

# **Schrittmotorsteuerung**

## **ProStep4**

**PC-Einsteckkarte ISA-Bus / PC104**

**Version 1.2 Mai 1997**

Herzel Steuerungstechnik GmbH  
Thumer Straße 15  
09439 Schloßchen/ Amtsberg  
Tel./Fax.: 03725/23496  
E-Mail: herzel@schrittmotor.de  
<http://www.schrittmotor.de>

DN0011-DE  
Version 2.0

# ProStep4

## 1. Einleitung

Die Schrittmotorsteuerung "ProStep" mit der Softwareversion 1.2 ermöglicht das gleichzeitige Positionieren von maximal vier Schrittmotoren. Sie liefert die Steuersignale für die Schrittmotorleistungstreiber und die übrige Motorperipherie.

Dieses Steuerungssystem gibt es in zwei verschiedenen Ausführungen, um in unterschiedlichen Applikationen benutzbar zu sein:

- └─ PC-ISA-Bus Einsteckkarte
- └─ PC104 Modul

Die Bedienung/ Programmierung der Schrittmotorsteuerungen ist identisch. Sie unterscheiden sich nur im Formfaktor. Sämtliche Ein- und Ausgänge der Schrittmotorsteuerung sind galvanisch getrennt.

Die PC-Einsteckkarte wird in einem freien 8 Bit ISA Slot des PCs platziert und liefert die Steuersignale über einen 37-poligen SUB-D-Stecker nach außen.

Diese Karte ist besonders für den Betrieb in "sauberer" Umgebung (Labor- und Versuchsaufbauten, Prüfstände, Medizintechnik) geeignet, wo sich in relativ geringer Entfernung zum PC die Treiberelektronik und Motoren befinden.

Das PC104 Modul ist für die direkte Integration in die Zielhardware vorgesehen, wo es in einen IPC mit mit PC104 Interface nur aufgesteckt werden braucht. An den Seiten des Motorcontrollers werden die Steuersignale über ansteckbare Flachbandkabel den Endstufen und der Peripherie zugeführt.

Im folgenden werden "ProStep4" der PC-Karte und das PC104 Modul genauer beschrieben:

## ProStep4

Die PC-Karte und das PC104 Modul unterscheiden sich nur im Formfaktor, die Anzahl und die Qualität der Ein- u. Ausgänge sowie die Anwendungssoftware sind identisch.

Auf der Karte befinden sich 8 binäre Ausgänge sowie 16 binäre Eingänge. Die Ein- und Ausgänge sind alle galvanisch von der CPU getrennt. Für die vier Achsen existieren je 2 Steuersignale (Takt u. Richtung), die ebenfalls galvanisch von der CPU, und damit auch vom PC getrennt sind.

Da die Positionierbewegungen unabhängig vom PC von einer modulinternen CPU berechnet werden, ist die PC-Karte/PC104 Modul "ProStep4" auch unter nicht echtzeitfähigen Betriebssystemen wie Windows sinnvoll einsetzbar.

Für ein exaktes Timing der PC-Anwendung, unabhängig vom PC, ist ein schreib/lesbarer 32 Bit Timer auf der PC-Karte/PC104 Modul implementiert.

Die PC-Karte und das PC104 Modul sind als 8 Bit Versionen des jeweiligen Busses ausgeführt und benötigen vier freie 8 Bit Portadressen eines PCs.

Der Betrieb mehrerer PC-Karten/PC104 Module in einem PC ist möglich.

Theoretisch sind dadurch mit Hilfe eines PCs maximal 32 Achsen bedienbar.

Die Einstellung der Portadresse erfolgt über den an der Rückseite der PC-Karte/PC104 Moduls befindlichen BCD-Schalter (S1).

BCD-Schalter	Portadresse im PC
0	300 H
1	304 H
2	308 H
3	30C H
4	310 H
5	314 H
6	318 H
7	31C H

## ProStep4

### 2. Dual-Port-RAM (DPR)

#### 2.1 DPR Überblick

Zur Kommunikation der PC-Karte/PC104 Moduls mit dem PC wird ein virtueller Dual-Port-RAM (DPR) verwendet. Alle Bedienfunktionen laufen über diesen DPR.

Folgende Adressen des DPR sind in der Version 1.2 vom Mai 1997 implementiert:

<b>DualPortRAM</b>			
<b>Register- bezeichnung</b>	<b>Adresse</b>	<b>Größe</b>	<b>Beschreibung</b>
timer	\$0010	long	32 Bit Zähler
input	\$0020	word	binäre Inputs
output	\$0040	word	binäre Outputs
acmr	\$0k00	word	Achs Kommando Register
acr0	\$0k02	byte	Achs Control Register 0
acr1	\$0k03	byte	Achs Control Register 1
asr	\$0k04	word	Achs Status Register
vref1	\$0k1C	long	Referenzfahrgeschwindigkeit 1
vref2	\$0k20	long	Referenzfahrgeschwindigkeit 2
sref	\$0k24	long	max. Referenzfahrweg
v0	\$0k28	long	Startgeschwindigkeit
ve	\$0k2C	long	Stoppgeschwindigkeit
a0	\$0k30	word	Startbeschleunigung
ae	\$0k32	word	Bremsbeschleunigung
vm	\$0k38	long	max. Geschwindigkeit
ik	\$0k40	long	Istkoordinate
iv	\$0k44	long	Istgeschwindigkeit
error	\$0k4C	long	AchsError
sk	\$0k50	long	Sollkoordinate

wobei k die Nummer der Achse(1...4) ist.

## ProStep4

### 2.1 Physikalische Einheiten/ Wertebereiche

Die Parameter müssen in folgenden Einheiten eingegeben werden:

Wege:	Takte
Geschwindigkeiten:	Takte/Sekunde
Beschleunigungen:	Takte/Sekunde <sup>2</sup>

Das System erkennt alle Kommandos nach maximal 10 ms und führt sie aus.

Kann ein Kommando nicht ausgeführt werden oder treten Fehler bei Bewegungen auf, so wird ein Fehler codiert im Register "AchsError" (error) achsspezifisch ausgegeben. Siehe Abschnitt 2.9

Die Position jeder Achse hat den vollen Wertebereich einer 32 Bit vorzeichenbehafteten Zahl (in Pascal "LongInt").

Geschwindigkeiten werden immer als 32 Bit und die Beschleunigungen als 16 Bit Variablen eingegeben, wobei aber keine negativen Werte zugelassen sind (das Vorzeichen wird ignoriert!).

Die Register **vref1** und **vref2** bilden dabei eine Ausnahme, ihr Vorzeichen bestimmt die Bewegungsrichtung. Der Wertebereich ist je nach Register unterschiedlich.

Alle Werte die größer sind als der maximal zulässige Wert, werden auf dieses Maximum reduziert.

Alle Status- und Kommandoregister sind 8 oder 16 Bit vorzeichenlose Register. Deren Werte sind in der folgenden Beschreibung, aus Gründen der besseren Lesbarkeit, immer hexadezimal angegeben.

## ProStep4

### 2.2 Timer

Auf der PC-Karte/PC104 Modul "ProStep4" ist ein 10 ms Timer implementiert.

**timer** [long]

**\$0010**

Der 32 Bit Zähler wird alle 10 ms um 10 inkrementiert. Der Timer kann vom PC aus beschrieben und gelesen werden.

