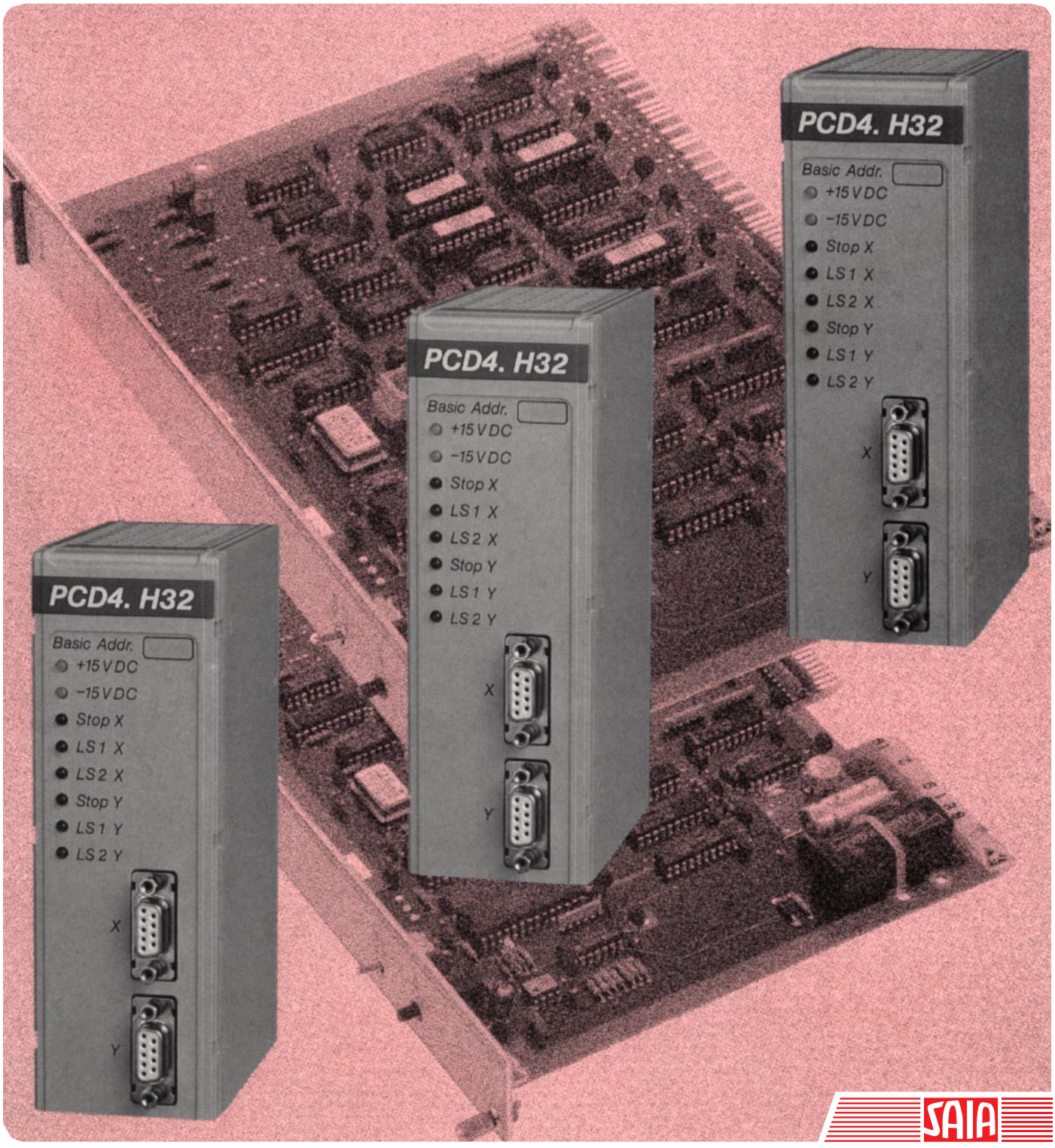
SAIA-Burgess Electronics

SWITCHES • MOTORS • CONTROLLERS

# SAIA®PCD

## Process Control Devices

# Handbuch Positioniermodule für Servoantriebe PCD4.H3..




**Ausgabe 26/729 D2**

SAIA-Burgess Electronics AG
Bahnhofstrasse 18
CH-3280 Murten (Schweiz)
http;//www.saia-burgess.com

GB: Electronic Controllers Telefon 026 / 672 72 72
Telefax 026 / 672 74 99

## SAIA-Burgess Gesellschaften

**Schweiz**
SAIA-Burgess Electronics AG
Freiburgstrasse 33
CH-3280 Murten
☎ 026 672 77 77, Fax 026 670 19 83

**Deutschland**
SAIA-Burgess Electronics GmbH
Daimlerstrasse 1k
D-63303 Dreieich
☎ 06103 89 060, Fax 06103 89 06 66

**Österreich**
SAIA-Burgess Electronics Ges.m.b.H.
Schallmooser Hauptstrasse 38
A-5020 Salzburg
☎ 0662 88 49 10, Fax 0662 88 49 10 11

**Italien**
SAIA-Burgess Electronics S.r.l.
Via Cadamosto 3
I-20094 Corsico MI
☎ 02 48 69 21, Fax 02 48 60 06 92

**Frankreich**
SAIA-Burgess Electronics Sàrl.
10, Bld. Louise Michel
F-92230 Gennevilliers
☎ 01 46 88 07 70, Fax 01 46 88 07 99

**Niederlande**
SAIA-Burgess Electronics B.V.
Hanzeweg 12c
NL-2803 MC Gouda
☎ 0182 54 31 54, Fax 0182 54 31 51

**Belgien**
SAIA-Burgess Electronics Belgium
Avenue Roi Albert 1er, 50
B-1780 Wemmel
☎ 02 456 06 20, Fax 02 460 50 44

**Ungarn**
SAIA-Burgess Electronics Automation Kft.
Liget utca 1.
H-2040 Budaörs
☎ 23 501 170, Fax 23 501 180

## Vertretungen

**Gross-britannien**
Canham Controls Ltd.
25 Fenlake Business Centre, Fengate
Peterborough PE1 5BQ UK
☎ 01733 89 44 89, Fax 01733 89 44 88

**Dänemark**
Malthe Winje Automation AS
Håndværkerbyen 57 B
DK-2670 Greve
☎ 70 20 52 01, Fax 70 20 52 02

**Norwegen**
Malthe Winje Automasjon AS
Haukelivn 48
N-1415 Oppegård
☎ 66 99 61 00, Fax 66 99 61 01

**Schweden**
Malthe Winje Automation AB
Truckvägen 14A
S-194 52 Upplands Våsby
☎ 08 795 59 10, Fax 08 795 59 20

**Suomi/ Finnland**
ENERGEL OY
Atomitie 1
FIN-00370 Helsinki
☎ 09 586 2066, Fax 09 586 2046

**Portugal**
INFOCONTROL Electronica e Automatismo LDA.
Praceta Cesário Verde, No 10 s/cv, Massamá
P-2745 Queluz
☎ 21 430 08 24, Fax 21 430 08 04

**Spanien**
Tecnosistemas Medioambientales, S.L.
Poligono Industrial El Cabril, 9
E-28864 Ajalvir, Madrid
☎ 91 884 47 93, Fax 91 884 40 72

**Tschechische Republik**
ICS Industrie Control Service, s.r.o.
Modranská 43
CZ-14700 Praha 4
☎ 2 44 06 22 79, Fax 2 44 46 08 57

**Polen**
SABUR Ltd.
ul. Druzynowa 3A
PL-02-590 Warszawa
☎ 22 844 63 70, Fax 22 844 75 20

**Australien**
Siemens Building Technologies Pty. Ltd.
Landis & Staefa Division
411 Ferntree Gully Road
AUS-Mount Waverley, 3149 Victoria
☎ 3 9544 2322, Fax 3 9543 8106

**Argentinien**
MURTEN S.r.l.
Av. del Libertador 184, 4° “A”
RA-1001 Buenos Aires
☎ 054 11 4312 0172, Fax 054 11 4312 0172

## Kundendienst

**USA**
SAIA-Burgess Electronics Inc.
1335 Barclay Boulevard
Buffalo Grove, IL 60089, USA
☎ 847 215 96 00, Fax 847 215 96 06

Issue : 22.11.2000

**SAIA® Process Control Devices**

# Positioniermodule für Servoantriebe

## PCD4.H3xx


Ausgabe 26/729 D2 - 09.1990

# Anpassungen

**Handbuch: PCD4.H3xx - Positioniermodule für Servoantriebe - Ausgabe D2**

| Datum | Abschnitt | Seite | Beschreibung |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Inhaltsverzeichnis

Notizen:

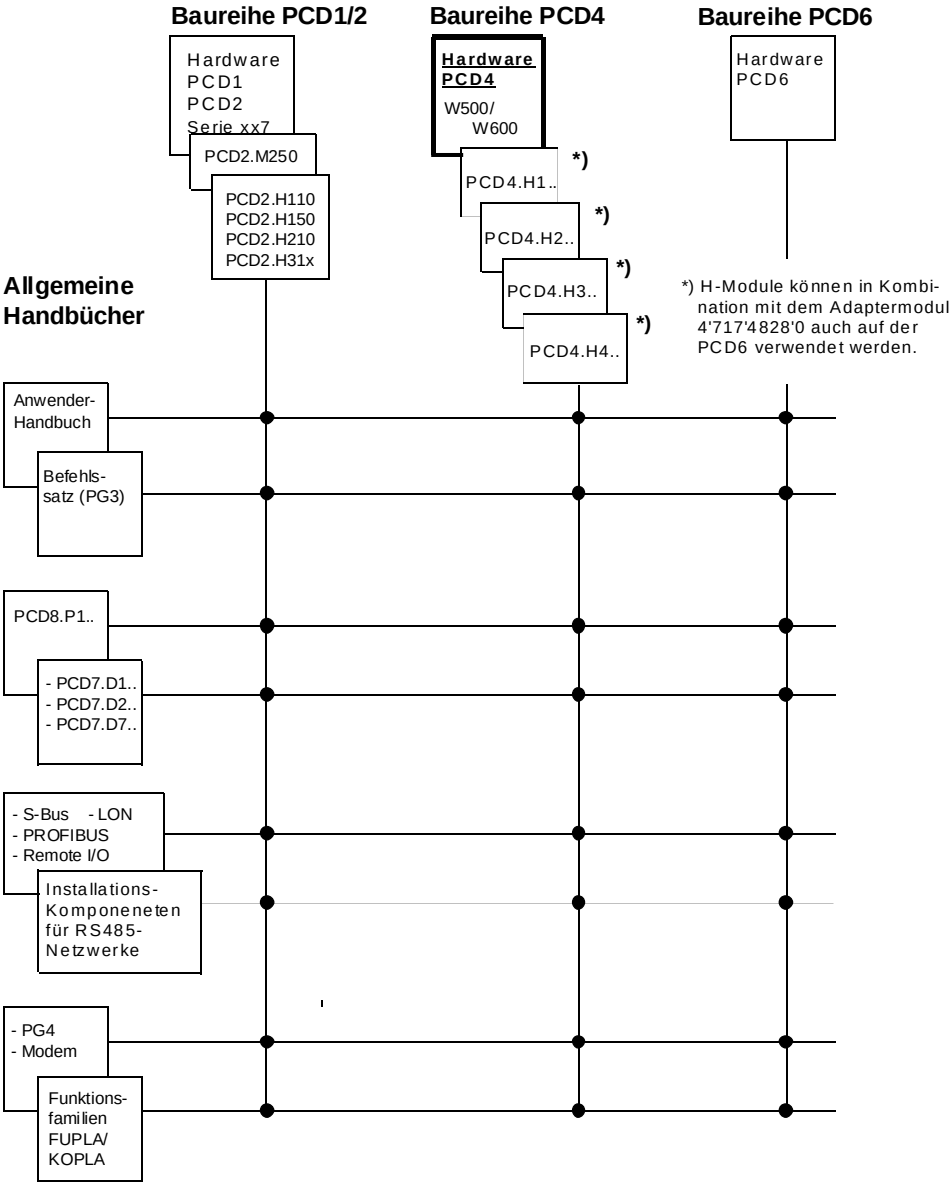
**Wichtiger Hinweis:**

Um den einwandfreien Betrieb von SAIA® PCD sicherstellen zu können, wurde eine Vielzahl detaillierter Handbücher geschaffen. Diese wenden sich an technisch qualifiziertes Personal, das nach Möglichkeit auch unsere Workshops erfolgreich absolviert hat.

Die vielfältigen Leistungen der SAIA® PCD treten nur dann optimal in Erscheinung, wenn alle in diesen Handbüchern aufgeführten Angaben und Richtlinien bezüglich Montage, Verkabelung, Programmierung und Inbetriebnahme genau befolgt werden.

Damit allerdings werden Sie zum grossen Kreis der begeisterten SAIA® PCD Anwendern gehören.

## Übersicht



# Zuverlässigkeit und Sicherheit elektronischer Steuerungen

Die Firma SAIA-Burgess Electronics AG konzipiert, entwickelt und stellt ihre Produkte mit aller Sorgfalt her:

- Neuster Stand der Technik
- Einhaltung der Normen
- Zertifiziert nach ISO 9001
- Internationale Approbationen: z.B. Germanischer Lloyd, United Laboratories (UL), Det Norske Veritas, CE-Zeichen ...
- Auswahl qualitativ hochwertiger Bauelemente
- Kontrollen in verschiedenen Stufen der Fertigung
- In-Circuit-Tests
- Run-in (Wärmelauf bei 85°C während 48h)

Die daraus resultierende hochstehende Qualität zeigt trotz aller Sorgfalt Grenzen. So ist z.B. mit natürlichen Ausfällen von Bauelementen zu rechnen. Für diese gibt die Firma SAIA-Burgess Electronics AG Garantie gemäss den "Allgemeinen Lieferbedingungen".

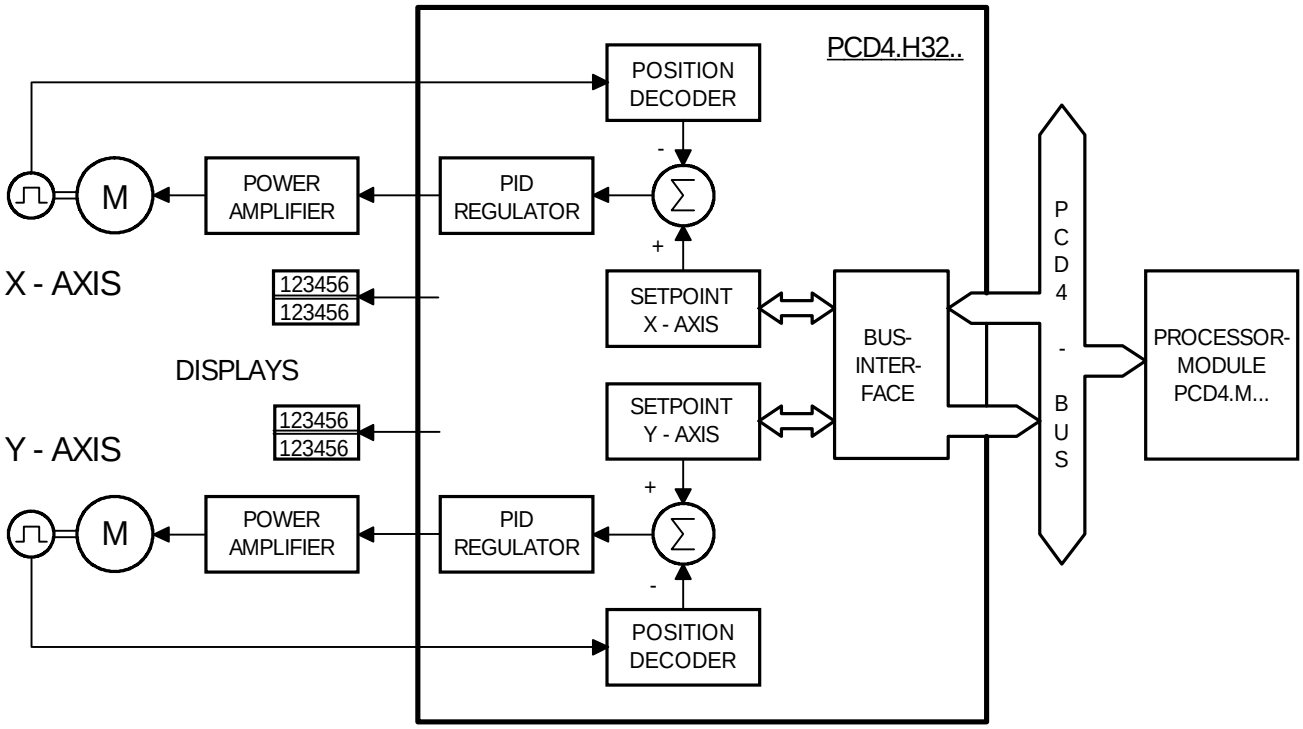
Der Anlagebauer seinerseits muss auch seinen Teil für das zuverlässige Arbeiten einer Anlage beitragen. So ist er dafür verantwortlich, dass die Steuerung datenkonform eingesetzt wird und keine Überbeanspruchungen, z.B. auf Temperaturbereiche, Überspannungen und Störfelder oder mechanischen Beanspruchungen auftreten.

Darüber hinaus ist der Anlagebauer auch dafür verantwortlich, dass ein fehlerhaftes Produkt in keinem Fall zu Verletzungen oder gar zum Tod von Personen bzw. zur Beschädigung oder Zerstörung von Sachen führen kann. Die einschlägigen Sicherheitsvorschriften sind in jedem Fall einzuhalten. Gefährliche Fehler müssen durch zusätzliche Massnahmen erkannt und hinsichtlich ihrer Auswirkung blockiert werden. So sind z.B. für die Sicherheit wichtige Ausgänge auf Eingänge zurückzuführen und softwaremässig zu überwachen. Es sind die Diagnoseelemente der PCD wie Watch-Dog, Ausnahme-Organisations-Blocks (XOB) sowie Test- und Diagnose-Befehle konsequent anzuwenden.

Werden alle diese Punkte berücksichtigt, verfügen Sie mit der SAIA® PCD über eine moderne und sichere programmierbare Steuerung, die Ihre Anlage über viele Jahre zuverlässig steuern, regeln und überwachen wird.

# 1. Einleitung

Blockschema eines Servoantriebes :




**Funktion und Anwendung :**

Das Positioniermodul PCD4.H3.. dient der Positionierung von einer oder zwei unabhängigen Achsen mit drehzahlregelbaren Antrieben (Servomotoren). Solche Servomotoren können regelbare DC- oder AC-Motoren sein, welche über eine Leistungsstufe und einen Inkrementaldrehgeber zur Positions- und Drehzahlerfassung verfügen.

Das Modul kommuniziert über den PCD4-Bus mit der PCD4-CPU. Am Bus werden 16 Adressen belegt pro Modul. D.h. an einem PCD4 System können theoretisch 16 Positioniermodule (32 Achsen) angeschlossen werden.

Auf dem Modul befindet sich für jede Achse ein Singlechip-Prozessor, welcher eine Bewegung, entsprechend den geladenen Parametern (Geschwindigkeit, Beschleunigung und Zielposition), selbständig ausführt. Die Achsen werden unabhängig voneinander gesteuert. D.h. es ist keine Interpolation möglich, um kurvenförmige Bahnen zu fahren. Hingegen ist eine Verkettung mehrer Achsen (Punkt-Punkt) im quasi Synchronbetrieb programmierbar.

**Typische Einsatzgebiete:**

- Handlingsroboter
- Bestückungs- und Montageautomaten
- Palletierungsautomaten
- Verpackungsmaschinen
- Blechbearbeitungsmaschinen

**Programmierung:**

Dem Anwender wird eine Bibliothek mit Software-Funktionsbausteinen zur Verfügung gestellt. Diese Funktionsbausteine bestehen aus PCD-Anwenderprogramm und sind somit für den Anwender transparent. Das Positioniermodul kann so auf einfache Weise programmiert werden, ohne dass noch ein komplizierter Befehlscode gelernt werden muss.

**Wichtigste Eigenschaften:**

- Position und Drehzahl sind PID-geregelt.

- Geschwindigkeit, Zielposition und PID-Parameter können während der Bewegung verändert werden.

- Analoger ± 10V oder Puls-Weiten-Modulierter (PWM) Ausgang zur Ansteuerung der Motorleistungsstufe.

- Digitale Eingänge für Referenz- und Endschalter unter 24V DC (Quellbetrieb)

- Encodersignal-Eingänge für 24V (Quell- oder Senkbetrieb) oder 5V RS422 (Antivalent Line Driver).

- Digitale Ausgänge zum Anschluss von Anzeige-Modulen PCA2.D14 mit 2 x 6 Digit pro Achse.

