(N)C




Bedingter Sprung zum Label x, wenn CARRY gleich (ungleich) 1 ist.
Die Sprungbedingung wird durch eine vorausgegangene Operation geschaffen, die das CARRY Bit beeinflußt hat.

Beispiel:

```
Lauflicht:          ;Label, Lauflicht, 16 Bit Name
     clc            ;lösche das Carry Bit
     ld 1           ;1. Stelle des Lauflichtes laden
loop:               ;label, loop
     shl 1          :schiebe d.Inhalt d.Akkus 1 Stelle nach links
     jc Lauflicht   ;Neuer Anfang, wenn Carry = 1 (Oberlauf)
     jnc loop       ;Sprung zur Schiebe-Schleife
```

-> JUMP CALL RET
-> J(N)E J(N)C JPL JMI J(N)Z

JPL

JPL x
- x → Label
- JPL → Befehlsname

Bedingter Sprung zum Label x, wenn das Ergebnis einer direkt vorausgegangenen Vergleichs- oder Rechenoperation plus/positiv war.

JMI

JMI x
- x → Label
- JMI → Befehlsname

Bedingter Sprung zum Label x, wenn das Ergebnis einer direkt vorausgegangenen Vergleichs- oder Rechenoperation minus/negativ war.

J(N)Z

J(N)Z xxxx
- xxxx → Label
- J(N)Z → Sprung wenn CC = 0
- J(N)Z → Sprung wenn CC = 1
- J(N)Z → Befehlsname

Bedingter Sprung zum Label xxxx, wenn das Ergebnis einer direkt vorausgegangenen Vergleichs- oder Rechenoperation Null war.

ACHTUNG:

> Die Klammern stehen nur zur Andeutung der anderen Möglichkeit.
> Entweder heißt dieser Befehl: JZ (Sprung wenn gleich 0)
> oder: JNZ (Sprung wenn ungleich 0)

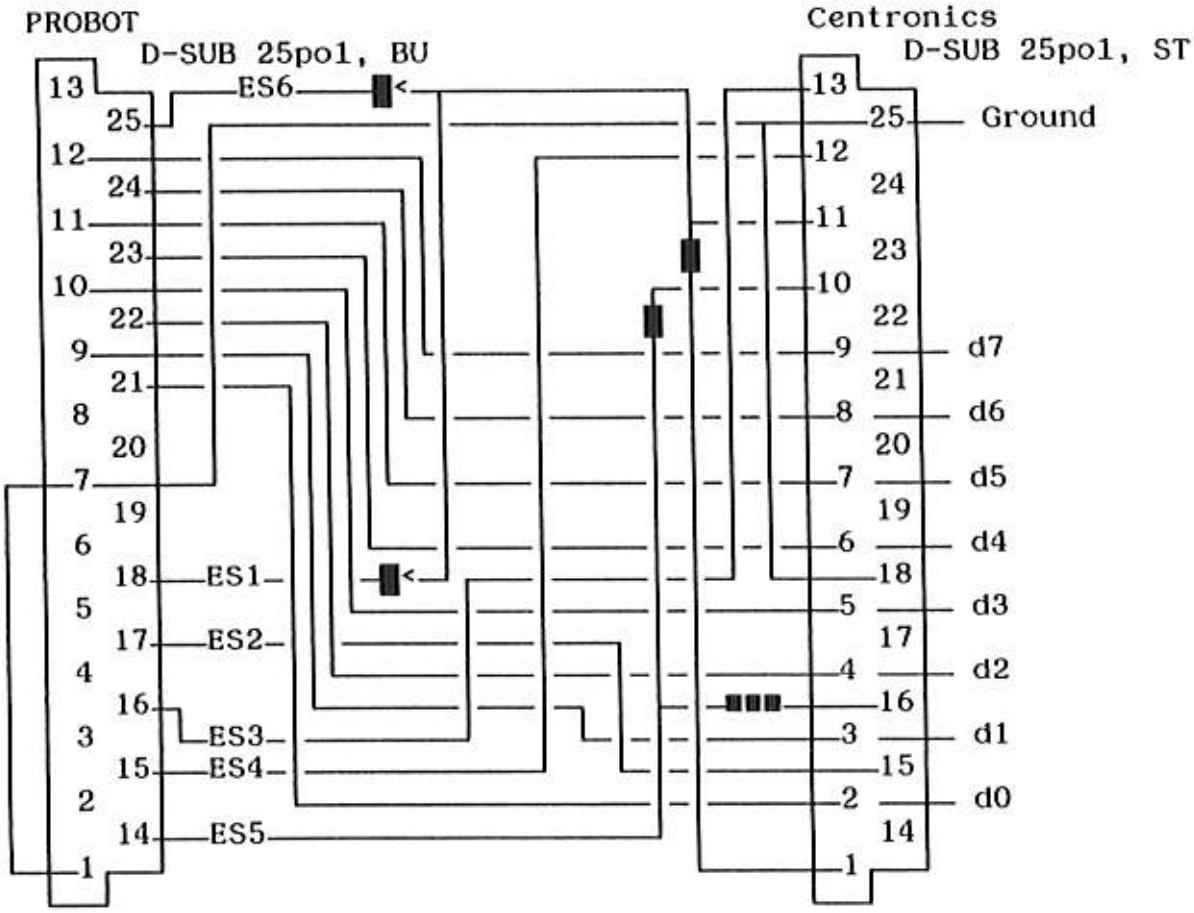
Beispiel:

```
Vergleich:          ;Label ,Name
     ld 10          ;Lade Akku mit 10 (dez)
     st a           ;lade Akku in die Variable a
     ld 0           ;lade Akku mit 0
     cmp a          ;vergleiche den Wert von A mit dem Inhalt des
                    ;Akkus
     jz yes         ;springe zum Label yes wenn gleich
     jnz no         ;springe zum Label no wenn ungleich
yes:                ;Label yes
     halt           ;Programm-Unterbrechung
no:                 ;Label no
     halt           ;Programm-Unterbrechung
```