Bsp: Timer nullen  
PS4WriteLong(\$0010,0);

Timer lesen  
Timer1 := PS4ReadLong(\$0010);

## ProStep4

### 2.3 Eingänge

#### 2.3.1 Allgemein

Der PC kann alle auf der PC-Karte/PC104 Modul "ProStep4" befindlichen binären Eingänge im DPR wortweise lesen.

**input** [word] **\$0020**

Bsp: Eingänge lesen  
Inputs := PS4ReadWord(\$0020);

Bedeutung der Bits in der Variable "Inputs" :

	Pin am Sub-D-Stecker der PC-Karte (P1)	PC104 Modul (P4)
Bit 0: Input 0 (Ref1)	9	11
Bit 1: Input 1 (E+1)	27	12
Bit 2: Input 2 (E-1)	8	13
Bit 3: Input 3	26	14
Bit 4: Input 4 (Ref2)	7	15
Bit 5: Input 5 (E+2)	25	16
Bit 6: Input 6 (E-2)	6	17
Bit 7: Input 7	24	18
Bit 8: Input 8 (Ref3)	5	19
Bit 9: Input 9 (Ref3)	23	20
Bit 10: Input 10 (E+3)	4	21
Bit 11: Input 11 (E-3)	22	22
Bit 12: Input 12	3	23
Bit 13: Input 13 (Ref4)	21	24
Bit 14: Input 14 (E+4)	2	25
Bit 15: Input 15 (E-4)	20	26

Die Refreshzeit der Eingänge beträgt 10 ms.

## ProStep4

### 2.3.2 Referenz- und Endlagenschalter

Die PC-Karte/PC104 Modul ist in der Lage eine automatische Referenzpunktfahrt auszuführen, die Endpunkte des Bewegungsbereiches zu überwachen und die Bewegung abubrechen, wenn der zulässige Bereich überschritten wird.

Dafür muß der Referenzpunktschalter und die Endschalter an die Eingänge Ref, E+ und E- angeschlossen und mit Hilfe des Registers **acr0** aktiviert werden.

Das Register **acr1** bestimmt die Polarität (Öffner / Schließer) der jeweiligen Schalter.

#### **acr0**

Bit 0	1 - enable Ref;	0 - disable Ref
Bit 1	1 - enable E+;	0 - disable E+
Bit 2	1 - enable E-;	0 - disable E-
Bit 3 ... 7	frei	

#### **acr1**

Bit 0	1 - Ref Öffner ;	0 - Ref Schließer
Bit 1	1 - E+ Öffner;	0 - E+ Schließer
Bit 2	1 - E- Öffner ;	0 - E- Schließer
Bit 3 ... 7	frei	

Sind die Inputs disabled so können sie vom PC aus als normale Eingänge verwendet werden.

## ProStep4

### 2.4 Ausgänge

Die auf der PC-Karte/PC104 Modul befindlichen binären Ausgänge können jederzeit mit Hilfe des DPR (wortweise) verändert werden.

**output** [word] **\$0040**

Bsp: Ausgänge beschreiben

```
WriteWord($0040, Outputs);
```

Bedeutung der Bits in der Variable "Outputs":

		Pin am Sub-D-Stecker der PC-Karte (P1)	PC104 Modul (P4)
Bit 0:	Output 0	13	3
Bit 1:	Output 1	31	4
Bit 2:	Output 2	12	5
Bit 3:	Output 3	30	6
Bit 4:	Output 4	11	7
Bit 5:	Output 5	29	8
Bit 6:	Output 6	10	9
Bit 7:	Output 7	28	10
Bit 8 ... 15	frei		

Die Refreshzeit der Ausgänge beträgt 10 ms.

## ProStep4

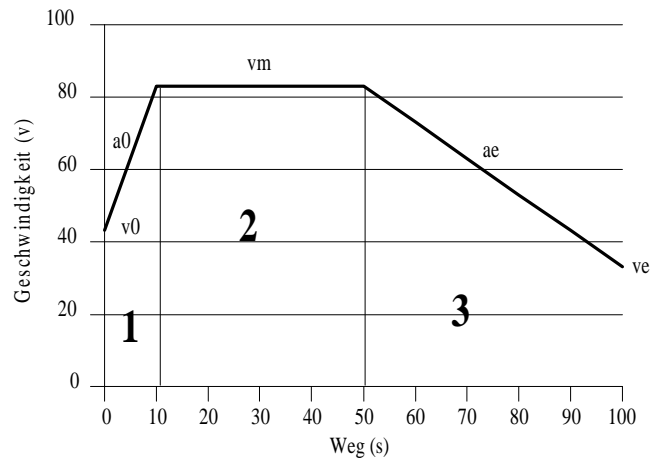
### 2.5 Rampenregister

Für die Parametrierung der Rampen stehen je fünf Register pro Achse zur Verfügung.  
Nach Reset sind alle diese Register Null.

└	Startgeschwindigkeit (v0) [long] Geschwindigkeit mit der die Achse beim Start beginnt Wertebereich: 0... 45000 Takte/s	<b>\$0k28</b>
└	Startbeschleunigung (a0) [word] Steilheit der Startrampe Wertebereich: 0... 32767 Takte/s <sup>2</sup>	<b>\$0k30</b>
└	Maximalgeschwindigkeit (vm) [long] maximale Geschwindigkeit auf welche beschleunigt werden soll Wertebereich: 0 ... 45000 Takte/s	<b>\$0k38</b>
└	Stoppgeschwindigkeit (ve) [long] Geschwindigkeit beim Stoppen der Achse Wertebereich: 0... 45000 Takte/s	<b>\$0k2C</b>
└	Bremsbeschleunigung (ae) [word] Steilheit der Rampe beim Bremsen Wertebereich: 0... 32767 Takte/s <sup>2</sup>	<b>\$0k32</b>

## ProStep4

Die folgende Abbildung zeigt ein Beispiel für eine Start/Stopp Rampe mit unterschiedlichen Steilheiten im Start- und Stoppbereich.



- 1 - Beschleunigungsphase
- 2 - Phase konstanter Geschwindigkeit
- 3 - Bremsphase

## ProStep4

### 2.6 Referenzfahrtregister

Für die Referenzpunktfahrt einer Achse stehen mehrere Register zur Verfügung.  
Nach Reset sind alle diese Register Null.

- └ Referenzfahrgeschwindigkeit 1 (vref1) [long] **\$0k1C**  
Geschwindigkeit des ersten Teiles der Referenzpunktfahrt  
das Vorzeichen bestimmt die Richtung der Referenzfahrt 1  
Wertebereich: -45000... 45000
  
- └ Referenzfahrgeschwindigkeit 2 (vref2) [long] **\$0k20**  
Geschwindigkeit des zweiten Teiles der Referenzpunktfahrt  
das Vorzeichen bestimmt die Richtung der Referenzfahrt 2  
Wertebereich: -45000 ... 45000
  
- └ maximaler Referenzfahrweg (sref) [long] **\$0k24**  
maximaler relativer Weg, welcher bei einer Referenzfahrt  
zurück gelegt werden darf  
Wertebereich: 32 Bit
  
- └ Achs Kommando Register (acmr) [word] **\$0k00**  
Kommandoregister zum Starten und Steuern  
der Referenzpunktfahrt

siehe Abschnitte 3.1 und 3.2

## ProStep4

### 2.7 Positionierregister

Für die Positionierbewegungen sind folgende Register vorgesehen:  
Nach Reset sind alle diese Register Null.