**Typenübersicht :**

| Bezeichnung | Achsen | Stellgrösse | Encodersignale |
|---|---|---|---|
| PCD4.H310 [1)] | X | ± 10V | 24V |
| PCD4.H320 [1)] | X , Y | ± 10V | 24V |
| PCD4.H311 [1)] | X | ± 10V | 5V (RS422) |
| PCD4.H321 [1)] | X , Y | ± 10V | 5V (RS422) |
| PCD4.H316 [2)] | X | PWM | 24V |
| PCD4.H326 [2)] | X , Y | PWM | 24V |
| PCD4.H317 [2)] | X | PWM | 5V (RS422) |
| PCD4.H327 [2)] | X , Y | PWM | 5V (RS422) |

1) Standard-Sortiment

2) Lieferung auf Anfrage

Notizen:

# 2. Technische Daten

| | |
|---|---|
| **Wegerfassung** | Inkremental :<br>2 um 90° versetzte Impulse und Indexmarke<br>(bzw. deren inverse Signale) |
| 24V-Eingänge | |
| Signalpegel | Low = 0..4V<br>High = 19..32V |
| Eingangsstrom<br>bei 24V | 10mA |
| Betriebsart | Quell- oder Senkbetrieb |
| 5V-Eingänge | Antivalenteingänge 5V nach RS422 |
| Potentialtrennung | Nein |
| Zählfrequenz | Max. 100kHz |
| **Digitale Eingänge** | Pro Achse: 2 Endschalter und<br>1 Referenzschalter an 24VDC |
| Signalpegel | Low = 0..4V<br>High = 19..32V |
| Eingangsstrom<br>bei 24V | 10mA |
| Betriebsart | Quellbetrieb |
| Eingangsfilter | 100kHz |
| **Digitale Ausgänge** | Zur Ansteuerung eines Anzeigemoduls<br>PCA2.D14 (2*6 Digit pro Achse) |
| 2 Ausgänge | Data und Clock<br>(gemeinsam für beide Achsen) |
| 1 Ausgang<br>pro Achse | Enable<br>(umgekehrte Logik: aktiv low) |
| Ausgangsstrom Ia | 1..100mA (nicht kurzschlussfest)<br>Lastwiderstand min. 240 Ohm an 24V DC |

**Reglerausgang** Zur Ansteuerung des Servoverstärkers

Analoger Ausgang ±10V (Auflösung 12 Bit plus Vorzeichen)
Lastwiderstand min. 3kOhm
Kurzschlussfest

PWM Ausgang **P**uls-**W**eiten-**M**oduliert (Auflösung 8 Bit)

Signale : SIGN und MAGNITUDE bzw. angepasst zur direkten Ansteuerung eines Brückenverstärkers

Open Collector Ausgänge : Imax. = 500mA
Umax. = 50V

**Betriebsart** Positionier- oder Drehzahlregelung

**Positioniervorgaben**

Position Einheit : µm
Bereich :
$[ -2^{30} ... + ( 2^{30} - 1 ) ] / k * 10^{-3}$
k = f{Encoderauflösung u. Spindelsteigung}

Geschwindigkeit Einheit : µm/s
Bereich :
$[ -2^{30} ... + ( 2^{30} - 1 ) ] / k * 22348 * 10^{-6}$
k = f{Encoderauflösung u. Spindelsteigung}

Beschleunigung Einheit : $\mu m/s^2$
Bereich :
$[ -2^{30} ... + ( 2^{30} - 1 ) ] / k * 76206 * 10^{-9}$
k = f{Encoderauflösung u. Spindelsteigung}

PID-Regler Abtastzeit : 341µs
Programmierbare Proportional-, Integral- und Differentialfaktoren.
(Abtastzeit für Differentialteil separat programmierbar)

**Programmierung** Mit Funktionsbausteinen, basierend auf PCD-Anwenderprogramm

**Stromversorgung**

| | |
|---|---|
| Extern (Anwender) | + 24V DC ( 19V .. 32V ) geglättet<br>Welligkeit 10% |
| Stromaufnahme der externen 24V-Speisung | |
| für H310, 320, 316 u. 326 | $I_{max}$ = 150mA/Achse + Encoderspeisung |
| für H311, 321, 317 u. 327 | $I_{max}$ = 300mA/Achse |

(Max. Belastung der 5V-Encoderspeisung 300mA/Achse)

| | |
|---|---|
| Intern ab PCD4-Bus | +5V / 120mA pro Achse<br>±15V / 5mA pro Achse (nur bei H310, 311, 320 u. 321) |

**Betriebsbedingungen**

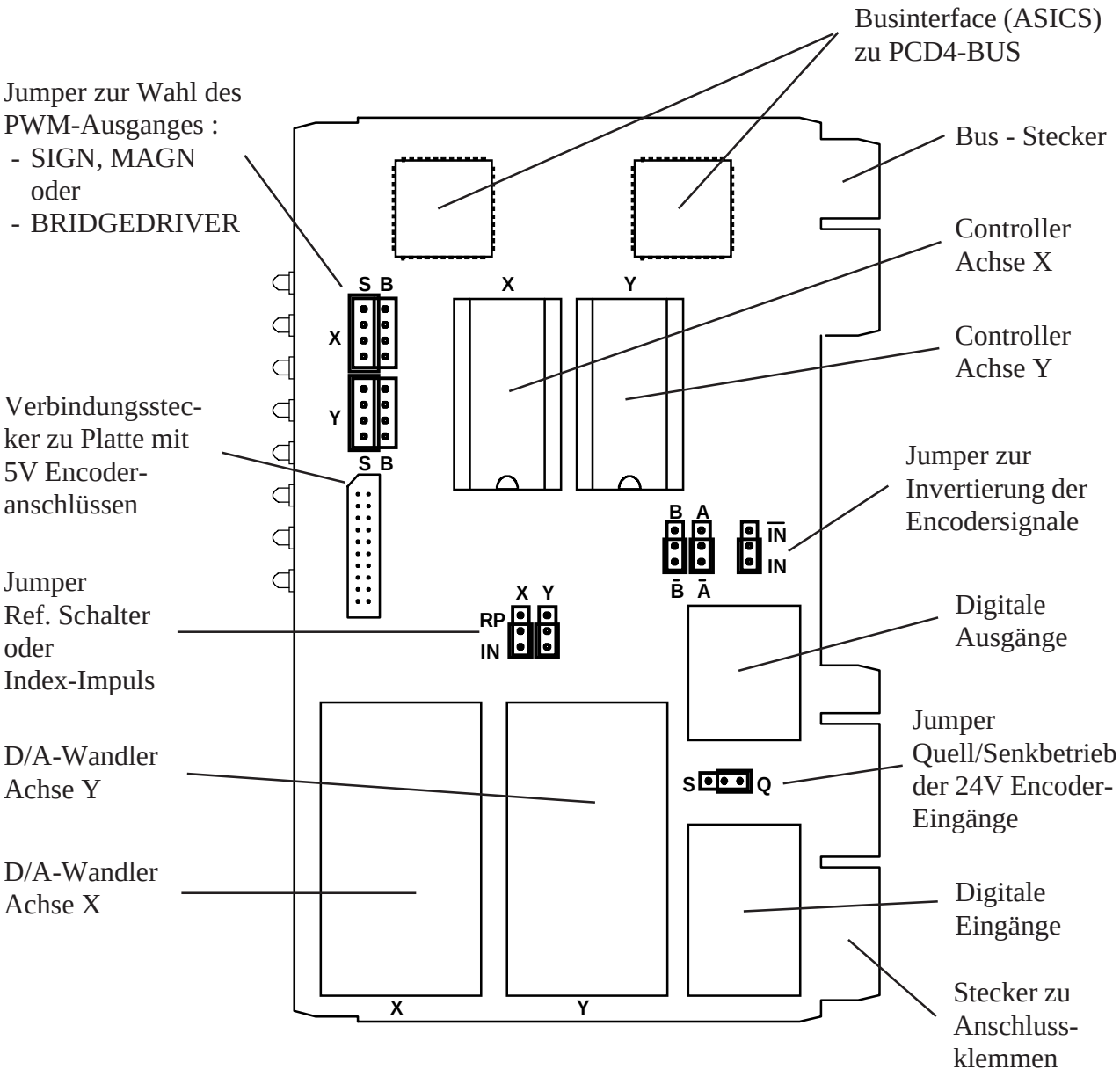
| | |
|---|---|
| Umgebungstemperatur | 0°C .. +50°C ohne Zwangsbelüftung |
| Störfestigkeit | 1kV in kapazitiver Kopplung nach IEC 801-4 |
| Mechanische Festigkeit | Nach IEC 65A |
| Lagerbedingungen | Temperatur: -20°C .. +85°C<br>Luftfeuchtigkeit: 0 .. 95% |

Notizen:

# 3. Präsentation

## 3.1 Leiterplatte

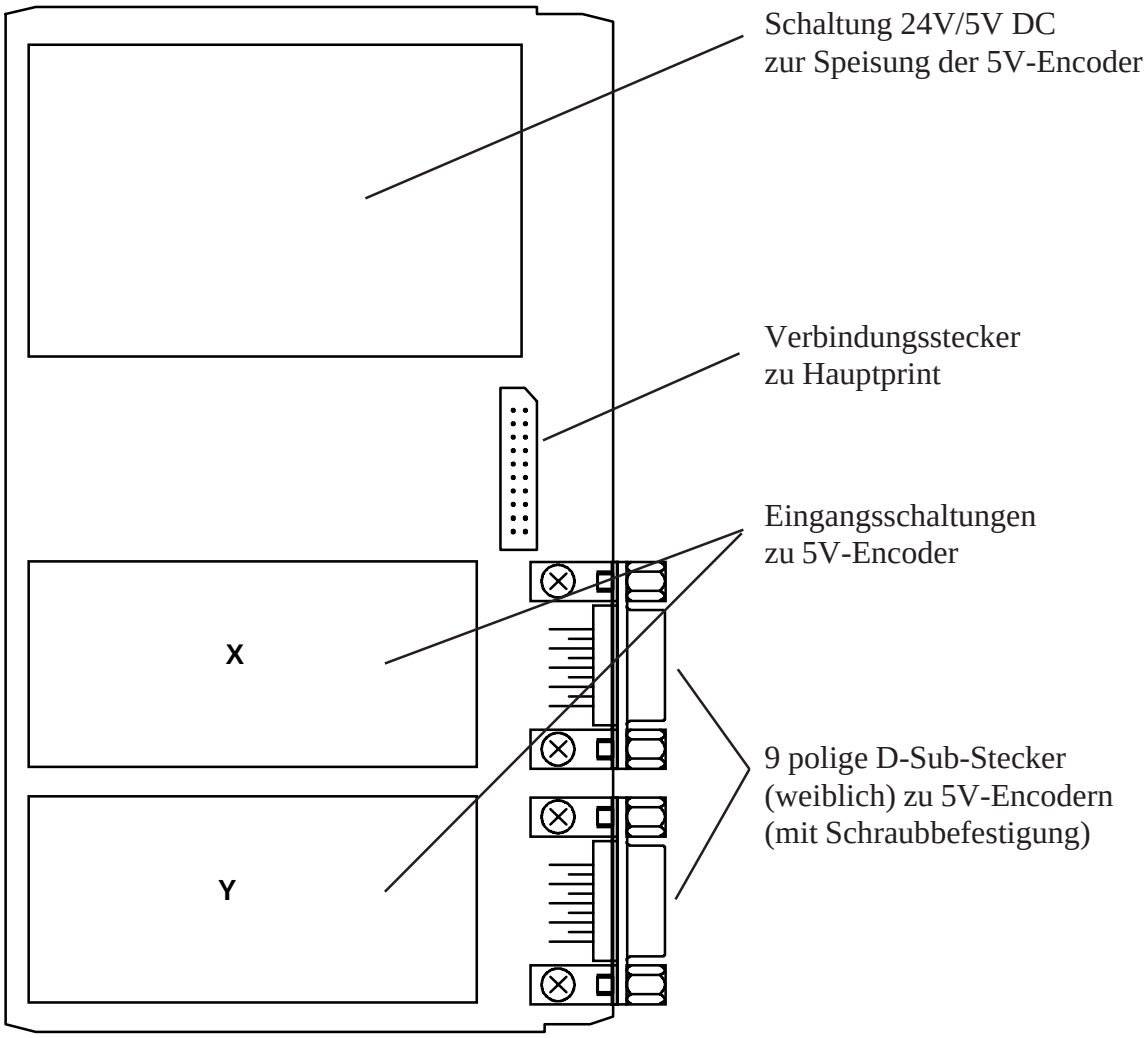
### 3.1.1 Hauptprint (2 Achsen)



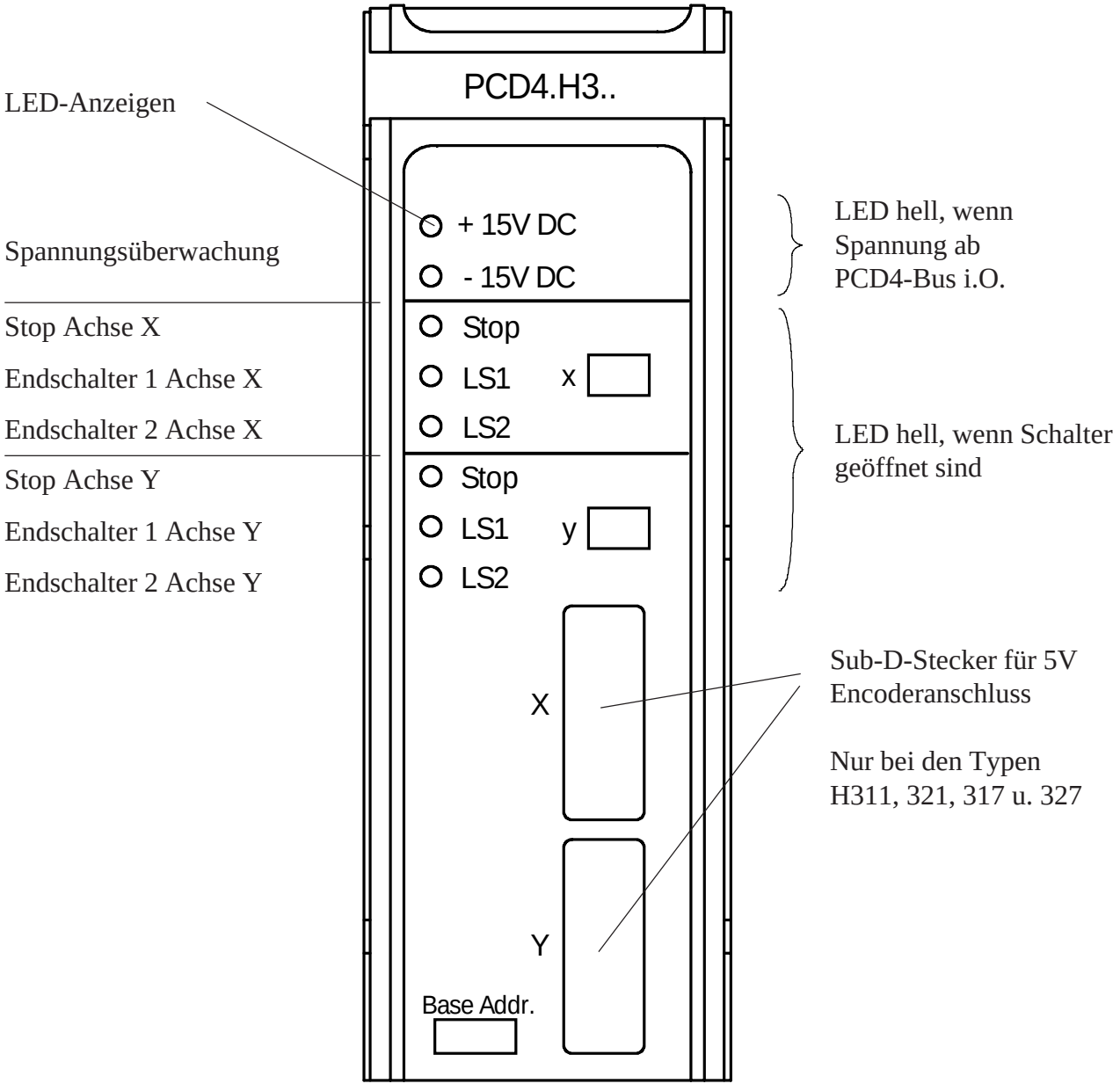

- D/A-Wandler Module sind nur bestückt bei den Typen : H310, 320, 311 u. 321

- Jumper zur Wahl des PWM-Ausganges nur bestückt bei den Typen : H316, 317, 326 u. 327

### 3.1.2 Zusatzprint für 5V-Encoder (2 Achsen)



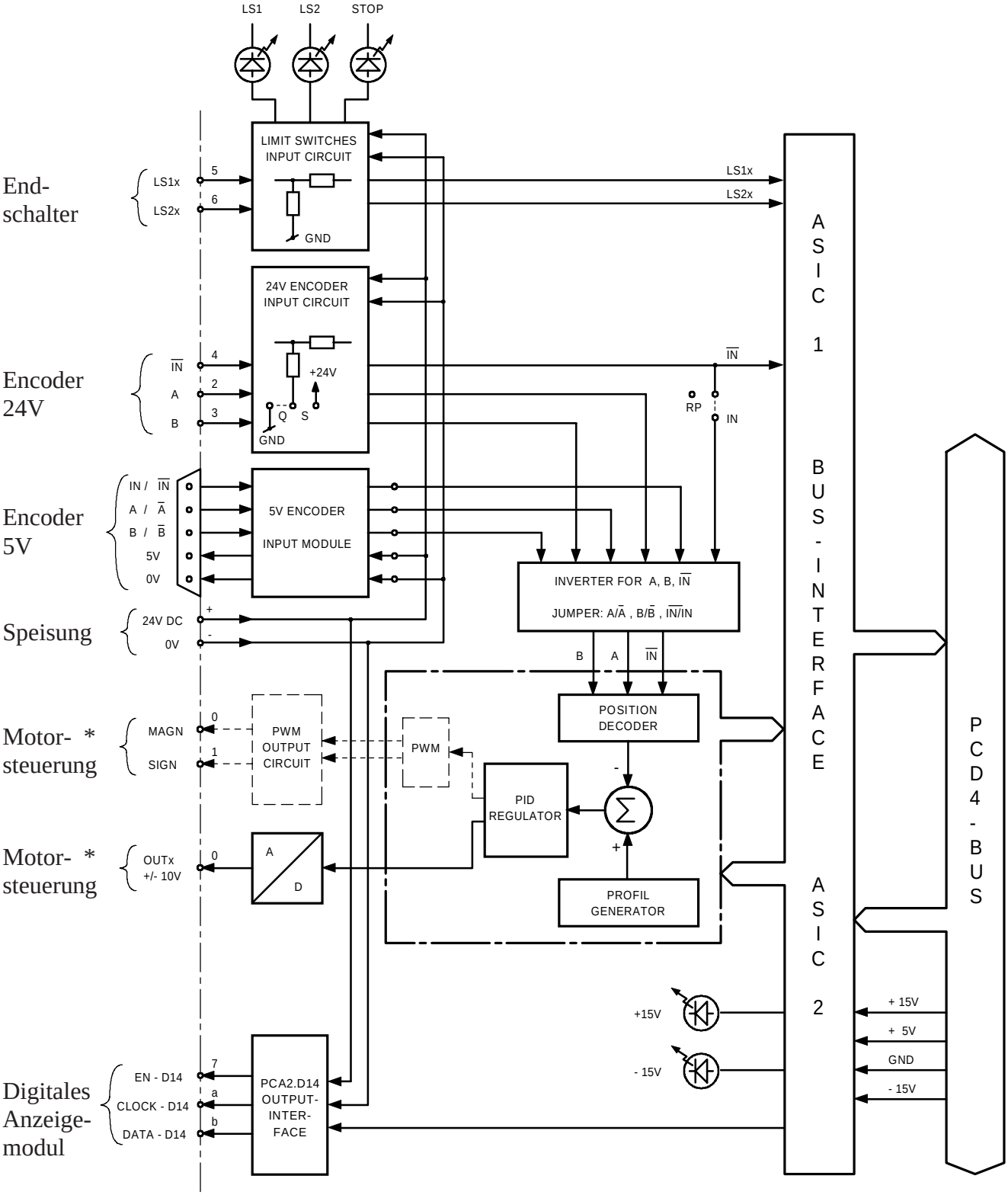
## 3.2 Frontplatte



Notizen:

# 4. Blockschaltbild

Dargestellt ist nur die X-Achse

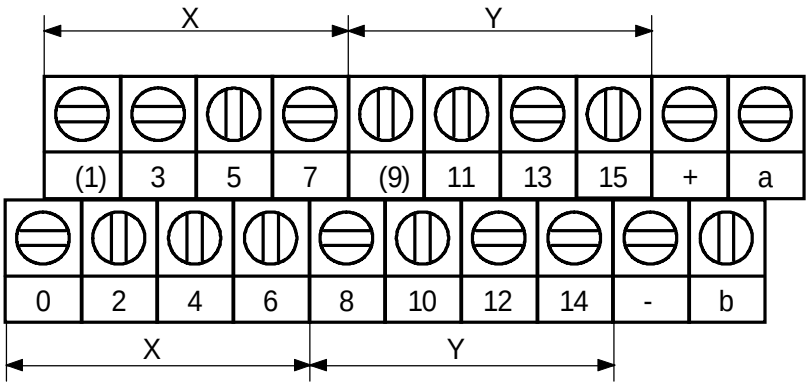



*) Ausgang zu Motorsteuerung PWM oder analog ±10V sind alternativ. (siehe Typenübersicht)