HINWEIS:
Ist ein Label falsch angegeben oder fehlt es, so wird das Programm gestoppt und eine Fehlermeldung ausgegeben.
Überprüfen Sie also, bevor Sie Ihr Programm sichern, nochmals mit einem freien Durchlauf im Editor (mit den Cursortasten hoch und runter) ob eine Fehlermeldung übersehen wurde.

3.9 Für Ihre Notizen 3.9

3.9.5 Anschluß von PROBOT (mit Schrittmotoren) an das Centronics Druckerport 3.9.5




ES1-6 sind Endschalter
—▮<— D1 und D2
3 Widerstände je 1KOhm

HINWEIS:

Mögliche Portbelegungen der Centronics-Schnittstelle sind:
primary printer: 8378 (Druckeradr. = 0378, aber für PROBOT 8378)
secondary printer: 8278
Bei monochrome graphics/printer: 83BC

ACHTUNG:
Verbindung herstellen, bevor der PC eingeschaltet wird!

Die Verbindung zwischen dem PC und PROBOT ist eine Spezial-Leitung. Sie gehört zum Lieferumfang.

ACHTUNG:

Bitte verwenden Sie ausschließlich diese, gekennzeichnete Leitung!
Eine falsche Leitung kann Zerstörungen verursachen!

LEISTUNGSENDSTUFEN

Die Leistungsendstufen zur Ansteuerung der Schrittmotoren sind im Sockel-Gehäuse des PROBOT untergebracht.

STROMVERSORGUNG

Die Stromversorgung ist ebenfalls im Sockel-Gehäuse des PROBOT untergebracht.

REFERENZMARKEN

PROBOT ist mit sog. Referenzmarken versehen, d.h. optische (magn.) Sensoren erfassen eine bestimmte mechanische Stellung und nehmen diese dann als Referenz. (Nullposition)

INBETRIEBNAHME

PROBOT mit der Spezialleitung an das Druckerport anschließen.
PC einschalten.
Nachdem der PC gebootet hat, Netz für PROBOT einschalten.
Programme laden, u.s.w.

STEUERUNG ANDERER ROBOTER DURCH DIE ROBOT-SPS

Sie ROBOT-SPS ist, auf Grund ihrer Struktur und Vielseitigkeit, in der Lage, auch andere Roboter oder Handling Einheiten zu steuern.

Wenn Sie dazu Fragen oder Probleme haben, dann wenden Sie sich bitte an unseren Programmierdienst.

FAX 07223/20471

Wir helfen Ihnen weiter.

Alphabetische Reihenfolge

Alphabetische Reihenfolge (Fortsetzung)

Alphabetische Reihenfolge

Alphabetische Reihenfolge