└	Istkoordinate(ik) [long] zeigt die aktuelle Position der Achse an Wertebereich: -2G ... 2G	<b>\$0k40</b>
└	Istgeschwindigkeit(iv) [long] zeigt die aktuelle Geschwindigkeit der Achse an Wertebereich: 0 ... 45000	<b>\$0k44</b>
└	Sollkoordinate(sk) [long] Zielkoordinate der folgenden Posi.-bewegung Wertebereich: -2G ... 2G	<b>\$0k50</b>
└	Achs Kommando Register(acmr) [word] startet und stoppt die Bewegungen	<b>\$0k00</b>

## ProStep4

### 2.8 Kommando- und Statusregister

Der DPR verfügt über jeweils ein Status- und ein Kommandoregister je Achse. Nach Reset sind alle diese Register Null.

└ Achs Kommando Register(acmr) [word]	<b>\$0k00</b>
└ Achs Status Register(asr) [word]	<b>\$0k04</b>

#### 2.8.1 Kommandoregister

Mit Hilfe des Kommandoregister starten und stoppen die einzelnen Bewegungen einer Achse. Das Kommandoregister entspricht einer 16 Bit vorzeichenlosen Variablen (in Pascal "word"). Das Setzen eines der nachfolgend beschriebenen Bits auf high, startet die zugehörige Bewegung.

Bedeutung der Bits im Kommandoregister:

Bit 0	Achse stoppen (ohne Rampe)
Bit 1	Achse bremsen (stoppen mit Rampe)
Bit 2	automatische Referenzfahrt starten
Bit 3	Referenzfahrt Teil 1 starten
Bit 4	Referenzfahrt Teil 2 starten
Bit 5	Stopp und Koordinate nullen
Bit 6 ... 8	frei
Bit 9	Posibewegung starten
Bit 10...15	frei

Ein Kommando wird in maximal 10 ms gestartet.

Hat die PC-Karte/PC104 Modul ein Kommando verstanden, löscht die PC-Karte/ PC104 Modul das Kommandoregister vollständig, um das Kommando zu bestätigen.

Sind mehrere Bits im Kommandoregister gesetzt, so wird nur das am höchsten priorisierte Kommandobit ausgeführt.

Bit 0 hat dabei die höchste und Bit 15 die niedrigste Priorität.

## ProStep4

### **Stopp (Bit 0)**

Setzt man das Bit 0 gleich high, bewirkt das einen sofortiges Stopp der Achse.  
Bei hohen Geschwindigkeiten muß dabei mit Schrittfehlern gerechnet werden.

### **Bremsen (Bit1)**

Setzt man das Bit 1 auf high, bewirkt das ein Abbremsen der Achse.

### **Automatische Referenzfahrt (Bit2)**

Setzt man das Bit 2 auf high, startet die automatische Referenzpunktfahrt.  
Siehe Abschnitt 3.2

### **Referenzfahrt Teil1 (Bit3)**

Setzt man das Bit 3 auf high, startet der Teil 1 der Referenzpunktfahrt.  
Siehe Abschnitt 3.1

### **Referenzfahrt Teil2 (Bit4)**

Setzt man das Bit 4 auf high, startet der Teil 2 der Referenzpunktfahrt.  
Siehe Abschnitt 3.1

### **Posi (Bit 9)**

Setzt man das Bit 9 auf high, startet eine Positionierbewegung.  
Siehe Abschnitt 3.3

## ProStep4

### 2.8.2. Statusregister

Das Statusregister gibt die Information über den aktuellen Zustand der Achse.  
Nach Reset ist das Statusregister Null.

**asr** [word]

**\$0k04**

Bedeutung der Bits im Statusregister:

Bit 0	Achse ist im RUN (Achse bewegt sich)
Bit 1	Achse ist synchronisiert
Bit 2	Error
Bit 3	Achse ist im ersten Teil der Referenzfahrt
Bit 4	Achse ist im zweiten Teil der Referenzfahrt
Bit 5 ... 7	frei
Bit 8	RUN in der Beschleunigungsphase
Bit 9	RUN mit konstanter Geschwindigkeit
Bit 10	RUN in der Bremsphase
Bit 11 ... 15	frei

#### **RUN Bit (Bit 0)**

Das RUN Bit ist high wenn die Achse sich bewegt. Steht die Achse wieder so wird es automatisch wieder low.

#### **Synchronisationsbit (Bit 1)**

Das Synchronisationsbit (Bit 1) wird mit Hilfe des Kommandos "Stoppen/Nullen" oder nach der automatischen Referenzpunktfahrt gesetzt.

Das Löschen erfolgt bei "Stop" (ohne Rampe), während der Referenzfahrten oder in Fehlerfällen.

#### **Errorbit (Bit 2)**

Das Errorbit (Bit 2) wird immer in Fehlerfällen high. Den Fehlertyp kann man dann aus dem Register error (\$0k4C) ermitteln.

Das Fehlerbit wird durch jedes neue Kommando wieder gelöscht.

#### **Ref1 (Bit 3)**

Das Bit 3 zeigt an, wenn die PC-Karte/PC104 Modul sich zur Zeit im ersten Teil der Referenzfahrt befindet.

#### **Ref2 (Bit 4)**

Das Bit 4 zeigt an, wenn die PC-Karte/PC104 Modul sich zur Zeit im zweiten Teil der Referenzfahrt befindet.

#### **Beschleunigung (Bit 8)**

Das Bit 8 zeigt an wenn die PC-Karte/PC104 Modul sich zur Zeit in der Beschleunigungsphase einer Positionierbewegung befindet.

#### **Konstante Geschwindigkeit (Bit 9)**

Das Bit 9 zeigt an, wenn die PC-Karte/PC104 Modul sich zur Zeit in der Phase der konstanten Geschwindigkeit einer Positionierbewegung befindet.

#### **Bremsen (Bit 10)**

## ProStep4

Das Bit 10 zeigt an, wenn die PC-Karte/PC104 Modul sich zur Zeit in der Bremsphase einer Positionierbewegung befindet.

### 2.9 Fehlerregister

In dem Register **error** werden bitcodierte Fehlerzustände der Achse abgelegt.

**error** [long] **\$0k4C**

Folgende Bits sind definiert:

Bit 0	maximaler Weg bei Referenzfahrt überschritten
Bit 1	positive Endlage (E+) überfahren
Bit 2	negative Endlage (E-) überfahren
Bit 3 ... 31	frei

Ist das Register **error** gleich Null so liegen keine Fehler vor.

Ein einmal aufgetretener Fehler wird durch das Starten einer neuen Bewegung wieder gelöscht.

## ProStep4

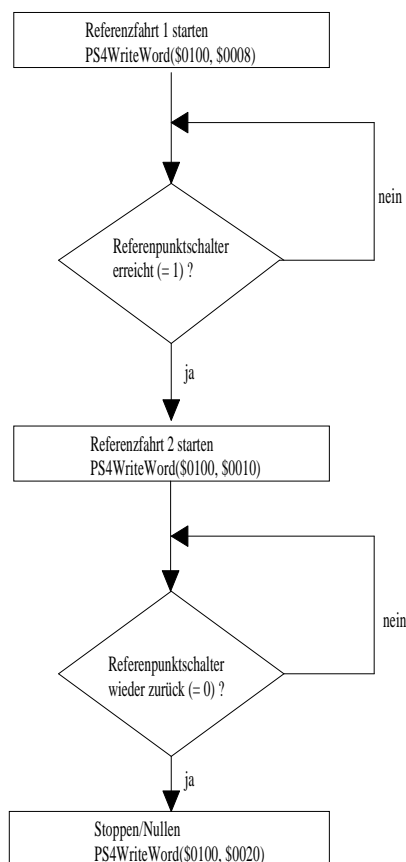
### 3. Bewegungen

#### 3.1. Halbautomatische Referenzfahrt

Eine Referenzfahrt benötigt die Register wie in Abschnitt 2.6 beschrieben.