Notizen:

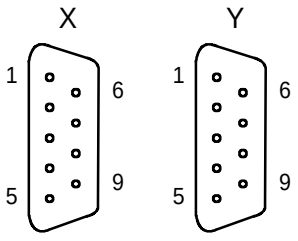
# 5. Anschlüsse und Adressierung

## 5.1 Anschlüsse




Klemmen-Anschlüsse (auf Bus-Modul) :

| | | | | | |
|---|---|---|---|---|---|
| 0 | Out-x | (PWM-MAGx) | 8 | Out-y | (PWM-MAGy) |
| 1 | | (PWM-SIGNx) | 9 | | (PWM-SIGNy) |
| 2 | Phase-Ax | | 10 | Phase-Ay | |
| 3 | Phase-Bx | | 11 | Phase-By | |
| 4 | /INx (Ref. x) | | 12 | /INy (Ref. y) | |
| 5 | LS1x | | 13 | LS1y | |
| 6 | LS2x | | 14 | LS2y | |
| 7 | EN-D14x | | 15 | EN-D14y | |
| - | GND | | + | +24V | |
| a | CLK-D14 | | b | DATA-D14 | |




D-Sub-Stecker :
(weiblich)

| | | |
|---|---|---|
| 1 | PGND | PGND |
| 2 | Ax | Ay |
| 3 | /Ax | /Ay |
| 4 | Bx | By |
| 5 | GND | GND |
| 6 | /Bx | /By |
| 7 | INx | INy |
| 8 | /INx | /INy |
| 9 | +5V | +5V |

Legende:

| | |
|---|---|
| Out-x, y | Ausgang der Stellgrösse ±10V bzw. PWM-MAGN |
| PWM-SIGN | Ausgang |
| Phase-A | Encoder-Eingang für Phase A (24V) |
| Phase-B | Encoder-Eingang für Phase B (24V) |
| /IN (Ref.) | Encoder-Eingang für Indexmarke bzw. Referenzschalter |
| LS1 | Eingang für Endschalter (24V) |
| LS2 | " " " " |
| EN-D14 | Ausgang ENABLE für PCA2.D14 Anzeigemodul |
| CLK-D14 | Ausgang CLOCK " " " |
| DATA-D14 | Ausgang DATA " " " |
| +24V | Einspeisung für +24V DC |
| +5V | 5V-Ausgang zur Speisung des RS422 Encoders |
| GND | Minus-Anschluss für die 24VDC - respektive 5VDC-Speisung |
| PGND | Schutzerdeanschluss für die 5V-Encoder. Muss nicht angeschlossen werden, da die Kabel-abschirmung über das Metallgehäuse des D-Sub-Steckers und die Schraubverbindung verbunden ist mit der Schutzerde. Siehe auch Kapitel 6.4 . |

## 5.2 Adressierung

Das Modul belegt 16 Adressen am PCD4-Bus. Da das Modul über zwei Businterface (ASIC) verfügt, können die 16 Adressen doppelt verwendet werden.

Bedeutung der 16 Adressen :

| | Data-In : | Data-Out : |
|---|---|---|
| | 0 Data-Bus (LSB) | 0 Data-Bus (LSB) |
| | 1 " " | 1 " " |
| | 2 " " | 2 " " |
| | 3 " " | 3 " " |
| | 4 " " | 4 " " |
| | 5 " " | 5 " " |
| | 6 " " | 6 " " |
| | 7 Data-Bus (MSB) | 7 Data-Bus (MSB) |
| X | 8 —— | 8 Write (WR) |
| X | 9* Limit-Switch (LS1x) | 9 Read (RD) |
| X | 10* Limit-Switch (LS2x) | 10 Port-Sel. (PS) |
| X | 11* In/Ref.Switch X-Axis | 11* Chip-Sel. (1/0=X/Y-Axis) |
| Y | 12 —— | 12 Clock (PCA2.D14) |
| Y | 13* Limit-Switch (LS1y) | 13 Data ( " ) |
| Y | 14* Limit-Switch (LS2y) | 14* Enable X-Axis (PCA2.D14) |
| Y | 15* In/Ref.Switch Y-Axis | 15* Enable Y-Axis (PCA2.D14) |

Alle angeführten Adressen sind relativ.
Absolutadresse = Modul-Basisadresse + Relativadresse

Für den Anwender sind nur die Adressen mit * von Bedeutung.
Alle andern Adressen werden von Funktionsblöcken benützt.

Das Enable-Signal für die PCA2.D14 Anzeigen ist aktiv tief. Auf dem Positioniermodul befindet sich ein Inverter für diese Ausgänge.

Notizen:

# 6. Funktionsbeschreibung

## 6.1 Betriebsarten

Grundsätzlich werden zwei verschiedene Betriebsarten unterschieden :

- POSITIONSGEREGELTER BETRIEB

- DREHZAHLGEREGELTER BETRIEB

**Positionsgeregelter Betrieb**

Die Positionierung erfolgt nach folgendem Befehlsschema:

1. Position- und Parametereingabe für das Geschwindigkeitsprofil
2. Positionierung starten
3. Signal abwarten für "**Zielposition erreicht**".

Im Positionierbetrieb wird nach Eingabe verschiedener Parameter (PID-Faktoren,Beschleunigung,Geschwindigkeit usw.) eine vorgegebene **Zielposition geregelt angefahren**, wobei Geschwindigkeit, PID-Faktoren und Zielposition während der Bewegung verändert werden dürfen.

**Drehzahlgeregelter Betrieb**

Befehlsschema :

1. Parametereingabe für das Geschwindigkeitsprofil
2. Bewegung starten
3. Bewegung abbrechen durch eingeben eines **Stopbefehls**

Im drehzahlgeregelten Betrieb wird mit der definierten Beschleunigung beschleunigt bis zur Sollgeschwindigkeit. Es wird **geregelt mit dieser Geschwindigkeit gefahren bis ein Stopbefehl erfolgt**. Die Sollgeschwindigkeit kann während der Bewegung verändert werden.

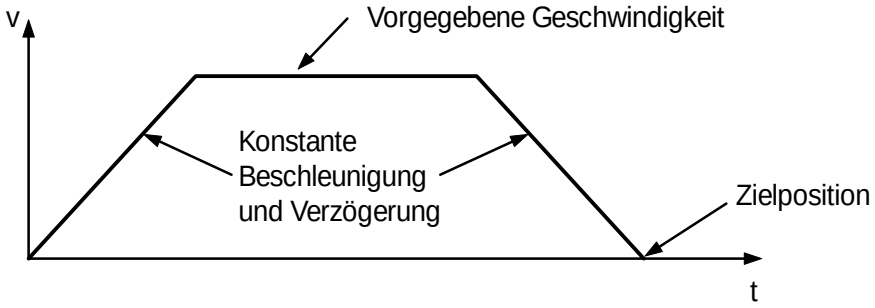
**Funktionseinheiten**

Aus dem Blockschaltbild ist ersichtlich, dass das Positioniermodul im Wesentlichen aus folgenden Funktionseinheiten besteht:

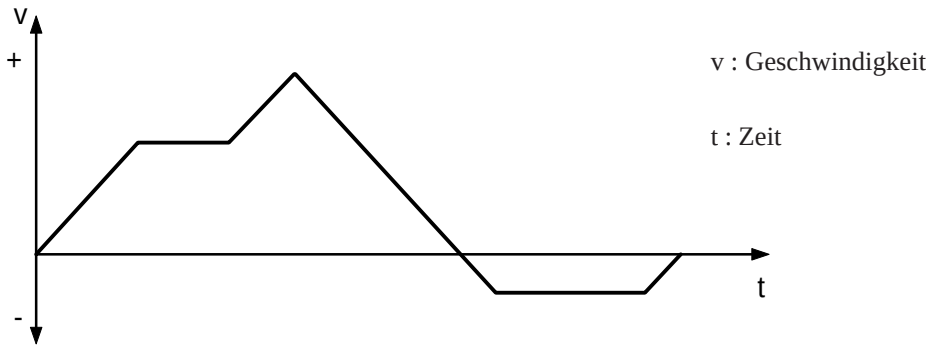
- Generator für Geschwindigkeitsprofil
- PID-Regler
- Positionsdecoder und Eingangsschaltung
- Businterface (ASIC) zum PCD-Bus
- D/A-Wandler für die analoge Stellgrösse
  oder Generator für Puls-Weiten-Modulation (PWM)

## 6.2 Generator für das Geschwindigkeitsprofil

Entsprechend der vorgegebenen Beschleunigung und Geschwindigkeit berechnet der Profilgenerator die Sollgeschwindigkeit in Funktion der Zeit im Positionier- und Drehzahlmodus. Im Positionierbetrieb wird während der Bewegung die Differenz von Sollposition und Istposition dem PID-Regler zugeführt. Damit wird eine sehr genaue Positionierung des Motors erreicht.




Standard Geschwindigkeitsprofil




Geschwindigkeitsprofil mit geänderter Sollgeschwindigkeit und Zielposition während der Bewegung.

Die Geschwindigkeit und die Zielposition können an einer beliebigen Position während der Bewegung verändert werden, und der Controller wird entsprechend mit der definierten Beschleunigung beschleunigen oder bremsen.

Im drehzahlgeregelten Betrieb beschleunigt der Controller bis zur, vom Anwender definierten Sollgeschwindigkeit, und fährt mit konstanter Geschwindigkeit bis ein Stopbefehl erfolgt.
Funktionsprinzip der Drehzahlregelung:
Die Sollposition wird kontinuierlich (entsprechend der verlangten Geschwindigkeit) vergrössert. Die Differenz aus Soll- und Istposition (welche mit dem Encoder erfasst wird) wird wiederum dem PID-Regler zugeführt. Dieser kompensiert Geschwindigkeitsschwankungen, verursacht durch irgendwelche Störeinflüsse, sofort durch vergrössern oder verkleinern der Stellgrösse.

Falls der Motor die Sollgeschwindigkeit nicht erreicht (z.B. durch blockierten Rotor), so wird die Differenz zwischen Soll- und Istposition sehr gross. Dies erzeugt eine Positionsfehlermeldung, welche einen Interrupt oder einen automatischen Stop des Motors auslösen kann. Der Wert für den maximal zulässigen Positionsfehler ist programmierbar.

## 6.3 PID-Regler

Mit Hilfe des PID-Reglers kann der Motor die Zielposition genau anfahren und wird in dieser Position gehalten, da der Regler solange aktiv ist bis ein Stopbefehl erfolgt.
Der Controller benutzt folgenden Regelalgorithmus:

$$U(n) = kp * e(n) + ki * \sum_{N=0}^{n} e(n) + kd * [e(n') - e(n'-1)]$$

wobei :

- U(n) --> Stellgrösse für den Motor
- e(n) --> Positionsfehler bei der n'ten Abtastung
- n --> Abtastung für den Integralteil
- n' --> Abtastung für den Differentialteil
- kp --> Proportionalfaktor
- ki --> Integralfaktor
- kd --> Differentialfaktor

Parameter, die vom Anwender programmiert werden können:

- Regelfaktoren kp, ki, kd
- Differentialabtastzeit
- Integrationsgrenze (IL) für den Integralanteil

Die **Regelfaktoren kp, ki und kd** dürfen während einer Bewegung verändert werden.
Die **Abtastzeit für den Proportional- und Integralteil** beträgt 341µs. Das bedeutet die Stellgrösse wird in einem Intervall von 341µs aufgefrischt!
Die **Abtastzeit vom Differentialteil** kann in Schritten von 341µs eingestellt werden (max. 256*341µs). Für den Betrieb mit langsamen Geschwindigkeiten sollte eine grössere Abtastzeit gewählt werden.

**Integrationsgrenze IL** : begrenzt wird der Betrag vom Ausdruck

$$ki * \sum_{N=0}^{n} e(n)$$

## 6.4 Positionsdecoder und Eingangsschaltung

**Positions- und Geschwindigkeitserfassung**

Die genaue Position und/oder Drehzahl des Motors wird mit einem Inkremental-Drehencoder erfasst. Folgende Encodersignale können angeschlossen werden:

PCD4 .H310
" .H320
" .H316
" .H326

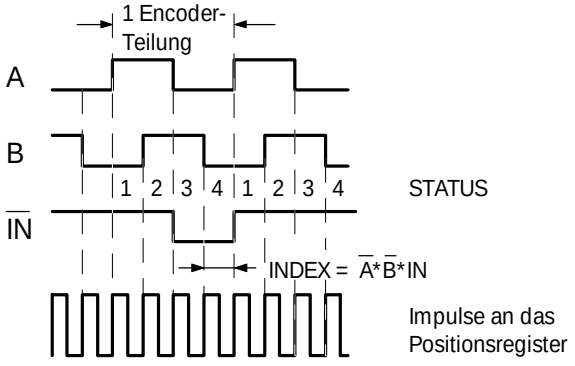
} A,B,IN bzw. $\overline{A}$,$\overline{B}$,$\overline{IN}$ (Klemmenanschlüsse)
24V-Signale im Quell- oder Senkbetrieb

PCD4 .H311
" .H321
" .H317
" .H327

} A,$\overline{A}$; B,$\overline{B}$; IN,$\overline{IN}$; (Front-Sub-D-Stecker)
5V RS 422 Eingänge (Antivalent Line Driver)

**Eingänge A, B, $\overline{IN}$ :**

Zustandsdiagramm der Signale A,B,$\overline{IN}$ am Positionsdecoder :




Drehrichtung

| | Neg. | | | | Pos. | | | |
|---|---|---|---|---|---|---|---|---|
| STATUS | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| A | I | I | O | O | I | I | O | O |
| B | O | I | I | O | O | I | I | O |

**Eingänge A, B :**

Auf jeden Zustandswechsel (0->1 und 1->0) der Signale A und B wird das interne Positionsregister um 1 erhöht oder erniedrigt. Dadurch erhält man die vierfache Auflösung der Encoderteilung. Die Eingabe für die Zielposition muss dementsprechend auch mit vier multipliziert werden.
Für den Positionsdecoder müssen die Signale genau die Folge aufweisen, wie im Bild oben dargestellt ist. Liefert der Encoder andere Signale, so müssen diese mittels Jumper entsprechend invertiert werden (siehe folgende Seiten) .

**Eingang $\overline{IN}$ :**

Bei den Modulen für 24V-Encoder (Typen H310, 320, 316 u. 326) kann der Eingang $\overline{IN}$ verwendet werden als Indeximpuls- (Nullsignal vom Encoder) oder als Referenzpunkt-Eingang.

- Verwendung als Indeximpulseingang :
  (Jumper IN/RP in Position IN)

  Jedesmal wenn alle 3 Encodersignale den Zustand Null haben und vorher der Funktionsblock "SetIP" (Set Index Position) aufgerufen wurde, wird die absolute Motorposition in das Indexpositionsregister geschrieben .

- Verwendung als Referenzpunkteingang :
  (Jumper IN/RP in Position RP)

  Es kann ein Referenzschalter angeschlossen werden, um z.B. die Nullposition zu definieren. Dabei muss jedoch Jumper IN/RP in Position RP gesteckt werden, damit der Eingang nicht mehr aktiv ist für den Positionsdecoder.
  Bei Verwendung der Module H310,320,316 oder 326, und Eingang $\overline{IN}$ als Indexsignal, muss der Referenzschalter an einen Eingang von einem digitalen Eingangsmodul (z.B. E100) angeschlossen werden.

**Module für den Anschluss von 5V Encodersignalen**
Diese sind mit einem zusätzlichen Print bestückt, wo die 5V Speisung erzeugt wird und die Encoderanschlüsse auf Sub-D-Stecker geführt sind. Bei Verwendung der Module H311,321,317 oder 327 kann sowohl ein Referenzschalter (an den Klemmen) als auch das Indexsignal (Sub-D-Stecker) angeschlossen werden.
Wie aus dem folgenden Eingangsschaltbild ersichtlich ist, muss für die Verbindung zum 5V-Encoder ein abgeschirmtes Kabel verwendet werden:

- max. Kabellänge : 20m
- min. Leiterquerschnitt : 0.25mm$^2$

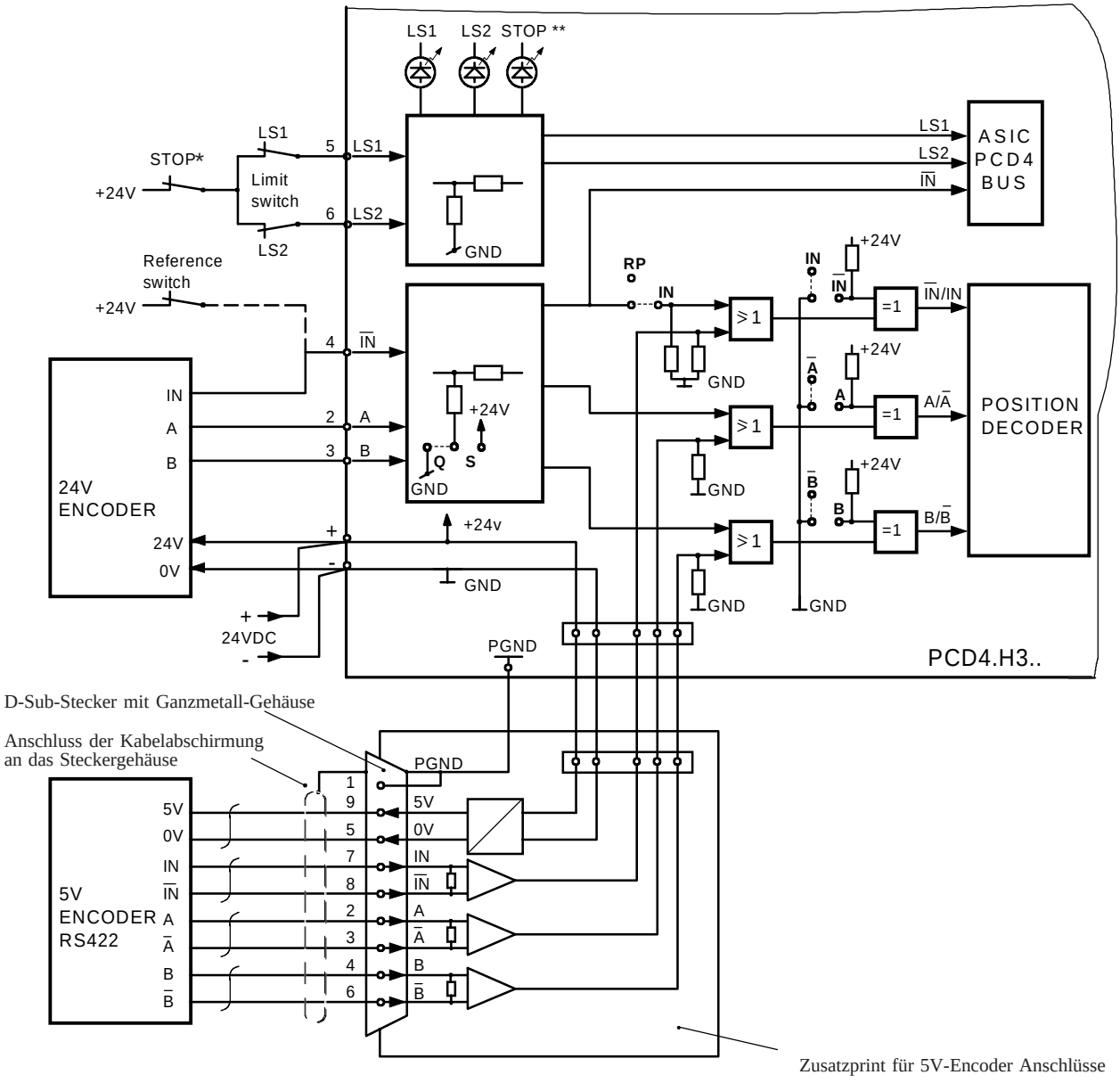
Um die angegebene Störsicherheit zu gewährleisten, muss ein D-Sub-Stecker mit Ganzmetall-Gehäuse verwendet werden, was die direkte Verbindung zur Schutzerde (PGND) des PCD4-Systems ergibt.

**Endschalter, Referenzschalter**

Der Anschluss für Endschalter (Eingänge LS1 und LS2) und Referenzschalter (Eingang IN) erfolgt unter 24VDC im Quellbetrieb. Die Signale dieser Schalter werden direkt auf den PCD4-Bus geführt. Das bedeutet, sie müssen vom Anwenderprogramm überwacht werden, um die notwendigen Aktionen auszulösen.
Diese Endschalter (LS1/2) und ein allenfalls in serie geschalteter Stop-Schalter dürfen keine Abschaltfunktionen im Sinne von **Sicherheitsvorschriften** übernehmen. Dafür sind zusätzliche Sicherheits-Endschalter und Notstop-Schalter vorzusehen, welche direkt auf die Hauptstromkreise der Antriebe wirken.

## Eingangsschaltbild und Anschlüsse

(Dargestellt ist nur eine Achse)




*) Bezüglich Sicherheitsaspekte siehe vorangehenden Abschnitt "Endschalter, Referenzschalter"

**) Zustandstabelle der LED's (LS1, LS2 u. STOP):

| EINGÄNGE | | LED | | |
|---|---|---|---|---|
| LS1 | LS2 | LS1 | LS2 | STOP |
| 24V | 24V | DUNKEL | DUNKEL | DUNKEL |
| OFFEN | 24V | HELL | DUNKEL | DUNKEL |
| 24V | OFFEN | DUNKEL | HELL | DUNKEL |
| OFFEN | OFFEN | DUNKEL | DUNKEL | HELL |

**Jumper-Selektion** (siehe auch Abschnitt 3.1 Leiterplatte) **:**

| Jumper | Funktion | Position ab Werk | |
|---|---|---|---|
| Q/S 1) | Quell/Senk-Betrieb Eingänge A,B,$\overline{IN}$ | Quellbetrieb (Q) | Nur für 24V Encoder |
| IN/RP 2) | Eingang für Indeximpuls oder Referenzpunkt | Indeximpuls (IN) | |
| A/$\overline{A}$ 3) | Invertierung Phasensignal A | $\overline{A}$ | Für alle Encoder 24V und 5V |
| B/$\overline{B}$ 3) | Invertierung Phasensignal B | $\overline{B}$ | |
| IN/$\overline{IN}$ 3) | Invertierung Indeximpuls $\overline{IN}$ | IN | |

1) Die Umschaltung Quell/Senk-Betrieb der 24V-Encodersignale erfolgt mit 1 Jumper gemeinsam für beide Achsen.

2) Bei **24V-Encodern** kann alternativ über die Klemmen 4 (X-Achse) bzw. 12 (Y-Achse) angeschlossen werden :

- Indexsignal des Encoders
  oder
- Referenzschalter

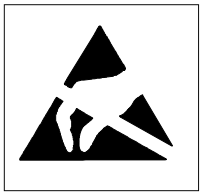
Bei **5V-Encodern** werden die Indexsignale über den D-Sub-Stecker zugeführt. D.h. die Klemmen 4 und 12 stehen frei für den Referenzschalter zur Verfügung (Jumper in Pos. "RP"). Der Jumper "IN/RP" ist auf der Leiterplatte jeweils für jede Achse getrennt wählbar.

3) Die Invertierung aller Encodersignale (24V und 5V) erfolgt mit je 1 Jumper für A, B und IN ebenfalls gemeinsam für beide Achsen.

### Öffnen des Modulgehäuses zum wechseln der Jumper

Um die Jumperposition wechseln zu können, muss die Leiterplatte aus dem Modulgehäuse herausgezogen werden. Dies geschieht durch Eindrücken der seitlichen Schnappverklinkungen der Frontabdeckung. Anschliessend ist auf der linken Modulseite oben die Printbefestigungs-schraube herauszuschrauben, womit die Leiterplatte aus dem Gehäuse gezogen werden kann.

Nach richtigem Stecken der Jumper, das Gehäuse wieder schliessen und die Printbefestigungsschraube anbringen.

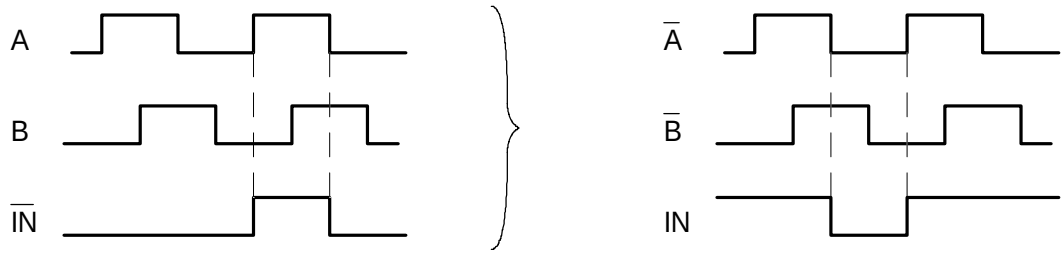


**Achtung:** Sowohl auf dem Basisprint, wie auch auf den Analogmodulen befinden sich Bauteile, die bezüglich elektrostatischen Entladungen empfindlich sind.

**Anwendungsbeispiel zur Jumper-Wahl:**

Ein Encoder liefert folgende 24V-Signale im Quellbetrieb :

Signale ab Encoder Sollsignale am Positionsdecoder




Damit die Signale am Eingang vom Positionsdecoder die verlangte Folge aufweisen, müssen alle drei Signale invertiert werden.

Jumper : A/$\overline{A}$ --> $\overline{A}$
B/$\overline{B}$ --> $\overline{B}$
$\overline{IN}$/IN --> IN

} entspricht den Positionen ab Werk

**Wichtig :**

Wird bei **24V-Encodern** mit den **Referenzschaltern** gearbeitet (Jumper "IN/RP" in Pos. "RP"), so muss gleichzeitig der Jumper "$\overline{IN}$/IN" in Pos. "**IN**" gesteckt werden.

Grund: Das Indexsignal $\overline{IN}$ darf am Eingang vom Positionsdecoder nicht dauernd Null sein, da dies bei hohen Drehzahlen zu einer Fehlfunktion des Controllers führen kann.

## 6.5 D/A-Wandler (analoge Stellgrösse)

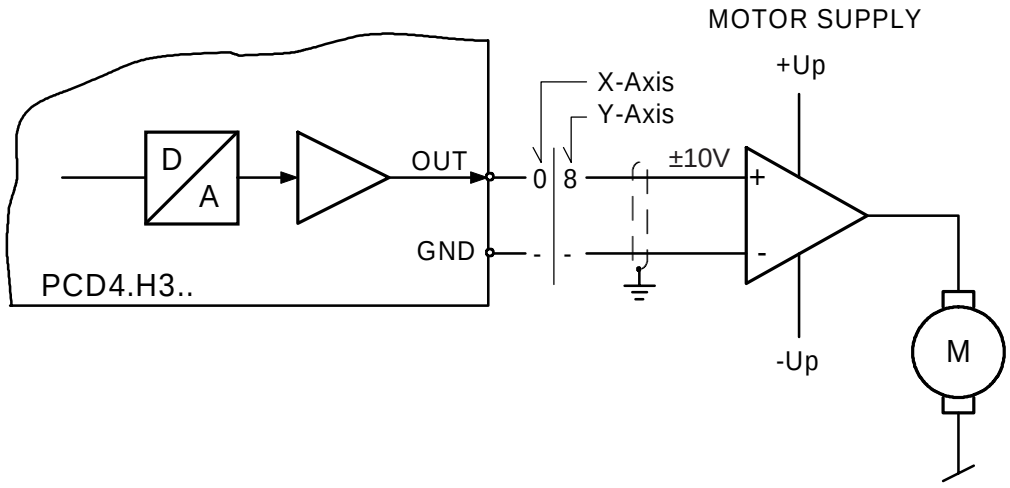
Die Module PCD4.H310
" .H320
" .H311
" .H321

verfügen über einen analogen Ausgang für die Motorstellgrösse.

Pro Achse ist ein 12 Bit D/A-Wandler eingesetzt.

**Analogausgang-Anschluss:**

(Dargestellt ist nur eine Achse)



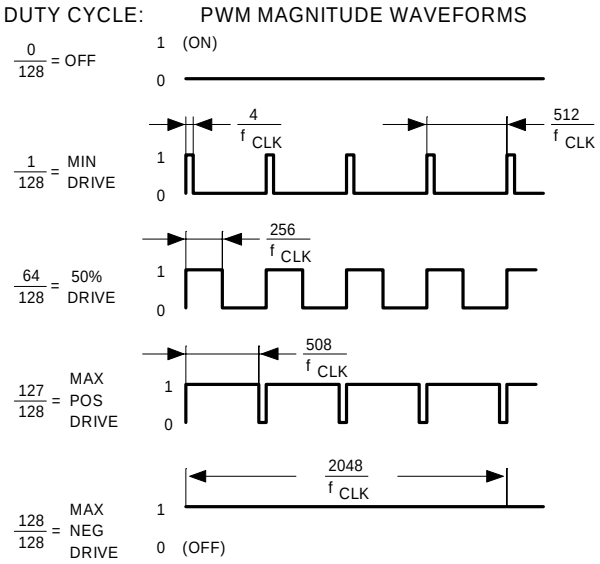
## 6.6 PWM - Generator

Die Module PCD4.H316 } verfügen über einen PWM-Ausgang
" .H326 } mit SIGN- und MAGNITUDE-Signal
" .H317 } oder mit einer Ausgangslogik zur
" .H327 } direkten Ansteuerung eines Brücken-
verstärkers.

Bei manchen Leistungsverstärkern ist obengenannte Logik integriert. Deshalb besteht die Möglichkeit das SIGN- und MAGNITUDE-Signal mittels Jumper-Wahl entweder direkt oder über eine Logik auszugeben.

**PWM-SIGNAL am Ausgang des PWM-Generators:**
(ohne SIGN-Signal)

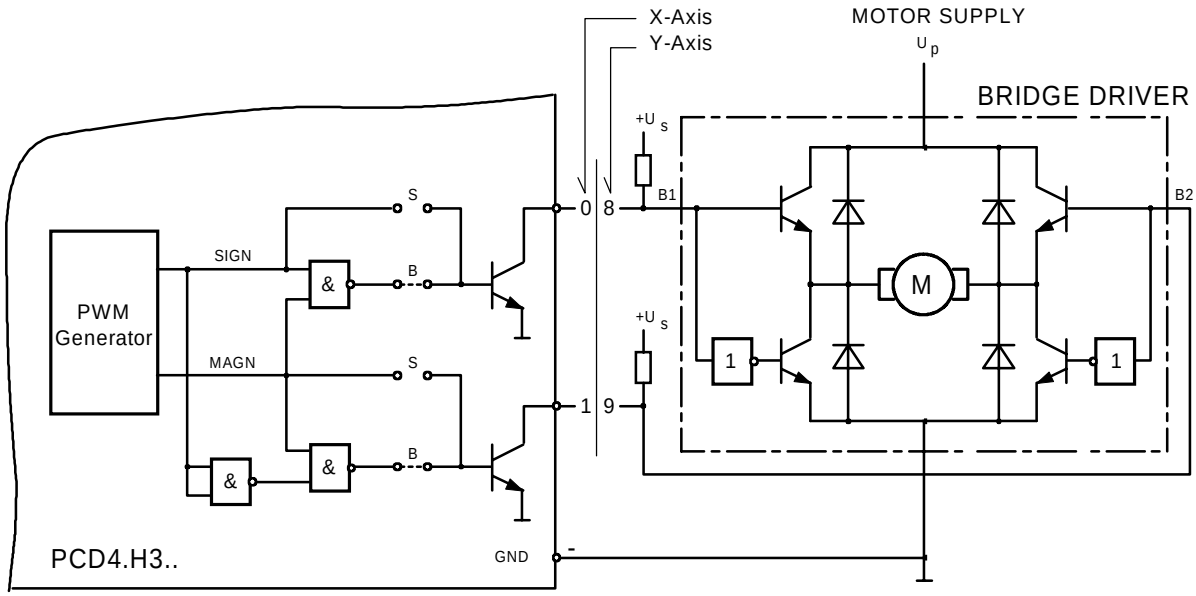
Die Auflösung des Signals beträgt 8 Bit.
Bereich: -128 .... +127



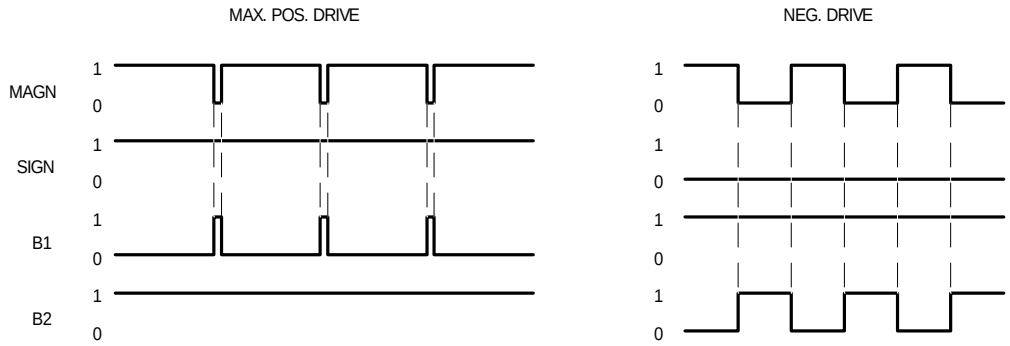

$f_{CLK}$ = 6MHz

**Blockschaltbild PWM-Ausgang:**

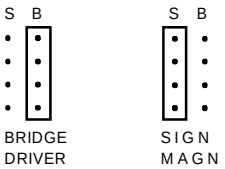
(Dargestellt ist nur eine Achse)




Folgendes Diagramm zeigt den Zusammenhang zwischen SIGN/ MAGN-Signal und dem Ausgang zur direkten Ansteuerung eines Brückenverstärkers.




Jumper-Selektion:




Ausgang für X- und Y-Achse separat wählbar.

Position ab Werk : SIGN, MAGN (Pos. S)

Notizen:

# 7. Programmerstellung für das H3-Modul

## 7.1 Installation der Software

### 7.1.1 Das Softwarepaket PCD9.H3E1

(PCD9.H3E1 für 5¼" Disketten, PCD9.H3E6 für 3½" Disketten)
Dem Benutzer des H3-Moduls steht das Softwarepaket PCD9.H3E1 zur Programmierung des Moduls zur Verfügung. Das Paket enthält Funktionsbausteine, welche auf PCD-Anwenderprogramm basieren. Durch Aufrufen dieser Funktionsbausteine im Anwenderprogramm wird das H3-Modul programmiert.
Das Paket enthält die folgenden zwei Dateien:

**H3DEF.SRC** In dieser Datei sind alle Symbolzuweisungen der gesamten H3-Software zusammengefasst. Hier wird die ganze H3-Installation konfiguriert.

**H3FB.SRC** Diese Datei enthält die Funktionsbausteine zur Programmierung des Moduls.

Das Gesamtpaket hat folgenden Umfang:

- Anzahl Programmzeilen $\leq$ 1250
- Benutzte FB-Ebenen 6

### 7.1.2 Assemblieren und Linken der Dateien

Es gibt grundsätzlich zwei Möglichkeiten die H3-Dateien zu assemblieren und zu linken. Der Anwender arbeitet mit externen Symboldefinitionen oder die Symboldefinitions-Datei H3DEF.SRC wird mit der Assemblerdirektive $INCLUDE eingebunden. Die H3-Dateien sind bereits für beide Arbeitsweisen vorbereitet. Je nach Methode der Assemblierung müssen in den H3-Dateien lediglich die Direktiven für bedingtes Assemblieren gesetzt werden.
Die Direktiven sind voreingestellt für eine Assemblierung ohne externe Symbolzuweisungen.

**Arbeiten ohne externe Symbolzuweisungen**

Die Symboldefinitionssdatei H3DEF.SRC wird mit $INCLUDE in das Anwenderprogramm eingebunden. Die Direktiven in den H3-Dateien müssen wie folgt definiert werden:

- H3DEF.SRC : PUBLSYM EQU 0

- H3FB.SRC : EXTNSYM EQU 0

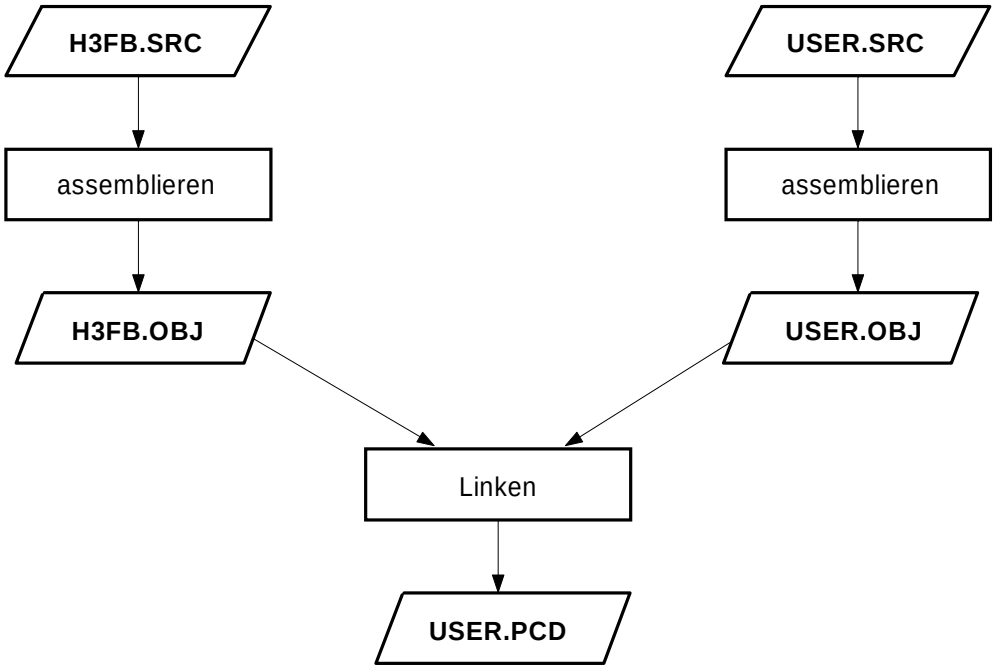
Einbinden der H3-Dateien in das Anwenderprogramm:

Anwenderprogrammdatei (z.B. USER.SRC)

```
Dateibeginn (Titel,Programmkopf usw.)

$INCLUDE        H3DEF.SRC
$INCLUDE        Weitere Anwender-
                programm Dateien
      "
      "
Anwenderprogramm
      "
      "
Dateiende
```

Die H3-Symboldef. Datei muss vor allen anderen Anwender-dateien (die Symbole aus der H3DEF.SRC Datei verwenden) eingebunden werden.

Untenstehendes Diagramm zeigt, wie die Dateien assembliert und gelinkt werden.



**Arbeiten mit externen Symbolzuweisungen**

Alle Dateien werden einzeln assembliert und dann gelinkt. Die Direktiven in den H3-Dateien müssen wie folgt definiert werden:

- H3DEF.SRC : PUBLSYM EQU 1

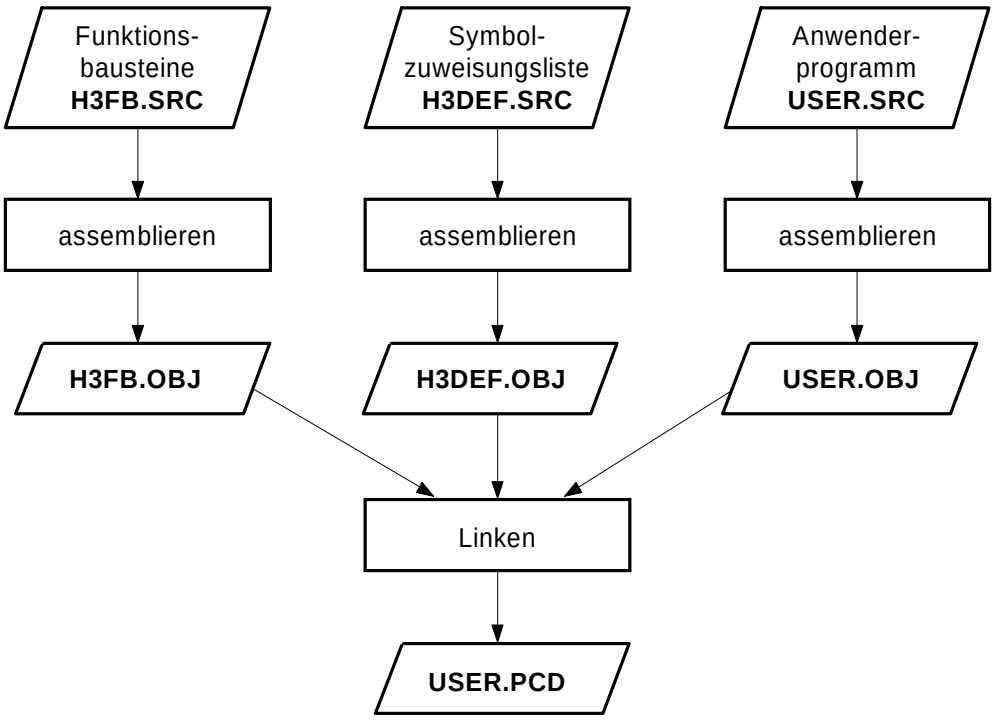
- H3FB.SRC : EXTNSYM EQU 1

Wählt man diese Methode der Assemblierung, so müssen alle verwendeten Symbole (aus der H3DEF.SRC Datei) im Anwenderprogramm als External definiert werden. Die Registerblock- und Flagfeldkonstanten müssen in der Anwenderdatei definiert werden, weil der Assembler eine Addition von zwei extern definierten Symbolen als symbolischen Operand (Bsp.: SET F FStart+FA1) nicht zulässt.
Definition von "RAi" und "FAi" im Anwenderprogramm (siehe dazu auch Kapitel 7.1.3)

```
RA1     EQU     0*NoRfeA
RA2     EQU     1*NoRfeA
 ¦

FA1     EQU     0*NoFfeA
FA2     EQU     1*NoFfeA
 ¦
```

Für die Symbole "NoRfeA" und "NoFfeA" müssen aus dem oben erwähnten Grund die Absolutwerte eingesetzt werden. Diese können der H3DEF-Datei entnommen werden.