Die PC-Karte/PC104 Modul ist für eine halbautomatische Referenzfahrt jeweils einer Achse ausgelegt. Sollen allen Achsen Referenz fahren , müssen die Startkommados nacheinander gegeben werden.

Die Abbildung zeigt den Ablauf einer Referenzfahrt wie sie mit dem PC ausgeführt werden sollte:



Jede Referenzfahrt ist in zwei Teile gegliedert:

- Teil 1: Hinfahrt ( Fahrt auf den Referenpunktschalter )
- Teil 2: Rückfahrt (langsame Fahrt vom Referenpunktschalter herunter)

Für die beiden Teile der Referenzfahrt ist die jeweilige Geschwindigkeit und Richtung (Vorzeichen) in den Registern **vref1** und **vref2** einzutragen.

Das Register **sref** (maximaler Referenzfahrweg \$0k24) dient der Sicherheit der Referenzfahrt und sollte nicht

## ProStep4

größer als der maximale Verfahrweg der Anlage gewählt werden.  
Der maximale Referenzfahrweg(sref) ist eine relative Größe.

Bei Überschreitung des Wertes von **sref** wird die Achse gestoppt und das Bit 0 im Register **error** (AchseError \$0k4C) gesetzt.

Als Eingang für den Referenzpunktschalter kann ein beliebiges Eingangsbit der PC-Karte/PC104 Modul verwendet werden; es ist aber nicht zwingend notwendig!

Ein vorzeitiges Beenden der Referenzpunktfahrt kann mit Hilfe eines der Stop-Kommandos(mit oder ohne Rampe, Stoppen/Nullen) erfolgen.

Bei den Referenzfahrten werden prinzipiell keine Rampen gefahren.

Probleme:

Sollte es bei dieser Methode zu nicht reproduzierbaren Referenzpunkten kommen, so muß die Referenzfahrgeschwindigkeit 2 (vref2) reduziert werden.

## ProStep4

### 3.2 Automatische Referenzfahrt

Mit Hilfe der PC-Karte/PC104 Modul "ProStep4" ist auch eine automatische Referenzfahrt möglich. Zu diesem Zweck muß der Referenzschalter (ref) am vorgesehenen Eingang der jeweiligen Achse angeschlossen sein.

Ob ein Öffner oder ein Schließer als Referenzpunktschalter Verwendung findet, ist im Register **acr1** einzustellen.

Der Referenzpunktschalter muß außerdem im Register **acr0** aktiviert werden.  
Siehe auch Abschnitt 2.3.2

Die automatische Referenzpunktfahrt läuft im Grunde genauso wie die halbautomatische Referenzfahrt ab.

Die zu verwendenden Register sind im Abschnitt 2.6 beschrieben.

Das Kommando "automatische Referenzfahrt starten" (Bit 2 im Kommandoregister) startet die automatische Referenzfahrt.

Am Ende der Referenzfahrt ist die jeweilige Achse genullt und synchronisiert.

Ein vorzeitiges Beenden der Referenzpunktfahrt kann mit Hilfe der beiden Stop-Kommandos (mit oder ohne Rampe) erfolgen.

**Beispiel 1** einer Referenzfahrt der ersten Achse:



PS4WriteLong(\$011C, -100);	{ v der Hinfahrt negativ }
PS4WriteLong(\$0120, 10);	{ v der Rückfahrt positiv }
PS4WriteLong(\$0124, 20000)	{ max. Fahrweg }
PS4WriteByte(\$0102, \$07);	{ Ref.- u. Endlagenschalter aktiviert }
PS4WriteByte(\$0103, \$07);	{ alle Schalter sind Schließer }
PS4WriteWord(\$0100, \$0004);	{ Start der automatischen Ref.-fahrt }

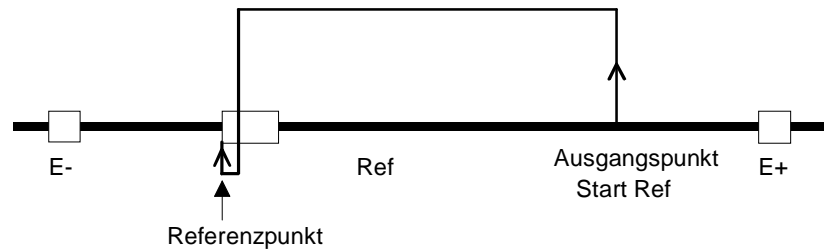
Nach max. 10 ms startet die Referenzfahrt und das Bit 0 im Statusregister (asr) wird high.  
Solange das Bit 0 high ist, läuft die Referenzfahrt. Ist das Bit 0 low, dann ist die Referenzfahrt beendet.

Bei einer Überschreitung des maximal zulässigen Referenzfahrweges stoppt die Achse und das Bit 0 im Fehlerregister (error) wird high. Siehe Abschnitt 2.9

Überfährt die Achse während der Referenzfahrt einen der aktivierten Endlagenschalter, stoppt die Bewegung und ein Fehlerbit im Fehlerregister (error) wird gesetzt.

## ProStep4

**Beispiel 2** einer Referenzfahrt der ersten Achse:



```
PS4WriteLong($011C, -100);      { v der Hinfahrt negativ }
PS4WriteLong($0120, -10);       { v der Rückfahrt negativ }
PS4WriteLong($0124, 20000)      { max. Verfahrensweg }
PS4WriteByte($0102, $07);       { Ref.- u. Endlagenschalter aktiviert }
PS4WriteByte($0103, $07);       { alle Schalter sind Schließer }

PS4WriteWord($0100, $0004);     { Start der automatischen Ref.-fahrt }
```

## ProStep4

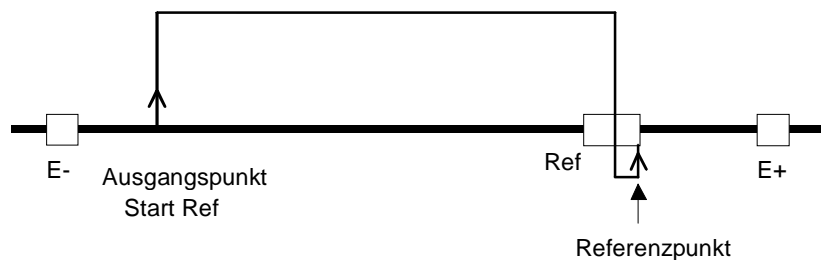
**Beispiel 3** einer Referenzfahrt der ersten Achse:



```
PS4WriteLong($011C, 100); { v der Hinfahrt positiv }
PS4WriteLong($0120, -10); { v der Rückfahrt negativ }
PS4WriteLong($0124, 20000) { max. Verfahrweg }
PS4WriteByte($0102, $07); { Ref.- u. Endlagenschalter aktiviert }
PS4WirteByte($0103, $07); { alle Schalter sind Schließer }

PS4WriteWord($0100, $0004); { Start der automatischen Ref.-fahrt }
```

**Beispiel 4** einer Referenzfahrt der ersten Achse:



```
PS4WriteLong($011C, 100); { v der Hinfahrt positiv }
PS4WriteLong($0120, 10); { v der Rückfahrt positiv }
PS4WriteLong($0124, 20000) { max. Verfahrweg }
PS4WriteByte($0102, $07); { Ref.- u. Endlagenschalter aktiviert }
PS4WirteByte($0103, $07); { alle Schalter sind Schließer }

PS4WriteWord($0100, $0004); { Start der automatischen Ref.-fahrt }
```

## ProStep4

### 3.3 Positionierbewegungen

Ist die Achse synchronisiert (nach der Referenzpunktfahrt oder Kommando "Stop/ Nullen"), können Positionierbewegungen ausgeführt werden.