Untenstehendes Diagramm zeigt, wie die H3-Dateien und das Anwenderprogramm assembliert und gelinkt werden müssen.



### 7.1.3 Konfiguration der H3-Installation in der H3DEF.SRC Datei

In der Datei muss die H3-Installation konfiguriert werden, bevor mit Programmieren begonnen wird.
Es müssen Angaben gemacht werden wie z.B. Basisadresse des ersten H3-Moduls, Anzahl der eingesetzten Achsen, Basisadressen der Speichermedien usw. Alle Definitionen sind bereits auf einen Standardwert voreingestellt und können angepasst werden. Untenstehende Tabelle zeigt, wie die Angaben in der Datei gemacht werden. Diese Tabelle steht am Anfang der Datei.

| Symbolname | | Adresse | Kommentar |
|---|---|---|---|
| FMAH3 | EQU | 0 | ; First Module Address H3<br>; Definiert die Basisadresse des<br>; ersten H3-Moduls<br>; Beachte: alle H3-Module müssen<br>; lückenlos nacheinander auf den<br>; PCD4-Bus gesteckt werden. |
| IMode | EQU | 6 | ; Initialisation Mode<br>; Wird entsprechend dem verwendeten<br>; Sollwertausgang definiert (Analog/PWM).<br>; Module H310,311,320 u. 321: IMode = 6<br>; Module H316,317,326 u. 327: IMode = 5 |
| MNA | EQU | 2 | ; Max. Number of Axes<br>; Definiert die Anzahl der eingesetzen<br>; Achsen. Entsprechend dieser Angabe<br>; werden die benötigten Speichermedien<br>; (Register, Flag ) reserviert. |
| BAF | EQU | 2000 | ; Base Address Flags<br>; Basisadresse der benützten Flag |
| BAR | EQU | 2000 | ; Base Address Registers<br>; Basisadresse der benützten Register |
| BAC | EQU | 1000 | ; Base Address Counters<br>; Basisadresse der benützten Zähler |
| BAFB | EQU | 900 | ; Base Address FunctionBlocks<br>; Basisadresse der benützten Funktions-<br>; blöcke |

```
Symbolname   Adresse        Kommentar

RA1          EQU 0*NoRfeA ; Registerblock constant Axis Nr. 1
                          ; Registerblock-Konstante Achse 1
                          ; Diese Konstante zeigt auf das erste
                          ; Register des von Achse 1 belegten
                          ; Registerblocks.
                          ; Die Konstante "NoRfeA" definiert die
                          ; Anzahl belegter Register pro Achse.
                          ; "NoRfeA" ist fest definiert und darf
                          ; nicht geändert werden.

RA2          EQU 1*NoRfeA ; Registerblock constant Axis Nr. 2
                          ; Registerblock-Konstante Achse 2

FA1          EQU 0*NoFfeA ; Flagrange constant Axis Nr. 1
                          ; Flagfeld-Konstante Achse 1
                          ; Diese Konstante zeigt auf das erste Flag
                          ; des  von Achse 1 belegten Flagfeldes.
                          ; Die Konstante "NoFfeA" definiert die

                          ; Anzahl belegter Flag pro Achse.
                          ; "NoFfeA" ist fest definiert und darf nicht
                          ; geändert werden.

FA2          EQU 1*NoFfeA ; Flagrange constant Axis Nr. 2
                          ; Flagfeld-Konstante Achse 2
```

Die Liste mit den Registerblock-Konstanten (RA1,RA2) und Flagfeld-Konstanten (FA1,FA2) muss, falls mehr als zwei Achsen eingesetzt sind, entsprechend erweitert werden.

Beispiel:

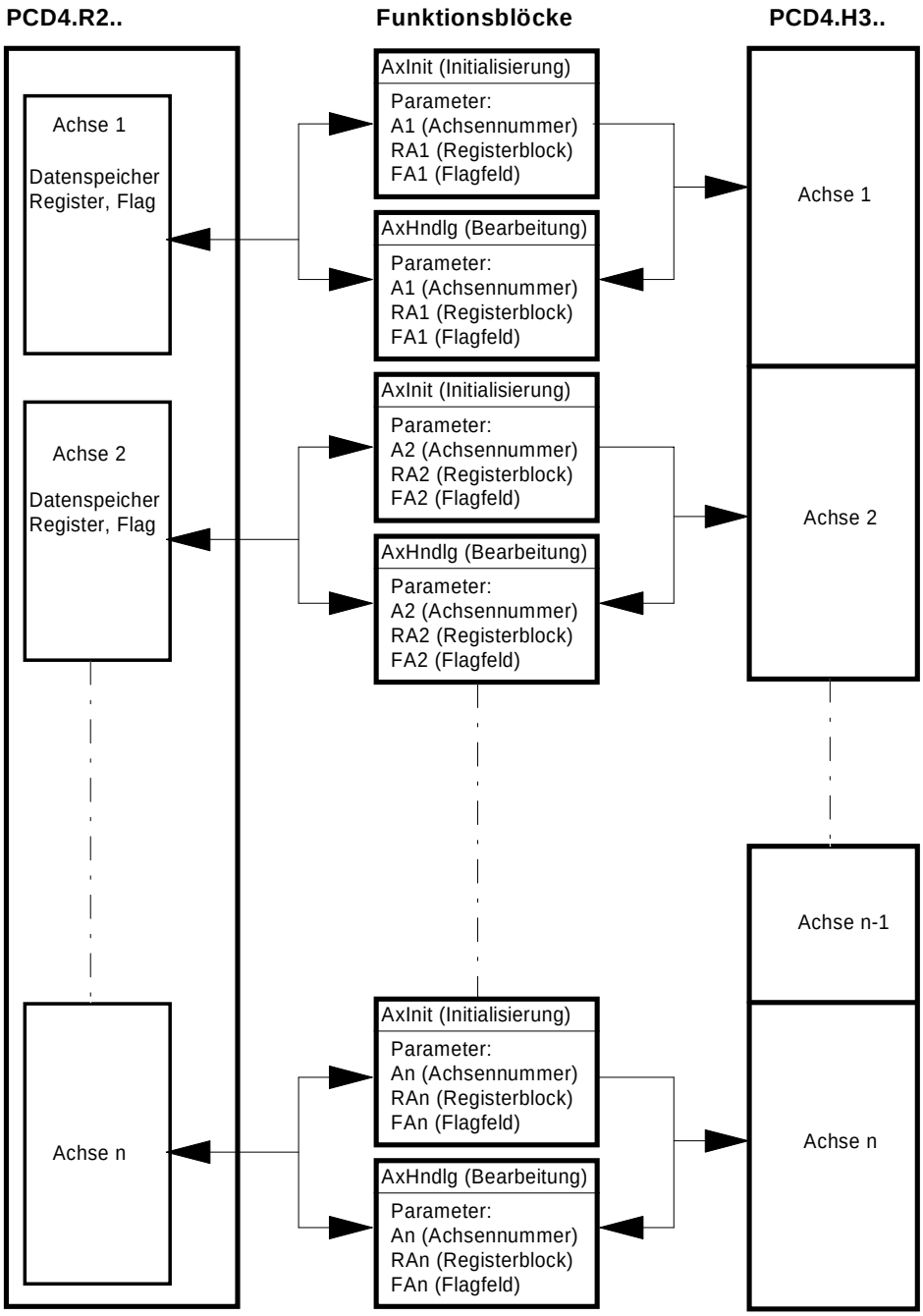
| Achse i | RAi | FAi |
|---|---|---|
| 1 | 0*NoRfeA | 0*NoFfeA |
| 2 | 1*NoRfeA | 1*NoFfeA |
| 3 | 2*NoRfeA | 2*NoFfeA |
| 4 | 3*NoRfeA | 3*NoFfeA |
| 5 | 4*NoRfeA | 4*NoFfeA |
| usw. | | |

Ist die Systemkonfiguration vorgenommen, so interessiert, wieviele Speichermedien die H3-Software insgesamt belegt. Am Ende der H3DEF.SRC Datei steht eine Tabelle, in welcher beim Assemblieren die Speicherbelegung ausgerechnet wird. Der H3DEF.LST-Datei kann dann entnommen werden, wieviele Speichermedien belegt sind (siehe folgende Tabelle).

Belegte Datenspeicher durch die H3-Software:

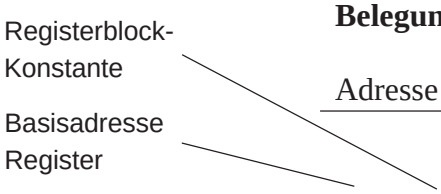
| Symbol | | | Adresse | Kommentar |
|---|---|---|---|---|
| TFl | EQU | F | NBF-BAF | ; Total used Flags<br>; Total belegte Flag |
| TCo | EQU | C | NBC-BAC | ; Total used Counters<br>; Total belegte Zähler |
| TRe | EQU | R | NBR-BAR | ; Total used Registers<br>; Total belegte Register |
| TFB | EQU | | NFB-BAFB | ; Total used Functionblocks<br>; Total belegte Funktionsblöcke |
| NFF | EQU | F | NBF | ; Next free Flag<br>; Nächstes freies Flag |
| NFC | EQU | C | NBC | ; Next free Counter<br>; Nächster freier Zähler |
| NFR | EQU | R | NBR | ; Next free Register<br>; Nächstes freies Register |
| NFFB | EQU | | NFB | ; Next free Functionblock<br>; Nächster freier Funktionsblock |

### 7.1.4 Datentransfer CPU <--> H3-Modul




Die Schnittstelle zwischen dem Anwender und dem H3-Modul bilden zwei Funktionsblöcke und die Achsenspezifischen Daten im Speichermodul PCD4.R2... Die Achsen werden mit den zwei FB "AxInit" und "AxHndlg" programmiert. Im Speichermodul PCD4.R2.. ist für jede Achse eine bestimmte Anzahl Register und Flag zur Speicherung der Parameter belegt. Dieser Speicherbereich wird von den FB gelesen und beschrieben um mit dem H3-Modul Daten auszutauschen. Durch die Angabe der Achsennummer (Ai), der Registerblockkonstante (RAi) und der Flagfeldkonstante (FAi) beim Aufruf der FB werden automatisch die Daten (Register u. Flag)der entsprechenden Achse im Speichermodul PCD4.R2.. und die richtige Achse im H3-Modul adressiert.

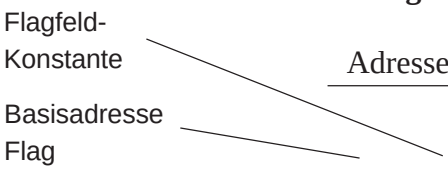
### 7.1.5 Organisation und Zugriff der Achsendaten im Speicher PCD4.R2..




**Belegung der Register**

| Adresse | Symbol | Bezeichnung |
|---|---|---|
| | Achse 1 | |
| BAR+RA1+0 | KProp | Proportionalfaktor |
| BAR+RA1+1 | KInt | Integrationsfaktor |
| BAR+RA1+2 | KDer | Differentialfaktor |
| BAR+RA1+3 | IntL | Integrationsgrenze |
| BAR+RA1+4 | SampI | Abtastzeit D-Anteil |
| BAR+RA1+5 | MCFac | Maschinenfaktor |
| BAR+RA1+6 | PosEr | Positionsfehler |
| BAR+RA1+7 | DestP | Zielposition |
| BAR+RA1+8 | BrkP | Breakposition |
| BAR+RA1+9 | Veloc | Geschwindigkeit |
| BAR+RA1+10 | Accel | Beschleunigung |
| BAR+RA1+11 | StaFRR | Status Flag Reset Register |
| BAR+RA1+12 | MCW | Bewegungs Control Wort |
| BAR+RA1+13 | RActP | Istposition |
| BAR+RA1+14 | RSetP | Sollposition |
| BAR+RA1+15 | RActV | Istgeschwindigkeit |
| BAR+RA1+16 | RSetV | Sollgeschwindigkeit |
| BAR+RA1+17 | RIndP | Indexposition |
| BAR+RA1+18 | RIntTS | Integrationssumme |
| BAR+RA1+19 | RSigB | Signalisationregister |
| BAR+RA2+0 | Achse 2 | |
| BAR+RA2+19 | | |
| BAR+RA3+0 | Achse 3 | |
| BAR+RA3+19 | | |
| BAR+RAn+0 | Achse n | |
| BAR+RAn+19 | | |
| | gemeinsame Register | Werden von den Funktionsblöcken als Zwischenspeicher benützt. |

**Belegung der Flag**




| Adresse | Symbol | Bezeichnung |
|---|---|---|
| | Achse 1 | |
| BAF+FA1+0 | OnDest | Zielpos. erreicht |
| BAF+FA1+1 | IPuls | Indeximpuls erfasst |
| BAF+FA1+2 | WrapOc | Positionsregister Überlauf |
| BAF+FA1+3 | ExcPEr | Positionsfehler |
| BAF+FA1+4 | BrkPos | Breakposition erreicht |
| BAF+FA1+5 | DplM | Anzeigeart auf PCA2.D14 |
| | | Funktionsbefehls-Flag |
| BAF+FA1+6 | FLdDR | Lade Zielposition relativ |
| BAF+FA1+7 | FLdDA | Lade Zielposition absolute |
| BAF+FA1+8 | FLdVR | Lade Geschwindigkeit relativ |
| BAF+FA1+9 | FLdVA | Lade Geschwindigkeit absolute |
| BAF+FA1+34 | FBackw | Rückwärts mit def. Geschwindigkeit |
| BAF+FA2+0 | Achse 2 | |
| BAF+FA2+34 | | |
| BAF+FA3+0 | Achse 3 | |
| BAF+FA3+34 | | |
| BAF+FAn+0 | Achse n | |
| BAF+FAn+34 | | |
| | gemeinsame Flag | Werden von den Funktionsblöcken als Zwischenspeicher benützt. |

**Belegung der Zähler**

Von allen Achsen wird gemeinsam nur ein Zähler als Zwischenspeicher verwendet.

**Zugriff des Anwenderprogramms auf Register und Flag**

Die Adressierung der einzelnen Achsenparameter erfolgt mit symbolischen Namen (definiert in der H3DEF.SRC Datei). Die Parameter haben für alle Achsen den gleichen Namen. Zwischen den einzelnen Achsen kann unterschieden werden, indem zum Symbolname die Anfangsadresse des Parameterblocks für die jeweilige Achse dazu addiert wird. Ist nur der Symbolname angegeben, so sind automatisch die Parameter der Achse 1 adressiert. Die Anfangsadressen der Parameterblöcke sind mit den Konstanten RAi (Registerblock-Konstante für die Achse i) und FAi (Flagfeldkonstante für die Achse i) definiert. Wie nun konkret auf die Parameter der einzelnen Achsen zugegriffen wird, sollen nachfolgende Beispiele verdeutlichen.
In der H3-Software werden die symbolischen Namen wegen der besseren Lesbarkeit mit Gross- und Kleinbuchstaben geschrieben. Da der PCD-Assembler jedoch zwischen Gross- und Kleinschrift nicht unterscheidet , können die Namen im Anwenderprogramm beliebig, gross oder klein geschrieben werden.

Bevor ein Parameter in das H3-Modul geladen werden kann, muss vorgängig das entsprechende Register mit dem gewünschten Wert geladen werden.

Beispiel: Register “DestP” (Zielposition) laden für

| | | | | |
|---|---|---|---|---|
| - Achse 1 | —> | LD | R | DestP+RA1<br>Wert |
| - Achse 3 | —> | LD | R | DestP+RA3<br>Wert |
| - Achse i | —> | LD | R | DestP+RAi<br>Wert |

Um nun diese Zielposition ins H3-Modul zu laden, muss die Funktion “Lade Zielposition” ausgeführt werden. Durch das Setzen des Flags “FLdDA” (Lade Zielposition absolut) wird das Register “DestP” vom Funktionsblock “AxHndlg” gelesen und ins H3-Modul geladen.

Beispiel**:** “Zielposition absolut” ins H3-Modul laden für

- Achse 1 —> SET F FLdDA+FA1
- Achse 3 —> SET F FLdDA+FA3
- Achse n —> SET F FLdDA+FAn

Nachdem eine Bewegung gestartet wurde, soll gewartet werden, bis die Zielposition erreicht ist. Dazu muss das Statusflag “OnDest” abgefragt werden.

Status Flag abfragen mit dem Anwenderprogramm für

- Achse 1 —> STH F OnDest+FA1
- Achse 3 —> STH F OnDest+FA3

## 7.2 Hauptfunktionsblöcke “AxInit” und “AxHndlg”

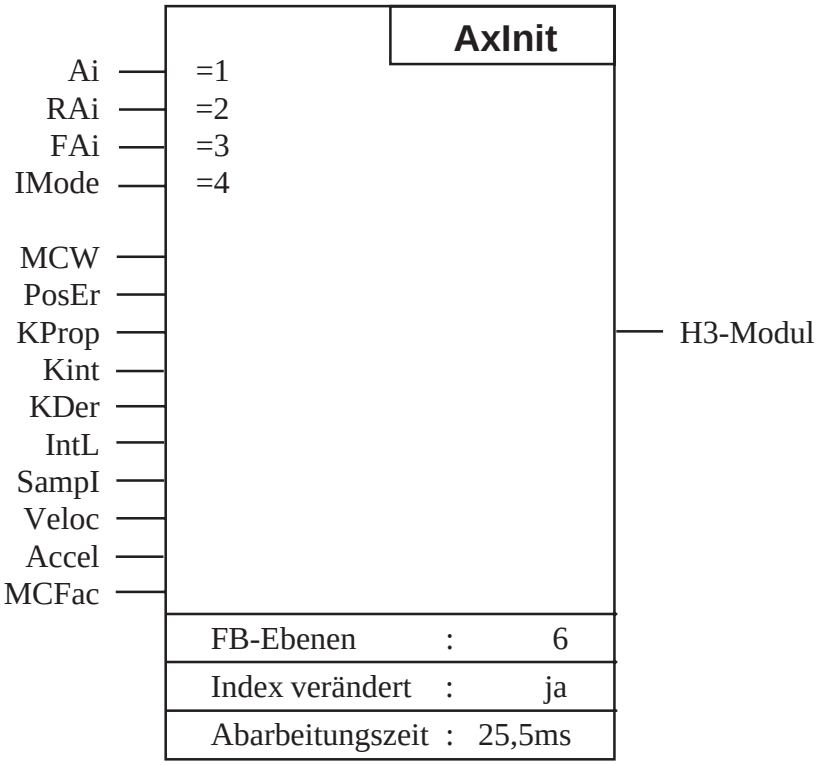
Die Kommunikation mit den H3-Modulen erfolgt ausschliesslich mit den zwei FB “AxInit” (Achsen-initialisierung) und “AxHndlg” (Achsenbearbeitung resp. Parametrierung).
Beim Aufruf dieser FB werden als Parameter die Nummer, Registerblockkonstante und Flagfeldkonstante der zu bearbeitenden Achse übergeben, damit die entsprechenden Daten im Speichermodul der PCD4 und das H3-Modul adressiert wird. Für jede Achse müssen diese zwei FB mit Angabe obengenannter Parameter aufgerufen werden.

### AxInit

Funktionsblock: - **Ax**is **Init**ialisation
- Achsen Initialisierung

**AxInit**

Softwarepaket: PCD9.H3E1




**Funktionsbeschreibung:**

Mit diesem FB wird im H3-Modul eine Achse initialisiert. Bevor mit einer Achse gearbeitet werden kann, muss zuerst dieser FB einmal abgearbeitet werden. Er wird vorteilhaft im XOB 16 aufgerufen, da er so nur einmal abgearbeitet wird. Bei der Ausführung dieses FB werden automatisch eine Reihe von Funktionen ausgeführt. Ein Teil der Funktionen sind die gleichen wie sie auch mit dem Achsenbehandlungs FB “AxHndlg” ausgeführt werden können und sind deshalb an dieser Stelle nicht detalliert beschrieben.

Folgende Funktionen werden ausgeführt:

**1. Reset des Controllers im H3-Modul**

Dabei werden alle Bewegungs- (Beschleunigung, Geschwindigkeit, Zielposition und Breakposition) und PID-Parameter (inkl. Stellgrösse) auf Null gesetzt. Ebenfalls wird die Istposition als Nullposition definiert.

**2. Stellgrössenausgangsport-Initialisierung**

Das Ausgangsport wird entsprechend dem Parameter “IMode” initialisiert. Die Konstante “IMode” muss in der H3DEF.SRC Datei entsprechend dem eingesetzten H3-Modul definiert sein.

**3. Wahl der Betriebsart** —> siehe Funktion “FSelOM”

**4. Laden des Positionsfehlers** —> siehe Funktion “FSetPEr”

**5. Regelparameter laden** —> siehe Funktion “FLdRP”

**6. Regelparameter updaten** —> siehe Funktion “FUpDRP”

**7. Beschleunigung laden** —> siehe Funktion “FLdAA”

**8. Geschwindigkeit laden** —> siehe Funktion “FLdVA”

**9. Reset aller Funktions-Befehlsflag** —> siehe FB “AxHndlg”

Beispiel: Aufruf des Funktionsblockes für Achse 3

```
CFB     AxInit
        3           ; Achsennummer
        RA3         ; Registerblock Achse 3
        FA3         ; Flagfeld Achse 3
        IMode       ; Initialisation Mode
```

**Beschreibung der Ein- und Ausgänge:**

Falls nichts anderes angegeben, sind alle verwendeten symbolischen Namen in der H3DEF.SRC-Datei definiert und müssen unverändert übernommen werden.

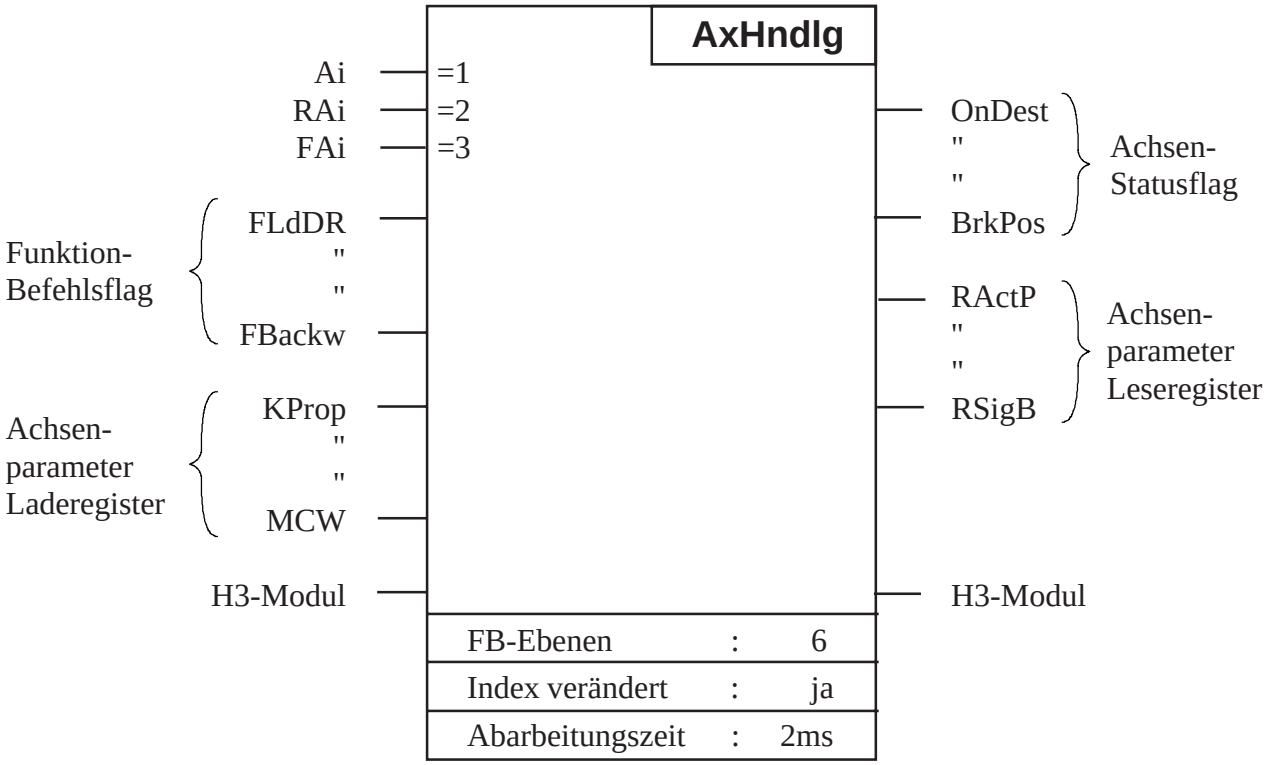
| Symbol | Bezeichnung/Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Ai | **A**chsennummer i<br>Ai ist nicht als Symbol in der H3DEF.SRC-Datei definiert , es kann direkt eine Konstante eingeben werden. | ja | K | Integer | 1...32 |
| RAi | **R**egisterblockkonstante **A**chse i | ja | K | Integer | (i-1) * 20 |
| FAi | **F**lagfeldkonstante **A**chse i | ja | K | Integer | (i-1) * 35 |
| IMode | **I**nitialisation **Mode**<br>Stellgrössenausgang Initialisierung | ja | K | Integer | 5H/6H |
| MCW | **M**otion **C**ontrol **W**ord<br>Bewegungs Control Wort | nein | R | Binär | - |
| PosEr | **Pos**ition **Er**ror<br>Positionsfehler<br>Einheit: Impulse | nein | R | Integer | 0...32767 |
| KProp | **Prop**ortional factor | nein | R | Integer | 0...32767 |
| KInt | **Int**egral Factor | nein | R | Integer | 0...32767 |
| KDer | **Der**ivative Factor<br>Differentialfaktor | nein | R | Integer | 0...32767 |
| IntL | **Int**egration **L**imit<br>Integrationsgrenze | nein | R | Integer | 0...32767 |
| SampI | **Samp**ling **I**nterval<br>Abtastzeit | nein | R | Integer | 0...255 |
| Veloc | **Veloc**ity<br>Geschwindigkeit | nein | R | Integer | siehe Funktion “FLdVA” |
| Accel | **Accel**eration<br>Beschleunigung | nein | R | Integer | siehe Funktion “FLdAA” |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fliess-punkt | siehe Funktion “FLdDA” |

# AxHndlg

Funktionsblock: - **Ax**is **H**a**ndl**i**ng**
- Achsen Bearbeitung

AxHndlg

Softwarepaket: PCD9.H3E1




**Funktionsbeschreibung:**

Nachdem eine Achse mit dem FB "AxInit" initialisiert wurde , erfolgt die Kommunikation mit dem H3-Modul ausschliesslich mit diesem Funktionsblock. Die Achsenparameter können ins H3-Modul geladen, sowie von demselben gelesen werden. Auch Bewegungsbefehle werden dem Modul über diesen FB erteilt. Die Ein- und Ausgänge des FB lassen sich in folgende Gruppen aufteilen:

Eingänge
- FB-Parameter
- Funktions-Befehlsflag
- Achsenparameter Laderegister
- H3-Modul

Ausgänge
- Achsen Statusflag
- Achsenparameter Leseregister
- H3-Modul

**FB-Parameter**

Durch die Übergabe von Achsennummer (Ai), Registerblockkonstante (RAi) und Flagfeldkonstante (FAi) beim Aufruf des FB, wird automatisch die richtige Achse im H3-Modul und die zugehörigen Achsendaten (Befehlsflag, Statusflag, Laderegister und Leseregister) im Speichermodul PCD4.R2.. adressiert.

**Funktions-Befehlsflag**

Durch setzen eines Befehlsflag im Anwenderprogramm kann eine bestimmte Funktion aktiviert werden.
(Z.B. laden einer Zielposition, Starten einer Bewegung usw.) Bei der Abarbeitung des FB werden alle Funktions-Befehlsflag abgefragt. Ist ein Flag gesetzt, so führt der FB unmittelbar nach dessen Abfrage eine Funktion aus. Diese wird realisiert indem ein Subfunktionsblock (hat den gleichen Namen wie das Befehlsflag, jedoch ohne vorangestellten Buchstabe F) aufgerufen wird. Nachdem diese Funktion ausgeführt ist, wird das Flag wieder zurückgesetzt. Jedes Flag ist einer Funktion zugeordnet. Ist kein Flag gesetzt, so wird der FB verlassen, ohne dass eine Funktion ausgeführt wurde.
Diese Befehlsflag sind für jede Achse separat im Speichermodul PCD4.R2.. abgelegt. Wie die Flag vom Anwender gesetzt werden, siehe vorangehendes Kapitel 7.1.5 .
Damit nach dem Aufstarten der PCD4 keine unbeabsichtigten Funktionen ausgeführt werden, werden die Befehlsflag mit dem Achseninitialisierungs-Funktionsblock “AxInit” automatisch zurückgesetzt.
Beschreibung der einzelnen Funktionen siehe Kapitel 7.4

**Achsenparameter Laderegister**

In den Laderegistern sind alle nötigen Parameter für den Betrieb einer Achse abgelegt. Die Register müssen vom Anwender mit den Werten für die Parameter geladen werden. Soll ein Parameter geladen werden, so muss die entsprechende Funktion aktiviert werden, welche den Wert vom Laderegister ins H3-Modul kopiert.

**Achsen Statusflag**

Bei jeder Abarbeitung des FB werden automatisch die Statusflag der Achse gelesen und in diese Flag kopiert. Der Status der Achse kann über diese Flag mit dem Anwenderprogramm abgefragt werden. Für die Bedeutung der einzelnen Flag siehe Beschreibung der Funktion “FResSF”.

**Achsenparameter Leseregister**

Die Achsenparameter, die vom H3-Modul gelesen werden können, werden in die Leseregister kopiert.

**H3-Modul**

Steht symbolisch für den Achsencontroller im H3-Modul.

**Aufruf des Funktionsblockes:**

Aufruf des FB für die Achse 2

```
CFB     AxHndlg
        2           ; Achsennummer
        RA2         ; Registerblock Achse 2
        FA2         ; Flagfeld Achse 2
```

## Programmstruktur und Anwendung der Funktionsblöcke "AxInit" und AxHndlg"

Das Anwenderprogramm für das H3-Modul lässt sich grob in 3 Teile aufteilen:

- Initialisierung der Achsen

- Zyklische Bearbeitung der Achsen

- Definition des Fahrprogrammes (Bewegungsablauf der Maschine)

**a) Initialisierung der Achsen**

Der erste Schritt in einem H3-Programm ist immer die Initialisierung der Achsen. Durch den Aufruf des FB "AxInit" im XOB 16 wird eine Achse initialisiert. Bei der Abarbeitung des FB werden mehrere Register gelesen, die vorgängig geladen werden müssen. Um die Werte für die Register zu bestimmen, müssen bereits verschiedene Daten des Antriebes bekannt sein (Siehe auch Kapitel 9. Anwendungsbeispiele). Nach der Ausführung des FB "AxInit" ist die Achse betriebsbereit.

**b) Zyklische Bearbeitung der Achsen**

Zur zyklischen Bearbeitung gehören all jene Aufgaben, die regelmässig ausgeführt werden. Deshalb werden diese Aufgaben in einem COB, der zyklisch bearbeitet wird, programmiert. Der Funktionsblock "AxHndlg" erledigt den gesamten Datenaustausch mit dem Controller auf dem H3-Modul. Mit den Funktions-Befehlsflag wird dem FB mitgeteilt, welche Aufgaben er auszuführen hat. Die Aufgabe des FB besteht also darin, laufend diese Befehlsflag abzufragen und die ihm erteilten Befehle auszuführen. Es ist deshalb naheliegend, den Funktionsblock in einem COB zu programmieren.

**c) Definition des Fahrprogrammes**

Da das Fahrprogramm eines Antriebes immer einem sequentiellen Ablauf entspricht, wird dieses in der bestens dazu geeigneten GRAF-TEC-Struktur programmiert.

Der prinzipielle Ablauf ist immer gleich und beinhaltet folgende Schritte:

- es wird eine Bewegung definiert und gestartet (--> Step)
- es wird gewartet bis die Achse die Bewegung vollendet hat (--> Transition)
- nächste Bewegung
- warten
- usw.

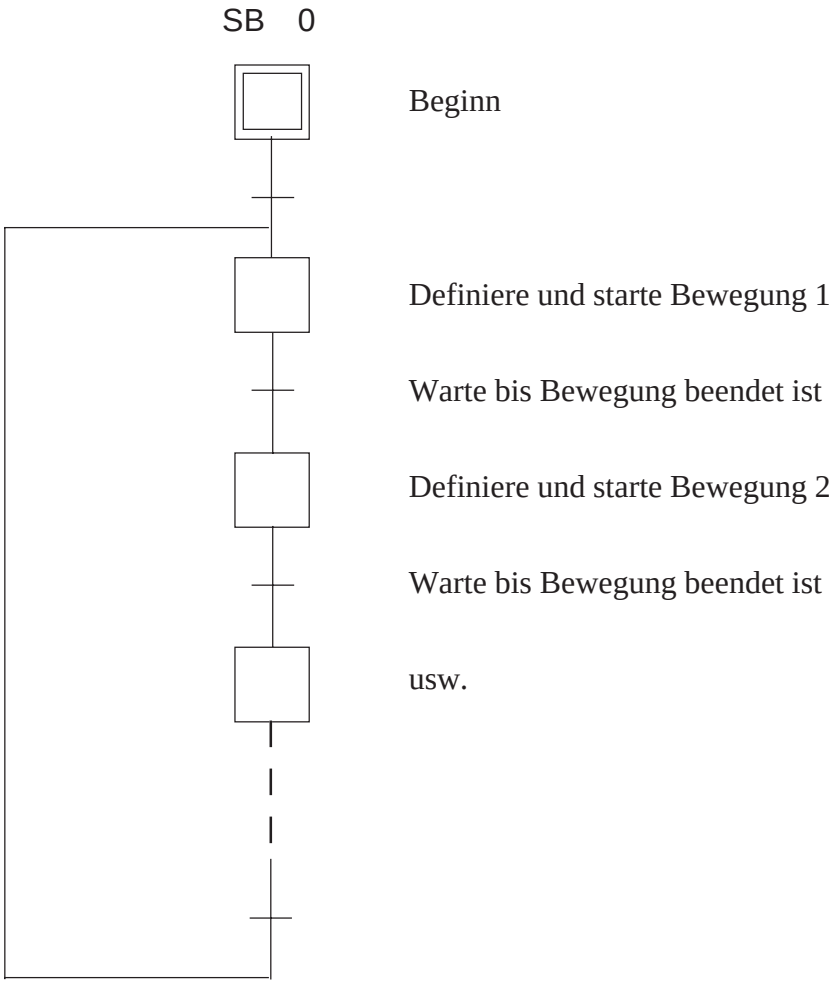
Bei jeder nichterfüllten Transition wird das GRAFTEC-Programm verlassen und mit der zyklischen Programmbearbeitung weitergefahren.

--> Der FB "AxHndlg" wird aufgerufen und führt die ihm, in der GRAFTEC-Struktur, erteilten Aufgaben aus.
(Bewegungsparameter ins H3-Modul laden und die Bewegung starten)

Grobstruktur eines H3-Programmes mit den FB-Aufrufen

```
XOB             16
Initialisierungsregister laden
CFB             AxInit          ; Initialisierung
                1               ; der Achse 1
                RA1
                FA1
                IMode
 :
 :
 :
CFB             AxInit          ; Initialisierung
                n               ; der Achse n
                RAn
                FAn
                IMode
EXOB
```

```
COB             0
CFB             AxHndlg         ; Bearbeitung
                1               ; der Achse 1
                RA1
                FA1
 :
 :
 :
CFB             AxHndlg         ; Bearbeitung
                n               ; der Achse n
                RAn
                FAn

CSB             0               ; Aufruf des
                                ; Fahrprogrammes

Andere zyklische Aufgaben
wie Überwachung u. Anzeige

ECOB
```



## 7.3. Übersicht der Funktionen

Folgende Funktionen können mit dem Achsenbearbeitungs-FB “AxHndlg” ausgeführt werden. Der angegebene Name bezeichnet das Flag, mit welchem die Funktion aktiviert wird.

**Betriebsdefinitionen:**

| | |
|---|---|
| **FSelOM** | Wahl der Betriebsart |
| **FSetPE** | Definition des Positionsfehlers |

**Parametereingaben für das Geschwindigkeitsprofil**

| | |
|---|---|
| **FLdDA** | Lade Zielposition absolut |
| **FLdDR** | Lade Zielposition relativ |
| **FLdVA** | Lade Geschwindigkeit absolut |
| **FLdVR** | Lade Geschwindigkeit relativ |
| **FLdAA** | Lade Beschleunigung absolut |
| **FLdAR** | Lade Beschleunigung relativ |
| **FLdRP** | Lade Regelparameter |
| **FUpDRP** | Aktiviere geladene Regelparameter |

**Fahrbefehle**

| | |
|---|---|
| **FStart** | Starte Bewegung oder übernehme neu geladene Parameter |
| **FStop** | Stoppe Bewegung |
| **FMotOff** | Regelung ausschalten |
| **FSStepV** | Einzelschritt vorwärts |
| **FSStepB** | Einzelschritt rückwärts |
| **FForw** | Vorärts mit definierter Geschwindigkeit |
| **FBackw** | Rückwärts mit definierter Geschwindigkeit |

**Lesebefehle für Daten**

| | |
|---|---|
| **FRdAP** | Lese Istposition |
| **FRdSP** | Lese Sollposition |
| **FRdAV** | Lese Istgeschwindigkeit |
| **FRdSV** | Lese Sollgeschwindigkeit |
| **FRdITS** | Lese Integrationssumme |
| **FRdIP** | Lese Indexposition |
| **FRdSR** | Lese Signalregister |

**Hilfsfunktionen**

| | |
|---|---|
| **FResSF** | Reset Statusflag |
| **FLdBPA** | Lade Breakposition absolut |
| **FLdBPR** | Lade Breakposition relativ |
| **FSetIP** | Erfasse Indexposition |
| **FSetZP** | Setze Nullposition |

Notizen:

## 7.4 Beschreibung der Funktionen

Um eine gute Übersicht zu bewahren, sind alle folgenden Funktionen nach dem gleichen Gestaltungsschema aufgebaut:


Es wird im folgenden die gleiche Reihenfolge benützt wie in der Übersicht Abschnitt 7.3.
Am Schluss dieses Handbuches finden Sie zudem eine alphabetische Auflistung aller hier benützten Namen und Symbole.

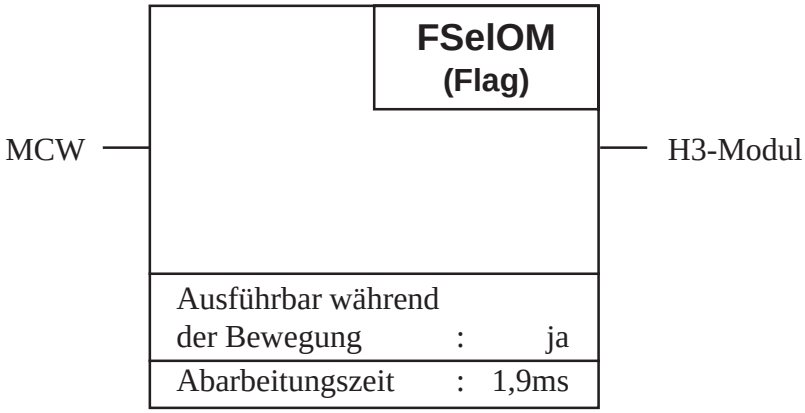
*) Zeigt an, dass der Name für ein Flag steht und die Funktion durch Setzen dieses Flag ausgeführt werden kann.