Das Bit 9 des Kommandoregisters (acmr) startet eine Positionierbewegung:

```
PS4WriteWord($0100, $0200);
```

Beim Start wird das Kommandoregister wieder gelöscht. Im Statusregister sind die entsprechenden Bits max. 10ms später gesetzt.

Beispiel einer Posibewegung der zweiten Achse:

Zuerst werden die Rampenregister geladen. Dies ist nur einmal zur Systeminitialisierung, siehe Abschnitt 6. erforderlich.

```
PS4WriteLong($0228, 200); { v0 }
PS4WriteLong($022C, 200); { ve }
PS4WriteWord($0230, 1000) { a0 }
PS4WriteWord($0232, 1000); { ae }
PS4WriteLong($0238, 1500); { vm }
PS4WriteLong($0250, 2000); { sk (Sollkoordinate) }

PS4WriteByte($0202, $07); { Ref.- u. Endlagenschalter aktiviert }
PS4WirteByte($0203, $07); { alle Schalter sind Schließer }
```

Die eigentliche Positionierbewegung wird mit folgendem Befehl gestartet:

```
PS4WriteWord($0200, $0200); { Start der Posbewegung }
```

Die Bewegung ist beendet, wenn das Bit 1(RUN) im Statusregister wieder gelöscht ist.

Im obigen Beispiel sind beide Endlagenschalter aktiviert.

Fährt die Achse in diesem Fall über einen der Endlagenschalter, so wird die Achse sofort gestoppt und das Bit 1 oder das Bit 2 im Errorregister (\$024C) sind gesetzt.

Die anderen Achsen bleiben davon unbeeinflusst.

## ProStep4

### 4. DPR - Funktionen

Zum Beschreiben und Lesen der Register im DualPortRAM (DPR) vom PC aus, können die im folgenden Abschnitt beschriebenen Funktionen verwendet werden.

Es ist dabei zu beachten, daß die einzelnen Register eine unterschiedliche Größe aufweisen. Für den jeweiligen Typ (byte, word , long) müssen die zugehörigen Funktionen verwendet werden. Geschieht dies nicht, so kann es zu Fehlern in der Bewegungsausführung kommen.

Die Funktionen sind als Quellcode in "PASCAL" und "C" bzw. als DLL für Windows 3.1 und Windows 95 auf der beiliegenden Diskette enthalten.

Die Funktionen sind bewußt nicht auf "High Speed" optimiert, sondern auf Funktionsgliederung und gute Lesbarkeit.

Sie können natürlich die Programmausführgeschwindigkeit noch steigern oder die Funktionen in andere Programmiersprachen portieren.

Im Abschnitt 5 werden die Portzugriffe zu diesem Zweck detaillierter beschrieben.

## ProStep4

### 4.1 PC-Karte/PC104 Modul initialisieren

Die Funktion PS4Init öffnet einen Kommunikationskanal zur PC-Karte/PC104 Modul "ProStep4" und testet das Vorhandensein dieser Karte auf der angegebenen Portadresse ab.

#### Pascal:

```
function PS4Init(Adr: word; Baud: integer; TimeOut: longint): integer;
```

Parameter:	Adr	PortAdresse der PC-Karte/PC104 Modul "ProStep4" Siehe Seite 3
	Baud	keine Bedeutung
	TimeOut	TimeOut

Return:	0 - OK
	1 - PC-Karte "ProStep4" nicht gefunden

#### C:

```
int PS4Init(unsigned short int Adr, int Baud, long TimeOut)
```

#### DLL:

```
function PS4Init(Adr: word; Baud: integer; TimeOut: longint): integer;
```

TimeOut entspricht der Zyklenanzahl einer Warteschleife.  
Siehe Quellcode "prostep.pas" in der function PS\_Wait.

Bei den DLLs für Windows 3.1 und Windows 95 entspricht TimeOut einer Zeit in ms.

## ProStep4

### 4.2 Verbindung schließen

Die Funktion PS4Close schließt den durch PS4Init geöffneten Kommunikationskanal wieder. In der vorliegenden Version der Software hat diese Funktionen keine Bedeutung, sie ist nur aus Kompatibilitätsgründen zu Windows NT und zu Steuerungen via serieller Schnittstelle vorhanden.

#### Pascal:

```
procedure PS4Close;
```

#### C:

```
void PS4Close(void)
```

#### DLL:

```
procedure PS4Close;
```

## ProStep4

### 4.3 DatenByte schreiben

Die Funktion PS4WriteByte schreibt ein Datenbyte auf eine bestimmte Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
procedure PS4WriteByte(Adr : Word; Daten : byte);
```

Parameter:           Adr     - 16 Bit Adresse des DPR von ProStep4  
                  Daten    - zu sendendes Datenbyte (8Bit)

#### C:

```
void PS4WriteByte(unsigned short int Adr, unsigned char Daten)
```

#### DLL:

```
procedure PS4WriteByte(Adr: Word; Daten: byte);
```

## ProStep4

### 4.4 Datenword schreiben

Die Funktion PS4WriteWord schreibt ein Datenword eine bestimmte Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
procedure PS4WriteWord(Adr: Word; Daten : byte);
```

Parameter:           Adr     - 16 Bit Adresse des DPR von ProStep4  
                  Daten    - zu sendendes Datenword (16 Bit)

#### C:

```
void PS4WriteWord(unsigned short int Adr, unsigned short int Daten)
```

#### DLL:

```
procedure PS4WriteWord(Adr: Word; Daten: word);
```

**Achtung!** Die Adresse muß bei Wordzugriffen immer geradzahlig sein!

## ProStep4

### 4.5 Datenlong schreiben

Die Funktion PS4WriteLong schreibt ein Datenlong auf eine bestimmte Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
procedure PS4WriteLong(Adr : Word; Daten :long);
```

Parameter:           Adr     - 16 Bit Adresse des DPR von ProStep4  
                  Daten    - zu sendendes Datenlong (32 Bit)

#### C:

```
void PS4WriteLong(unsigned short int Adr, long Daten)
```

#### DLL:

```
procedure PS4WriteLong(Adr: Word; Daten: byte);
```

**Achtung!** Die Adresse muß bei Longzugriffen immer geradzahlig sein!

## ProStep4

### 4.6 Datenbyte lesen

Die Funktion PS4ReadByte liest ein Datenbyte von einer bestimmten Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
function PS4ReadByte(Adr: Word): byte;
```