# FSelOM

Funktion : - **Sel**ect **O**peration **M**ode
- Wahl der Betriebsart

FSelOM

Softwarepaket : PCD9.H3E1



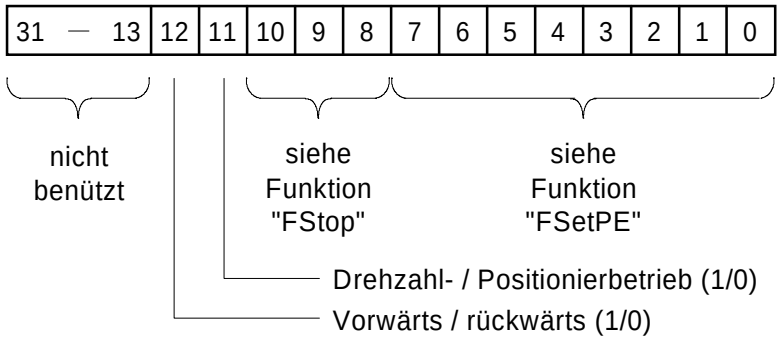

**Funktionsbeschreibung:**

Mit dieser Funktion wird die Betriebsart der Achse definiert (Positionier- oder Drehzahlbetrieb). Die neu definierte Betriebsart ist erst nach Ausführen des nächsten Startbefehls "FStart" gültig.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| MCW | Motion Control Word<br>Bewegungs Control Wort | nein | R | Binär | siehe folgende Seite |

**Bedeutung des Bewegungs Control Wortes "MCW":**




Die Funktion liest nur Bit 11 und 12 des Registers "MCW"

Bit 11: 0 --> Positionierbetrieb
1 --> Drehzahlbetrieb

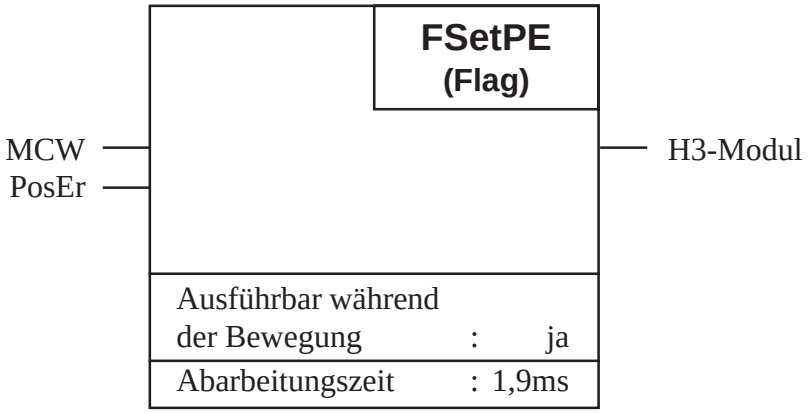
Bit 12: 0 --> rückwärts
1 --> vorwärts

# FSetPE

Funktion : - **S**et **P**osition **E**rror
- Definiere Positionsfehler

FSetPE

Softwarepaket : PCD9.H3E1



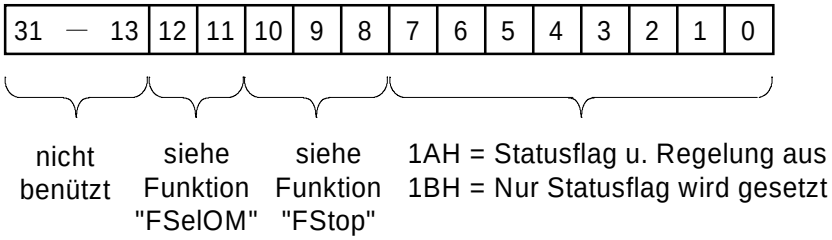

**Funktionsbeschreibung:**

Mit dieser Funktion wird die maximale Differenz (wird auch Schleppabstand genannt) zwischen der Soll- und Istposition festgelegt. Erreicht die Differenz (Betrag) den mit dieser Funktion definierten Wert, so wird das Statusflag "ExcPEr" gesetzt. Im Register "MCW" wird angegeben ob im Fehlerfall nur das Statusflag gesetzt, oder zusätzlich auch die Regelung ausgeschaltet wird (Stellgrössenausgang gleich Null).
Ein Positionsfehler signalisiert ernsthafte Probleme und kann so überwacht werden.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| PosEr | **Pos**ition **Er**ror Positionsfehler | nein | R | Integer | 0.. 32'767 Imp. |
| MCW | **M**otion **C**ontrol **W**ord Bewegungs Control Wort | nein | R | Binär | siehe folgende Seite |

**Bedeutung des Bewegungs Control Wortes "MCW":**




Die Differenz zwischen Soll- und Istposition wird direkt in Encoderimpulsen eingegeben. Es ist zu beachten, dass im Positionsdecoder eine Impulsvervierfachung (durch Auswertung der Flanken) der Encoderimpulse erzeugt wird.

Beispiel:

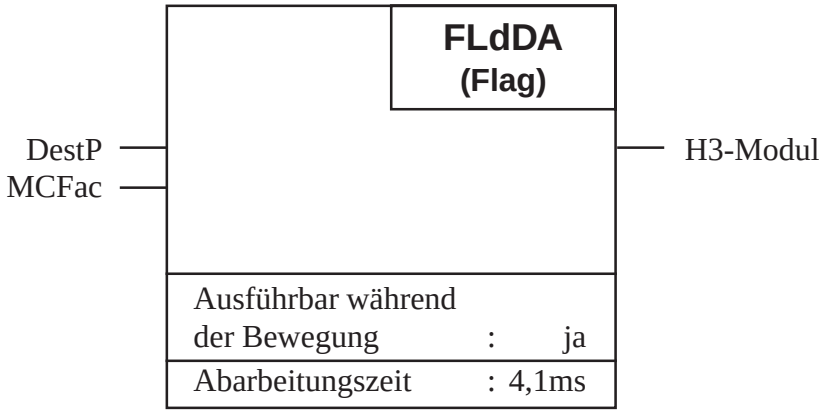
Soll die Differenz maximal 500 Encoderimpulse betragen, so muss in das Register "PosEr" der Wert 4*500 = 2'000 Impulse geladen werden.

# FLdDA

Funktion : - **L**oa**d** **D**estination **A**bsolute
- Lade Zielposition absolut

FLdDA

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Zielposition absolut in ein Zwischenregister auf dem H3-Modul geladen. Absolut laden heisst, der Wert ist bezogen auf die Nullposition. Die neue geladene Position wird vom H3-Modul erst beim nächsten Startbefehl "FStart" ins Arbeitsregister übernommen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| DestP | **Dest**ination **P**osition<br>Zielposition<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

Mit diesem Faktor k wird die Einheit für die Eingabe einer Zielposition, Geschwindigkeit und Beschleunigung bestimmt. Der Faktor errechnet sich aus der Encoderauflösung und der Übersetzung der Mechanik. Der Faktor k muss berechnet und in das Register "MCFac" geladen werden. Das Register wird von mehreren Funktionen zur Umrechnung eines metrischen Masses in Encoderimpulse und umgekehrt gelesen.

Es gilt: $k = \frac{4*In}{s}$

wobei In: Impulse / Umdrehung (Encoderauflösung)
s : Weg / Umdrehung (Spindelsteigung und Getriebe)

Mit der Einheit für den Weg wird zugleich auch die Einheit für die Eingabe einer Position, Geschwindigkeit und Beschleunigung festgelegt.

Beispiel:

Spindel mit 3mm Steigung
Encoderauflösung 1000 Imp. / Umdr.

Es soll eine Zielposition von 60mm angefahren werden und die Eingabe (respektive Auflösung) der Position soll in µm erfolgen.

$$k = \frac{4*In}{s} = \frac{4*1000\ \text{Imp./Umdr.}}{3000\ \mu\text{m/Umdr.}} = 1{,}33333\ \text{Imp./}\mu\text{m}$$

Eingaberegister "DestP" = 60000µm

Für obiges Beispiel soll die Position in einer Einheit von 1/10 mm eingegeben werden:

$$k = \frac{4*In}{s} = \frac{4*1000\ \text{Imp./Umdr.}}{30\ 1/10\text{mm / Umdr.}} = 133{,}333\ \text{Imp./ 1/10mm}$$

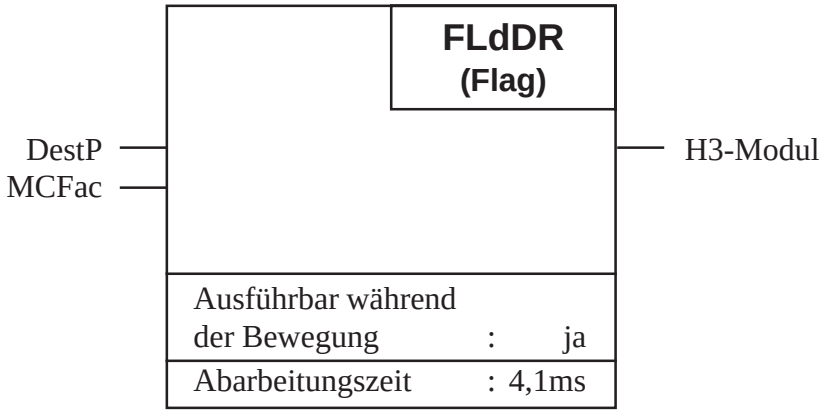
Eingaberegister "DestP" = 600 1/10mm

# FLdDR

Funktion : - **L**oa**d** **D**estination **R**elative
- Lade Zielposition relativ

FLdDR

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Zielposition relativ in ein Zwischenregister auf dem H3-Modul geladen. Relativ laden heisst, der Wert ist bezogen auf die momentane Zielposition. Die neue geladene Position wird vom H3-Modul erst beim nächsten Startbefehl "FStart" ins Arbeitsregister übernommen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| DestP | **Dest**ination **P**osition<br>Zielposition<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

Nach der Initialisierung des Moduls darf für die erste Bewegung der Achse nur eine absolute Zielposition geladen werden. Wird eine relative Position geladen, so erzeugt der H3-Controller einen "Command Error".

**Maschinenfaktor k in Register "MCFac":**

Der Faktor hat die gleiche Bedeutung wie bei der Funktion "FLdDA":

Beispiel:

Spindel mit 3mm Steigung
Encoderauflösung 1000 Imp. / Umdr.

Es soll relativ eine Strecke von -60mm gefahren werden und die Eingabe (respektive Auflösung) der Position soll in µm erfolgen.

$$k = \frac{4*In}{s} = \frac{4*1000\ \text{Imp./Umdr.}}{3000\ \mu\text{m/Umdr.}} = 1{,}33333\ \text{Imp./}\mu\text{m}$$

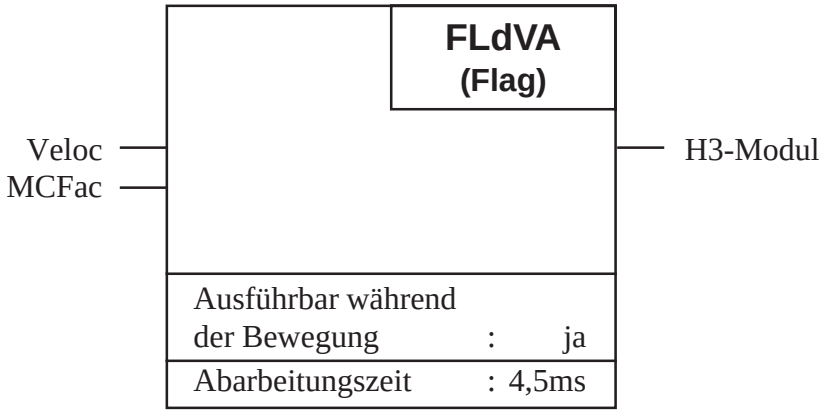
Eingaberegister "DestP" = -60000 µm

# FLdVA

Funktion : - **L**oa**d** **V**elocity **A**bsolute
- Lade Geschwindigkeit absolut

FLdVA

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Geschwindigkeit absolut in ein Zwischenregister auf dem H3-Modul geladen. Absolut laden heisst, der Wert ist bezogen auf Null. Die neue geladene Geschwindigkeit wird vom H3-Modul erst beim nächsten Startbefehl ins Arbeitsregister übernommen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Veloc | **Veloc**ity<br>Geschwindigkeit<br><br>Wert:<br>$[0..+(2^{30}-1)]/k*22348*10^{-6}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac" :**

Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Beachte, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

Beispiel:

Spindel mit 3mm Steigung
Encoderauflösung 1000 Imp. / Umdr.

Es soll eine Zielposition mit einer Geschwindigkeit von 0,1m/s angefahren werden und die Eingabe (respektive Auflösung) soll in mm/s erfolgen.

$$k = \frac{4*In}{s} = \frac{4*1000\ \text{Imp./Umdr.}}{3\ \text{mm/Umdr.}} = 1333{,}3\ \text{Imp./mm}$$

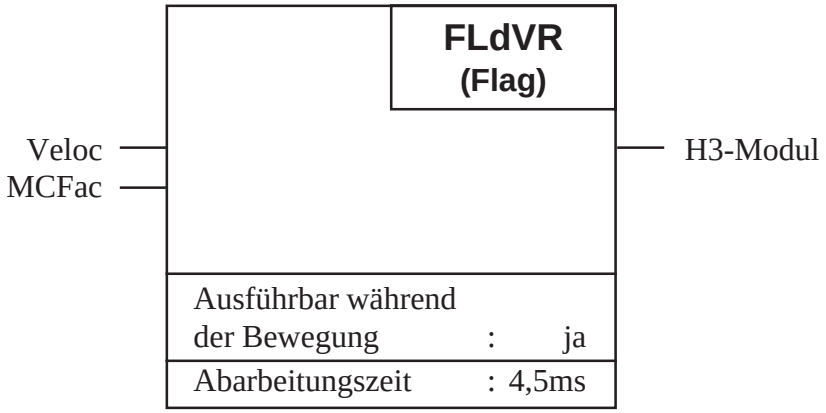
Eingaberegister "Veloc" = 100 mm/s

# FLdVR

Funktion : - **L**oa**d** **V**elocity **R**elative
- Lade Geschwindigkeit relativ

FLdVR

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Geschwindigkeit relativ in ein Zwischenregister auf dem H3-Modul geladen. Relativ laden heisst, der Wert ist bezogen auf die momentane Sollgeschwindigkeit. Die neue geladene Geschwindigkeit wird vom H3-Modul erst beim nächsten Startbefehl ins Arbeitsregister übernommen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Veloc | **Veloc**ity<br>Geschwindigkeit<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*22348*10^{-6}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

Nach der Initialisierung des Modules darf für die erste Bewegung der Achse nur eine absolute Geschwindigkeit geladen werden. Wird eine relative Geschwindigkeit geladen, so erzeugt der H3-Controller einen "Command Error".

**Maschinenfaktor k in Register "MCFac" :**

Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Beachte, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

Beispiel:

Spindel mit 3mm Steigung
Encoderauflösung 1000 Imp. / Umdr.

Es soll eine Geschwindigkeit relativ von -0,1m/s geladen werden und die Eingabe (respektive Auflösung) soll in mm/s erfolgen.

$$k = \frac{4*In}{s} = \frac{4*1000\ \text{Imp./Umdr.}}{3\ \text{mm/Umdr.}} = 1333{,}3\ \text{Imp./mm}$$

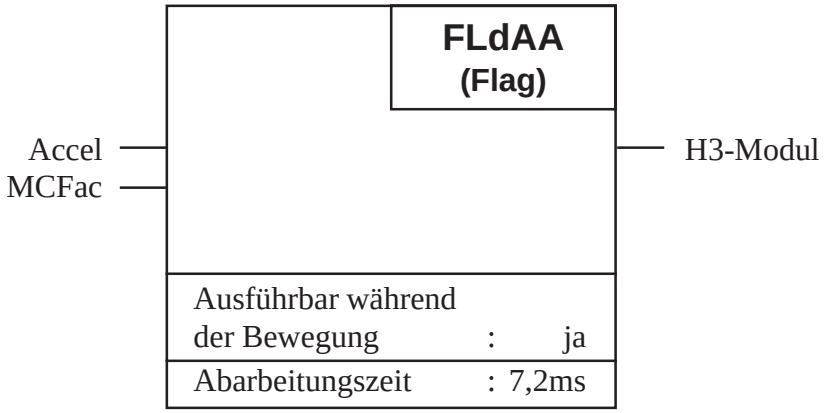
Eingaberegister "Veloc" = -100 mm/s

# FLdAA

Funktion : - **L**oa**d** **A**cceleration **A**bsolute
- Lade Beschleunigung absolut

FLdAA

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Beschleunigung absolut in ein Zwischenregister auf dem H3-Modul geladen. Absolut laden heisst, der Wert ist bezogen auf Null. Die neue geladene Beschleunigung wird vom H3-Modul erst beim nächsten Startbefehl ins Arbeitsregister übernommen. Die Funktion darf zwar während einer Bewegung ausgeführt werden, jedoch ein Startbefehl "FStart", welcher den H3-Controller veranlasst mit der neu geladenen Beschleunigung zu arbeiten, darf erst nach einer vollendeten Bewegung (kann auch durch einen Stopbefehl erfolgen) ausgeführt werden.

Beachte : Beim Aufruf dieser Funktion wird der Befehl "FMotOff" ausgeführt bevor die Beschleunigung geladen wird.
--> Die Regelung ist ausgeschaltet nachdem diese Funktion ausgeführt ist.
--> Regelung wieder einschalten mit "FStart".

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Accel | **Accel**eration<br>Beschleunigung<br><br>Wert:<br>$[0..+(2^{30}-1)]/k*76206*10^{-9}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac" :**

Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Beachte, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

Beispiel:

Spindel mit 3mm Steigung
Encoderauflösung 1000 Imp. / Umdr.

Es soll mit einer Beschleunigung von 0,005 $m/s^2$ beschleunigt werden und die Eingabe (respektive Auflösung) soll in $mm/s^2$ erfolgen.

$$k = \frac{4*In}{s} = \frac{4*1000\ Imp./Umdr.}{3\ mm/Umdr.} = 1333{,}3\ Imp./mm$$

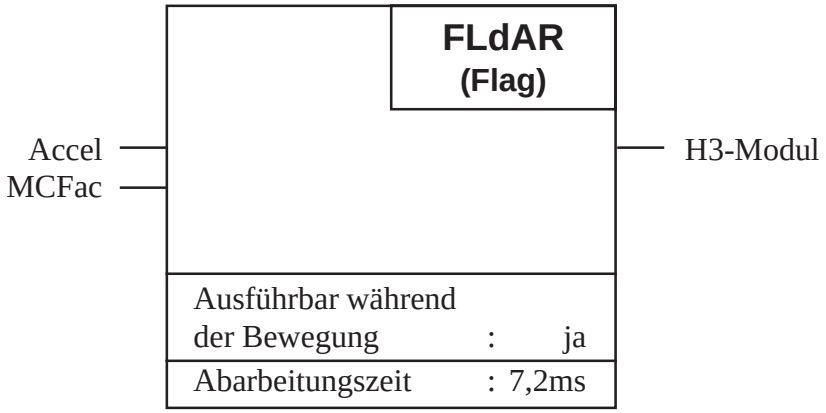
Eingaberegister "Accel" = 5 $mm/s^2$

# FLdAR

Funktion : - **L**oa**d** **A**cceleration **R**elative
- Lade Beschleunigung relativ

FLdAR

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Beschleunigung relativ in ein Zwischenregister auf dem H3-Modul geladen. Relativ laden heisst, der Wert ist bezogen auf die momentane Sollbeschleunigung. Die neue geladene Beschleunigung wird vom H3-Modul erst beim nächsten Startbefehl ins Arbeitsregister übernommen. Die Funktion darf zwar während einer Bewegung ausgeführt werden, jedoch ein Startbefehl "FStart", welcher den H3-Controller veranlasst mit der neu geladenen Beschleunigung zu arbeiten, darf erst nach einer vollendeten Bewegung (kann auch durch einen Stopbefehl erfolgen) ausgeführt werden.
Beachte : Beim Aufruf dieser Funktion wird der Befehl "FMotOff" ausgeführt bevor die Beschleunigung geladen wird.
--> Die Regelung ist ausgeschaltet nachdem diese Funktion ausgeführt ist.
--> Regelung wieder einschalten mit "FStart".

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Parameter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Accel | **Accel**eration<br>Beschleunigung<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*76206*10^{-9}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

Nach der Initialisierung des Modules darf für die erste Bewegung der Achse nur eine absolute Beschleunigung geladen werden. Wird eine relative Beschleunigung geladen, so erzeugt der H3-Controller einen "Command Error".

**Maschinenfaktor k in Register "MCFac" :**

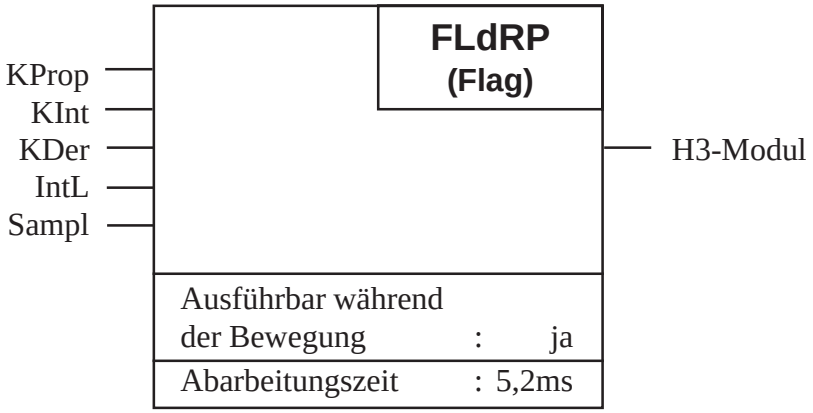
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Beachte, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

# FLdRP

Funktion : - **L**oad **R**egulator **P**arameter
- Lade Regelparameter

FLdRP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion werden die Regelparameter aus den Achsenregistern in Zwischenregister auf dem H3-Modul geladen. Der Regler übernimmt die Werte in die Arbeitsregister nachdem die Funktion "FUpDRP" ausgeführt wurde.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| KProp | **Prop**ortional Factor<br>Proportionalfaktor | nein | R | Integer | 0.. 32'767 |
| KInt | **Int**egral Factor<br>Integralfaktor | nein | R | Integer | 0.. 32'767 |
| KDer | **Der**ivative Factor<br>Differentialfaktor | nein | R | Integer | 0.. 32'767 |
| IntL | **Int**egration **L**imit<br>Integrationsgrenze | nein | R | Integer | 0.. 32'767 |
| Sampl | **Sampl**ing Interval<br>derivative term<br>Abtastzeit Differentialteil | nein | R | Integer | 0.. 255 |

Die Abtastzeit vom Differentialteil kann in Schritten von 341,33µs programmiert werden.