Parameter:       Adr     - 16 Bit Adresse des DPR von ProStep4

Return:           Datenbyte

#### C:

```
unsigned char PS4ReadByte(unsigned short int Adr)
```

#### DLL:

```
function PS4ReadByte(Adr: Word): byte;
```

## ProStep4

### 4.7 Datenword lesen

Die Funktion PS4ReadWord liest ein Datenword von einer bestimmten Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
function PS4ReadWord(Adr: Word): Word;
```

Parameter:       Adr     - 16 Bit Adresse des DPR von ProStep4

Return:           Datenword

#### C:

```
unsigned short int PS4ReadWord(unsigned short int Adr)
```

#### DLL:

```
function PS4ReadWord(Adr: Word): Word;
```

**Achtung!** Die Adresse muß bei Wordzugriffen immer geradzahlig sein!

## ProStep4

### 4.8 Datenlong lesen

Die Funktion PS4ReadLong liest ein Datenlong von einer bestimmten Adresse im DualPortRAM(DPR) der PC-Karte/PC104 Modul "ProStep4".

#### Pascal:

```
function PS4ReadLong(Adr: Word): long;
```

Parameter:       Adr     - 16 Bit Adresse des DPR von ProStep4

Return:           Datenlong

#### C:

```
long PS4ReadLong(unsigned short int Adr)
```

#### DLL:

```
function PS4ReadLong(Adr: Word): long
```

**Achtung!** Die Adresse muß bei Longzugriffen immer geradzahlig sein!

## ProStep4

### 4.9 Stop/Reset

Die Funktion PS4Reset stoppt alle Achsen und führt auf der PC-Karte/PC104 Modul "ProStep4" ein Reset aus. Die binären Outputs werden gelöscht und alle Register im DPR sind wieder Null. Die Funktion entspricht einem NotAUS.

#### Pascal:

```
procedure PS4Reset;
```

#### C:

```
void PS4Reset(void)
```

#### DLL:

```
procedure PS4Reset;
```

#### Achtung!

Nach dem Reset muß mindestens 2 s gewartet werden, weil das System der PC-Karte neu hochgefahren wird.

## ProStep4

### 4.10 Stop

Die Funktion PS4Stop hat die gleiche Funktionalität wie die Funktion Stop/Reset.

#### Pascal:

```
procedure PS4Stop;
```

#### C:

```
void PS4Stop(void)
```

#### DLL:

```
procedure PS4Stop;
```

## ProStep4

### 4.11 Fehler lesen

Die Funktion PS4GetError liest den aktuellen Fehler der Kommunikation.  
Ist das Ergebnis ungleich 0, dann liegt ein Fehler vor.

#### Pascal:

```
function PS4GetError: integer;
```

Return: FehlerCode

0	-	alles O.K.
1	-	TimeOutFehler
2	-	"ProStep" nicht O.K.
3	-	PC-Karte "ProStep" nicht gefunden

#### C:

```
int PS4GetError(void)
```

#### DLL:

```
function PS4GetError: integer;
```

## ProStep4

### 5. Inbetriebnahme

Im folgenden wird ein Inbetriebnahmebeispiel in Pascal beschrieben:

Es sei dabei S1 auf 0 (300H) eingestellt.

Nur die Achse1 soll bewegt werden.

```
var Fehler: integer;
begin
  Fehler := PS4Init($300,0,10000);      { Portadresse 300H, TimeOut 10000 }      if Fehler = 0 then

  begin
    PS4WriteByte($0102 ,3);           { Ref, E+, E- aktiviert }
    PS4WriteByte($0103, 3);           { alles Schließer }
    PS4WriteLong($011C, -100);        { neg. Ref1. Geschwindigkeit }
    PS4WriteLong($0120, 10);          { pos. Ref2. Geschwindigkeit }
    PS4WriteLong($0124, 10000);       { max. Referenzfahrweg = 10000 }

    PS4WriteLong($0128, 100);         { v0 = 100 }
    PS4WriteLong($012C, 100);         { ve = 100 }
    PS4WriteWord($0130, 100);         { a0 = 1000 }
    PS4WriteWord($0132, 100);         { ae = 1000 }
    PS4WriteLong($0138, 800);         { vm = 800 }

  end;
end;

procedure POSI(SollPos: longint)
begin
  PS4WriteLong($0150, SollPos);       { SollPosition eintragen }
  PS4WriteWord($0100, $200);          { Start Posibewegung }
end;
```

Aus Sicherheitsgründen kann nach jedem Zugriff auf die PC-Karte/PC104 Modul mit Hilfe der Funktion "PS4GetError" der Fehlercode gelesen werden. Ist der Fehlercode ungleich Null so liegt ein Fehler in der Kommunikation zwischen PC-Karte/PC104 Modul vor.

## ProStep4

Zur Inbetriebnahme liefern wir ein kleines Testprogramm für Windows 3.x "PS4DEMO.EXE" mit. Sie finden es auf der beiliegenden Diskette im Verzeichnis "\DEMO".

Das Programm "PS4DEMO.EXE" erfordert keinerlei Installation. Sie können es von Diskette aus starten, oder einfach auf die Festplatte kopieren.

	Achse 1	Achse 2	Achse 3	Achse 4
v0 [1/s]	100	100	100	100
ve [1/s]	100	100	100	100
a0 [1/s <sup>2</sup> ]	1000	1000	1000	1000
ae [1/s <sup>2</sup> ]	1000	1000	1000	1000
vm [1/s]	1000	800	1000	1000
Xsoll	200	25000	300	5000
Error	0000 HEX	0000 HEX	0000 HEX	0000 HEX
Status	0000 HEX	0201 HEX	0000 HEX	0201 HEX
Ist. v [1/s]	0	800	0	1000
Position	200	22587	300	3990
	Stop	Stop	Stop	Stop
	Bremsen	Bremsen	Bremsen	Bremsen
	Start Ref	Start Ref	Start Ref	Start Ref
	Start	Start	Start	Start

Fehler: keine                      Inputs: 0000 HEX                      Timer: 71630 ms

## ProStep4

### 6. Zubehör

- └ 37 poliges Kabel mit Sub-D-Buchsen  
zur Verbindung der PC-Karte und des Klemmbrettes  
0,5 m lang
- └ Flachbandkabelsatz zur Verbindung PC104 Modul und  
Klemmbrett  
0,3 m lang
- └ Klemmbrett für PC-Karte für alle Ein- bzw Ausgänge  
5 mm Schraubklemmen  
integrierte Spannungsversorgung(5V 300 mA) bei 24 V Einspeisung  
Ausgänge über Schmitt-Trigger verstärkt  
je Ein- und Ausgang ein Kontroll LED  
Maße: 160 x 100 mm
- └ Klemmbrett für PC104 Modul für alle Ein- bzw Ausgänge  
5 mm Schraubklemmen  
integrierte Spannungsversorgung(5V 300 mA) bei 24 V Einspeisung  
Ausgänge über Schmitt-Trigger verstärkt  
je Ein- und Ausgang ein Kontroll LED  
Maße: 160 x 100 mm
- └ MiniStep  
Endstufe für Hutschiene  
max. 1,5 A Phasenstrom (mit Lüfter) 37 V
- └ MediStep  
Endstufe für Hutschiene  
max. 2,5 A Phasenstrom 55 V
- └ weitere Endstufen bis 90 V für Hutschiene  
und 19 Zoll Technik

## ProStep4

### 7. Technische Daten

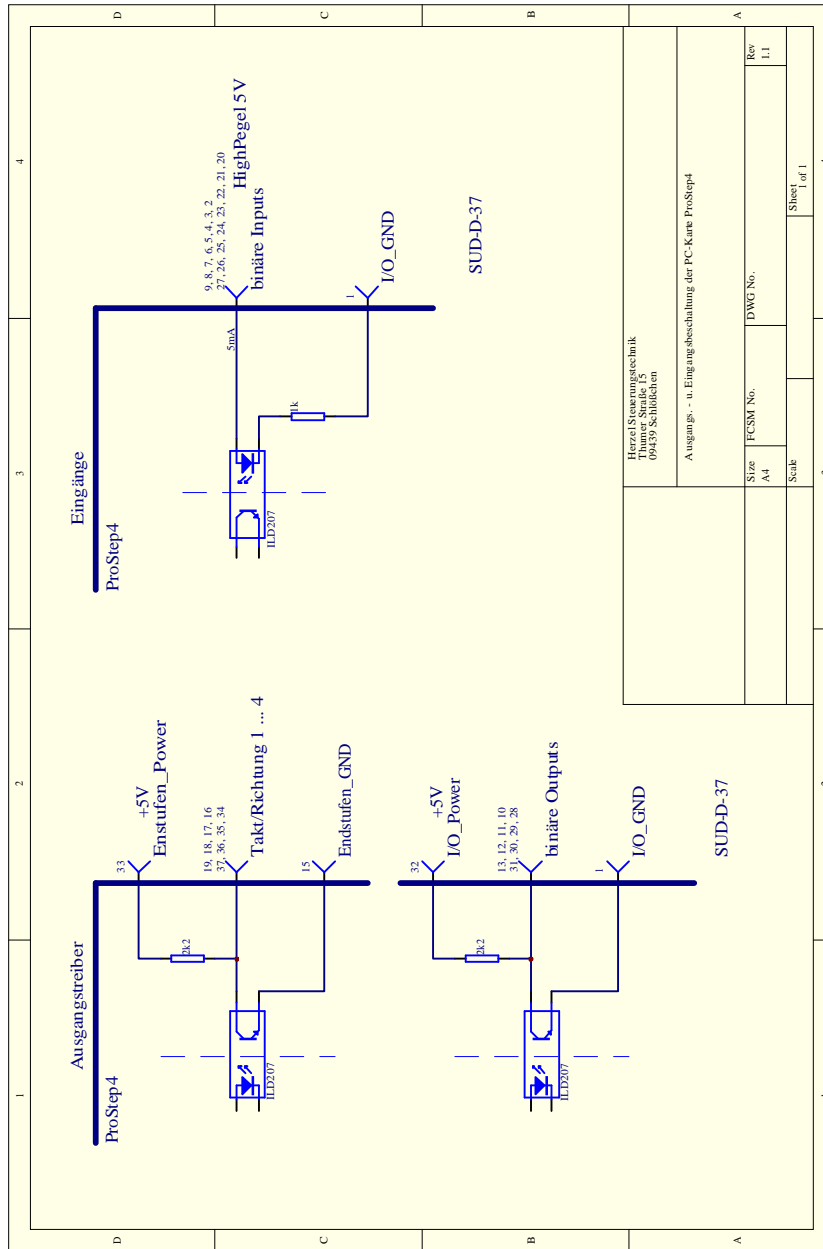
Typ	asynchrone vier Achsen Positionssteuerung je Achse Takt.- und Richtungsausgang
max. Taktfrequenz	45 kHz je Achse
BusType	ISA 8-Bit / PC104 8Bit
Ports	vier 8 Bit Ports
Adressen	300H ... 31CH einstellbar
Stromversorgung	5V/ 300 mA vom ISA Bus/ PC104 Bus
ext. Versorgung der Optokoppler	5V
Steckverbinder	
PC-Karte	37 pol. Sub-D-Stecker
PC104 Modul	20 + 26 pol. IDC - Stecker
Eingänge:	16 binäre Eingänge galvanisch vom PC getrennt
high	5mA
Ausgänge:	8 binäre Ausgänge 8 Ausgänge für Takt und Richtung galvanisch vom PC getrennt Open Collector + Pullup Widerstand(2k2)
Datenaustausch zum PC	Dual - Port - RAM
Wertebereich	31 Bit +/-
Rampen	linear
Referenzpunktfahrt	integriert
Einheiten	
Weg	Takte
Geschwindigkeit	Takte/s
Beschleunigung	Takte/s <sup>2</sup>

technische Änderungen vorbehalten

# ProStep4

## 8. Schaltpläne

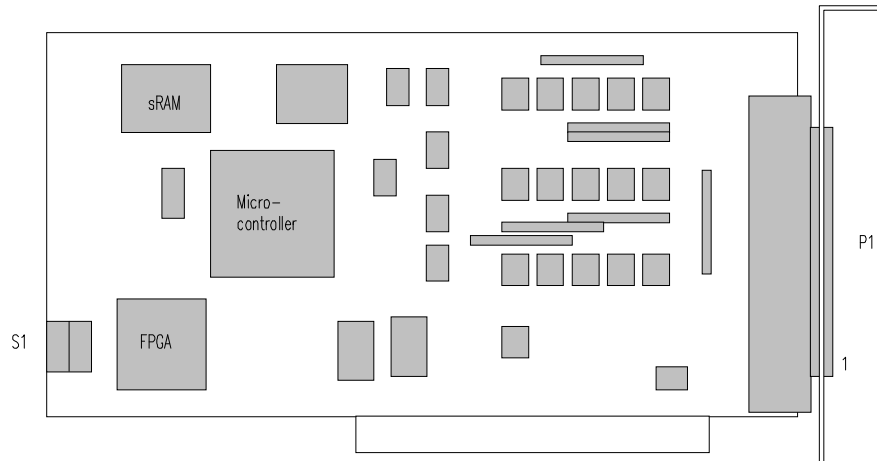
### 8.1 Ein- und Ausgangsbeschaltung der PC-Karte "ProStep4"





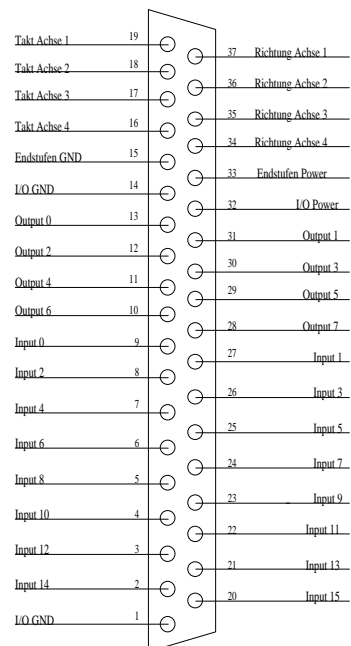
# ProStep4

## 8.3 ISA PC-Karte Hardware



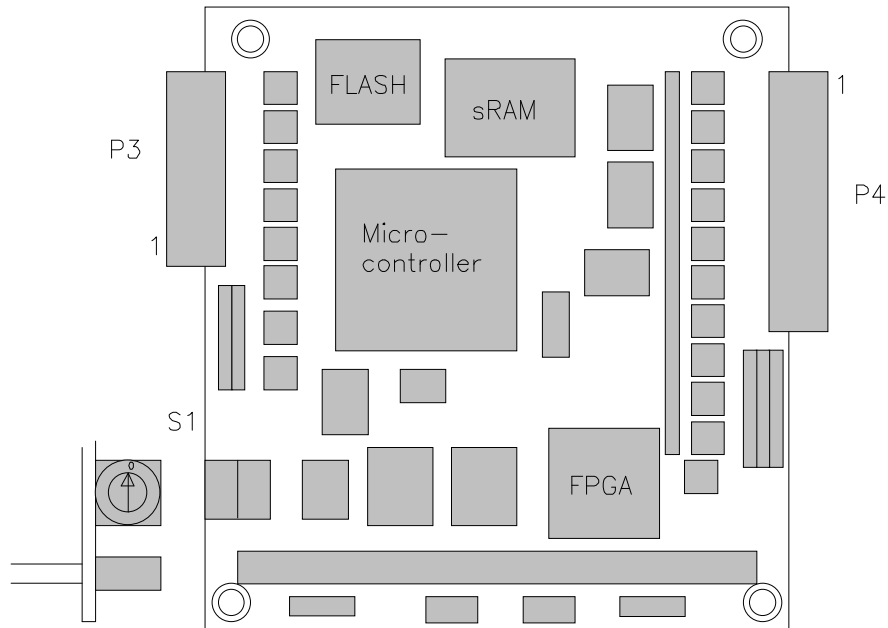
P1 Sub-D 37

S1	Portadresse
0	300H
1	304H
2	308H
3	30CH
4	310H
5	314H
6	318H
7	31CH



# ProStep4

## 8.4 PC104 Modul Hardware



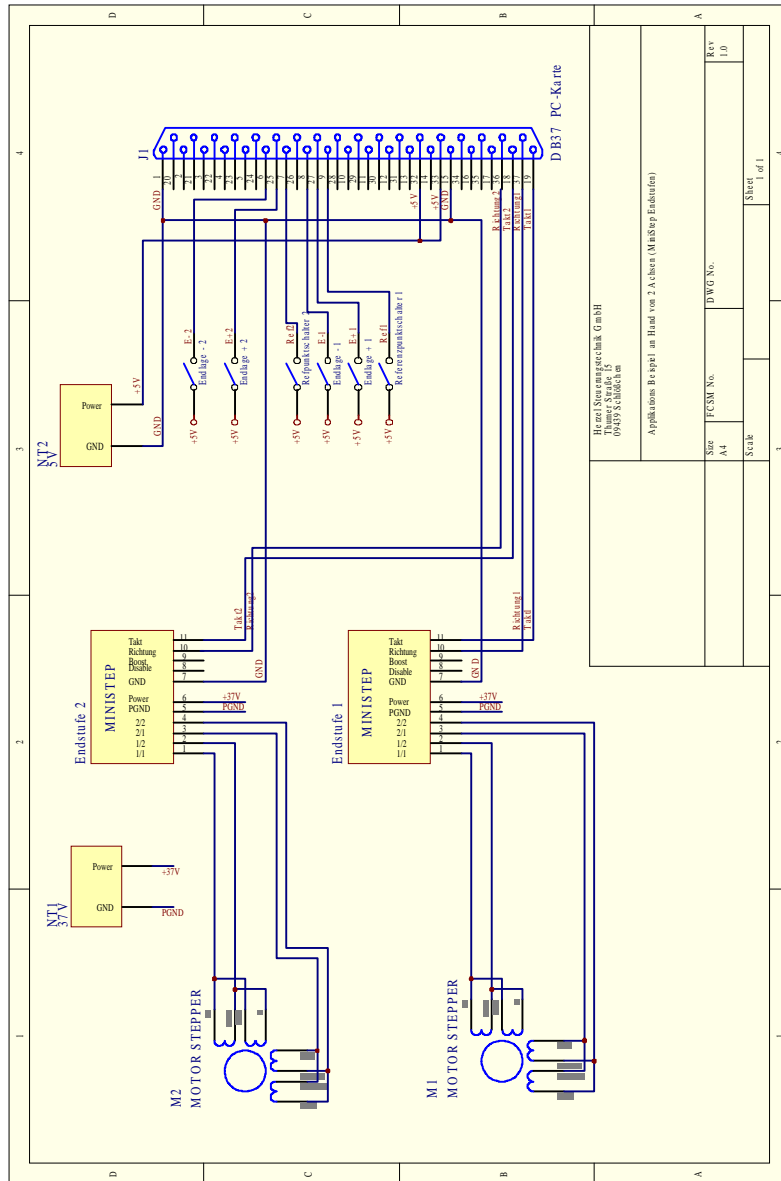
P3		P4	
1.. 10	frei	1	I/O Power +5V
11	Takt Achse 1	2	I/O GND
12	Richtung Achse 1	3	Output 0
13	Takt Achse 2	4	Output 1
14	Richtung Achse 2	5	Output 2
15	Takt Achse 3	6	Output 3
16	Richtung Achse 3	7	Output 4
17	Takt Achse 4	8	Output 5
18	Richtung Achse 4	9	Output 6
19	Endstufen GND	10	Output 7
20	Endstufen Power +5V	11	Input 0 (Ref Achse 1)
		12	Input 1 (E+ Achse 1)
		13	Input 2 (E- Achse 1)
		14	Input 3
		15	Input 4 (Ref Achse 2)
		16	Input 5 (E+ Achse 2)
		17	Input 6 (E- Achse 2)
		18	Input 7
		19	Input 8 (Ref Achse 3)
		20	Input 9 (E+ Achse 3)
		21	Input 10 (E- Achse 3)
		22	Input 11
		23	Input 12 (Ref Achse 4)
		24	Input 13 (E+ Achse 4)
		25	Input 14 (E- Achse 4)
		26	Input 15

S1	Portadresse
0	300H
1	304H
2	308H
3	30CH
4	310H
5	314H
6	318H
7	31CH

# ProStep4

## 8.5 Schaltungsbeispiel einer 2 Achssteuerung



# ProStep4

## 9. Inhalt

1. Einleitung	2
2. Dual-Port-RAM (DPR)	4
2.1 DPR Überblick	4
2.1 Physikalische Einheiten/ Wertebereiche	5
2.2 Timer	6
2.3 Eingänge	7
2.4 Ausgänge	9
2.5 Rampenregister	10
2.6 Referenzfahrtregister	12
2.7 Positionierregister	13
2.8 Kommando- und Statusregister	14
2.9 Fehlerregister	17
3. Bewegungen	18
3.1. Halbautomatische Referenzfahrt	18
3.2 Automatische Referenzfahrt	20
3.3 Positionierbewegungen	23
4. DPR - Funktionen	24
4.1 PC-Karte/PC104 Modul initialisieren	25
4.2 Verbindung schließen	26
4.3 DatenByte schreiben	27
4.4 Datenword schreiben	28
4.5 Datenlong schreiben	29
4.6 Datenbyte lesen	30
4.7 Datenword lesen	31
4.8 Datenlong lesen	32
4.9 Stop/Reset	33
4.10 Stop	34
4.11 Fehler lesen	35
5. Inbetriebnahme	36
6. Zubehör	38
7. Technische Daten	39
8. Schaltpläne	40
8.1 Ein- und Ausgangsbeschaltung der PC-Karte "ProStep4"	40
8.2 Ein- und Ausgangsbeschreibung des PC104 Moduls	41
8.3 ISA PC-Karte Hardware	42
8.4 PC104 Modul Hardware	43
8.5 Schaltungsbeispiel einer 2 Achssteuerung	44
9. Inhalt	45