Abtastzeit = (n+1) * 341,33µs

Die Zahl n wird in das Register "Sampl" geladen.

Beispiel: Die Abtastzeit soll 1024µs betragen.

--> Register "Sampl" = 2

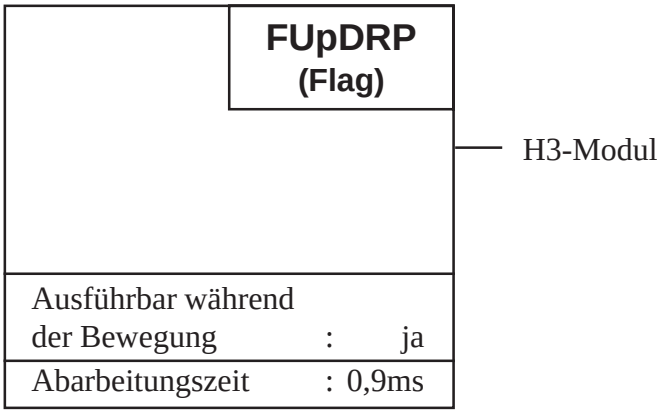
Die Abtastzeit für den Porportional- und Integralteil beträgt 341,33µs und kann nicht programmiert werden.
Wird ein Wert ausserhalb des erlaubten Wertebereichs in die Register geladen, so wird beim Ausführen der Funktion ein Befehls-Error erzeugt.

# FUpDRP

Funktion : - **Up D**ate **R**egulator **P**arameter
- Lade Regelparameter

FUpDRP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

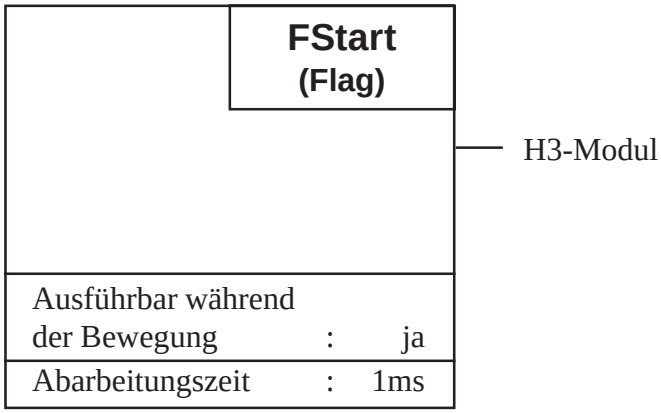
Mit dieser Funktion wird ein Update der Regelparameter ausgeführt. Der Regler im H3-Modul übernimmt die mit der Funktion "FLdRP" geladenen Parameter von den Zwischenregistern in die Arbeitsregister.

# FStart

Funktion : - **Start** motion
- Starte Bewegung

FStart

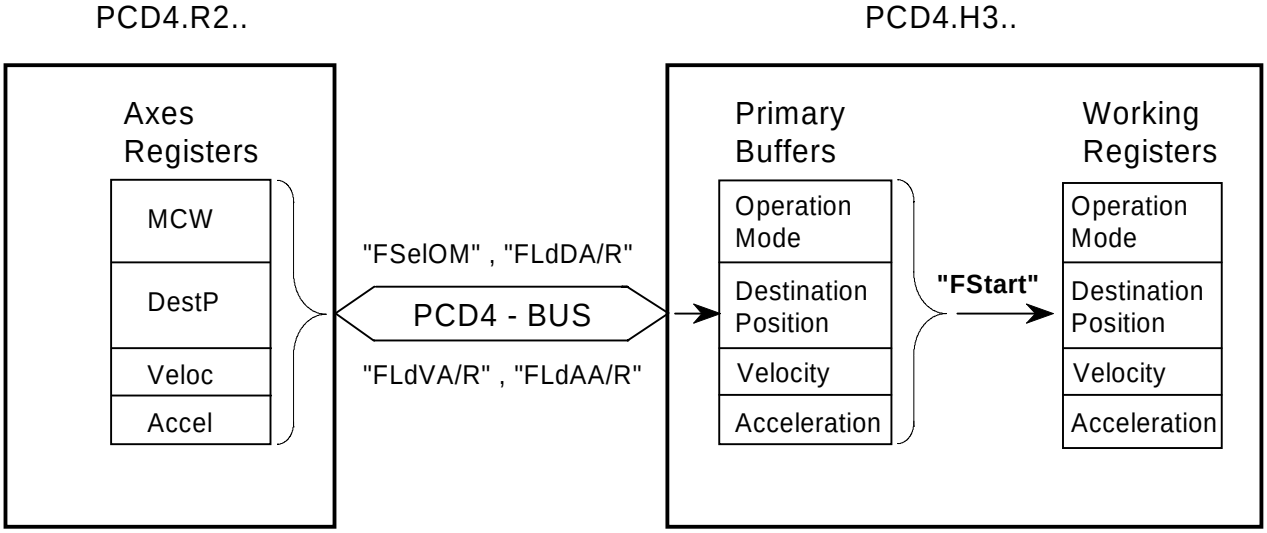
Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion kann eine Bewegung gestartet, oder der Controller im H3-Modul veranlasst werden mit einem neu geladenen Bewegungsparameter (z.B. eine Geschwindigkeit) zu arbeiten.
Wird während der Bewegung eine neue Beschleunigung geladen, so darf ein Startbefehl erst ausgeführt werden, nachdem die Bewegung abgeschlossen ist.

Untenestehendes Diagramm zeigt, welche Bewegungsparameter erst nach einem Startbefehl vom H3-Controller in die Arbeitsregister übernommen werden.

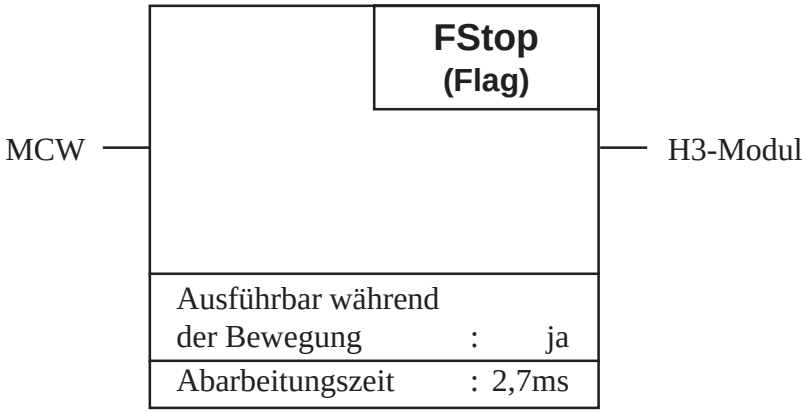


# FStop

Funktion : - **Stop** motion
- Stoppe Bewegung

FStop

Softwarepaket : PCD9.H3E1



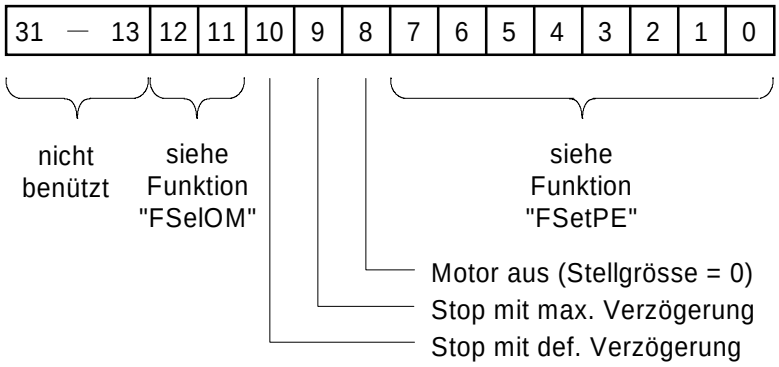

**Funktionsbeschreibung:**

Mit dieser Funktion kann eine Bewegung zu einem beliebigen Zeitpunkt gestoppt werden. Der Stop erfolgt mit der in Register "MCW" definierten Stopart. Das Statusflag "OnDest" wird nach einem ausgeführten Stop gesetzt.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| MCW | **M**otion **C**ontrol **W**ord<br>Bewegungs Control Wort | nein | R | Binär | siehe folgende Seite |

**Bedeutung des Bewegungs Control Wortes "MCW":**




Die Funktion liest nur Bit 8 bis 10 vom Register "MCW"

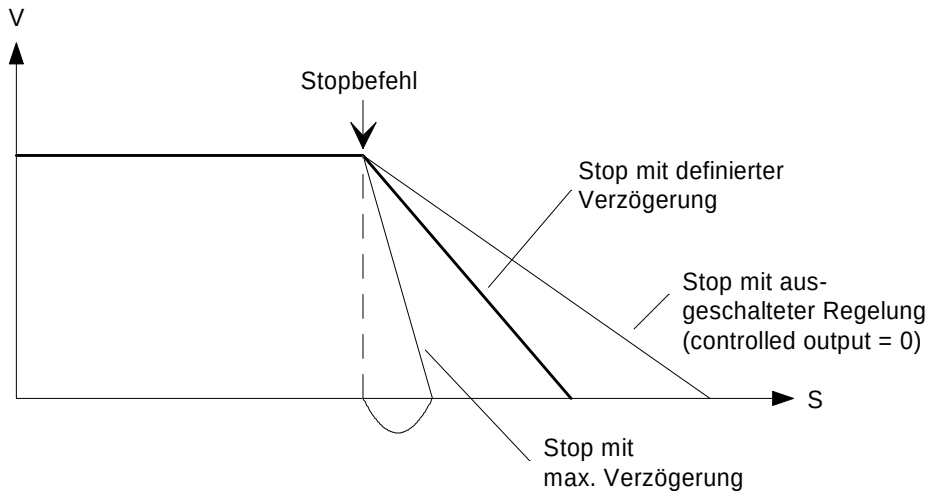
Bit 8 = "1" Bei einem Stopbefehl wird die Regelung ausgeschaltet. D.h. die Stellgrösse wird auf Null gesetzt.

Bit 9 = "1" Bei einem Stopbefehl wird mit der maximalen Verzögerung gebremst, indem im Arbeitsregister des H3-Controllers die Sollposition gleich der Istposition gesetzt wird.

Bit 10 = "1" Bei einem Stopbefehl wird mit der definierten Verzögerung (=negative Beschleunigung) gebremst.

Gleichzeitig darf immer nur eines der 3 Bit aktiv ("1") sein.
Nach einem Stop verliert der Controller die zuletzt geladene Zielposition nicht. Bevor allerdings die unterbrochene Bewegung, ohne einen neuen Parameter (Zielposition, Geschwindigkeit oder Beschleunigung) zu laden, fortgesetzt werden kann, muss zuerst die Betriebsart ("FSelOM") neu geladen werden.

Untenstehendes Liniendiagramm zeigt die drei Stoparten.

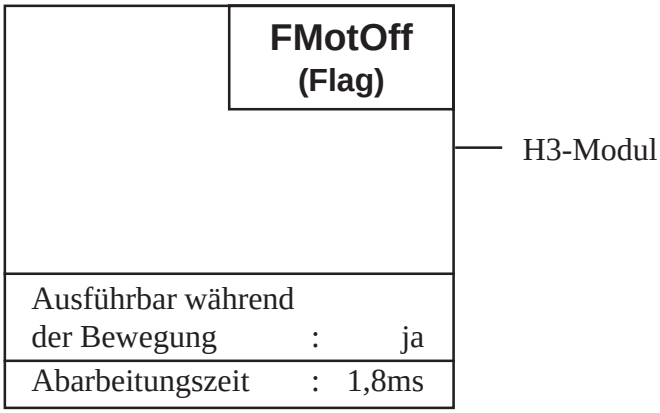


# FMotOff

Funktion : - **Mot**or **Off**
- Motor Aus

FMotOff

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

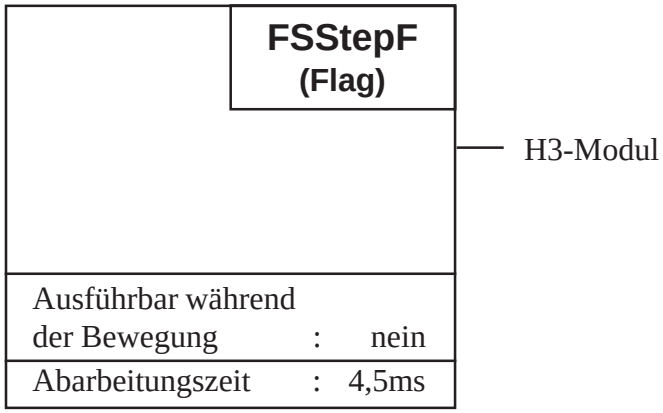
Mit dieser Funktion wird die Regelung ausgeschaltet. D.h. der Stellgrössenausgang wird Null gesetzt. Der Befehl "FMotOff" hat die gleiche Funktion wie der Befehl "FStop" wenn das Bit 8 im Register "MCW" gesetzt ist.

# FSStepF

Funktion : - **S**ingle **Step F**orward
- Einzelschritt vorwärts

FSStepF

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird ein Einzelimpuls in die positive Bewegungsrichtung gefahren. Es wird relativ eine Zielposition von +1 Impuls geladen und die Bewegung gestartet. Bei der Ausführung muss der Wertebereich der Zielposition beachtet werden. Die Funktion darf nicht ausgeführt werden, wenn die absolute Zielposition die positive Grenze des Wertebereichs erreicht hat.

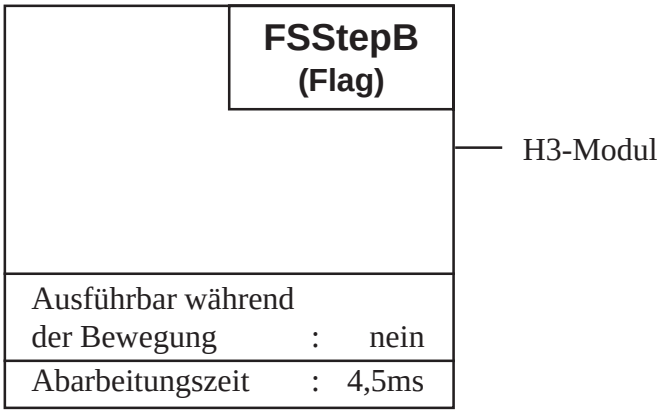
Beachte: 1 Impuls entspricht ¼ Encoderteilung (Impulsvervierfachung am Positionsdecoder).

# FSStepB

Funktion : - **S**ingle **Step B**ackwards
- Einzelschritt rückwärts

FSStepB

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird ein Einzelimpuls in die negative Bewegungsrichtung gefahren. Es wird relativ eine Zielposition von -1 Impuls geladen und die Bewegung gestartet. Bei der Ausführung muss der Wertebereich der Zielposition beachtet werden. Die Funktion darf nicht ausgeführt werden, wenn die absolute Zielposition die negative Grenze des Wertebereichs erreicht hat.

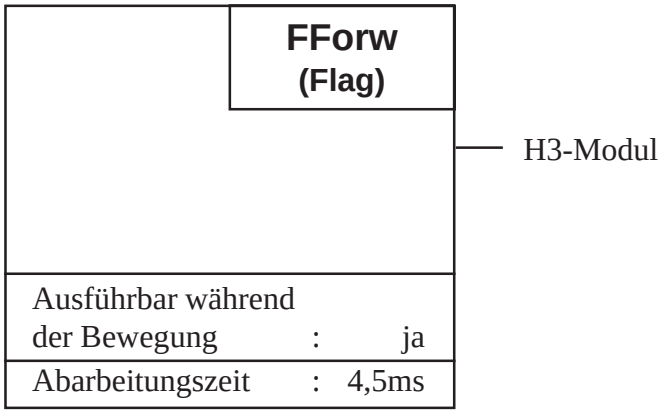
Beachte: 1 Impuls entspricht ¼ Encoderteilung (Impulsvervierfachung am Positionsdecoder).

# FForw

Funktion : - **Forw**ard with defined velocity
- Vorwärts mit definierter Geschwindigkeit

FForw

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

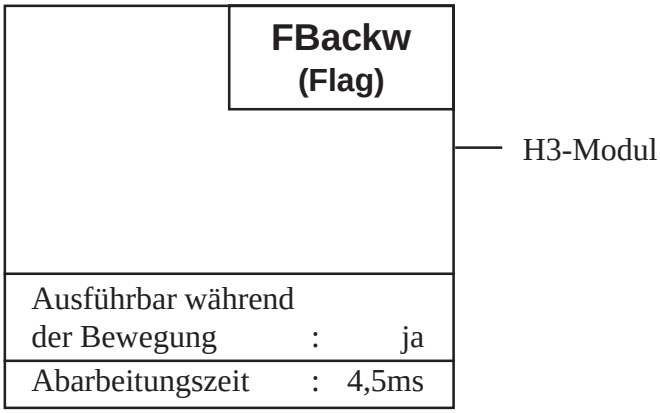
Mit dieser Funktion wird mit der zuletzt geladenen Geschwindigkeit in die positive Bewegungsrichtung gefahren.Um die Bewegung wieder zu stoppen, muss ein manueller Stopbefehl ausgeführt werden.
Die Funktion wird realisiert, indem die grösstmögliche, positive Zielposition geladen und dann ein Startbefehl erteilt wird.

# FBackw

Funktion : - **Backw**ards with defined velocity
- Rückwärts mit definierter Geschwindigkeit

FBackw

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

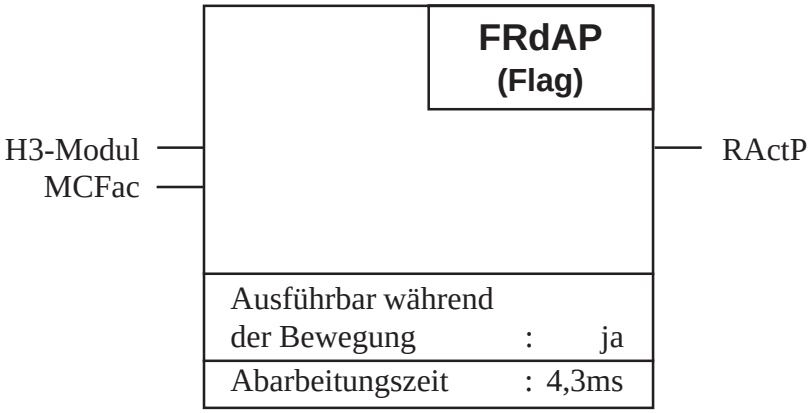
Mit dieser Funktion wird mit der zuletzt geladenen Geschwindigkeit in die negative Bewegungsrichtung gefahren. Um die Bewegung wieder zu stoppen, muss ein manueller Stopbefehl ausgeführt werden.
Die Funktion wird realisiert, indem die grösstmögliche, negative Zielposition geladen und dann ein Startbefehl erteilt wird.

# FRdAP

Funktion : - **R**ead **A**ctual **P**osition
- Lese Istposition

FRdAP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest die Istposition vom H3-Modul und kopiert diese in das Register "RActP".

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RActP | **R**egister **Act**ual **P**osition<br>Register Istposition<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

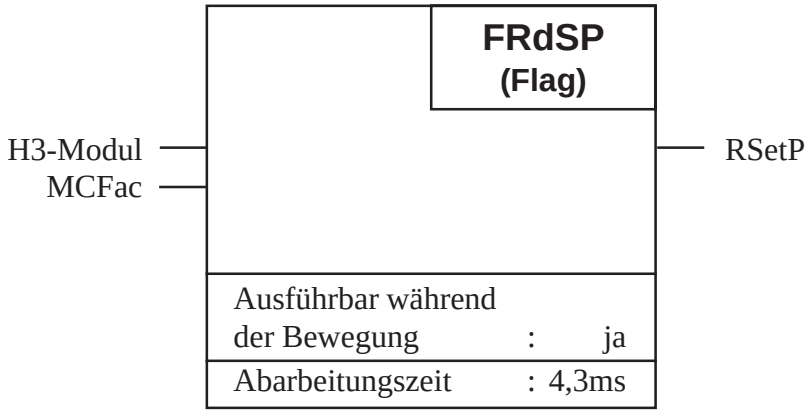
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Das Register "MCFac" wird von der obigen Funktion gelesen und zur Umrechnung der Position von Anzahl Impulsen in ein metrisches Mass benutzt.

# FRdSP

Funktion : - **R**ea**d** **S**etpoint **P**osition
- Lese Sollposition

FRdSP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest die momentane Sollposition am Ausgang des Generators für das Geschwindigkeitsprofil und kopiert diese in das Register "RSetP". Die Differenz dieser Sollposition und der Istposition wird dem PID-Regler zugeführt.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RSetP | **R**egister **Set**point **P**osition<br>Register Sollposition<br><br>Wert:<br>$[-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

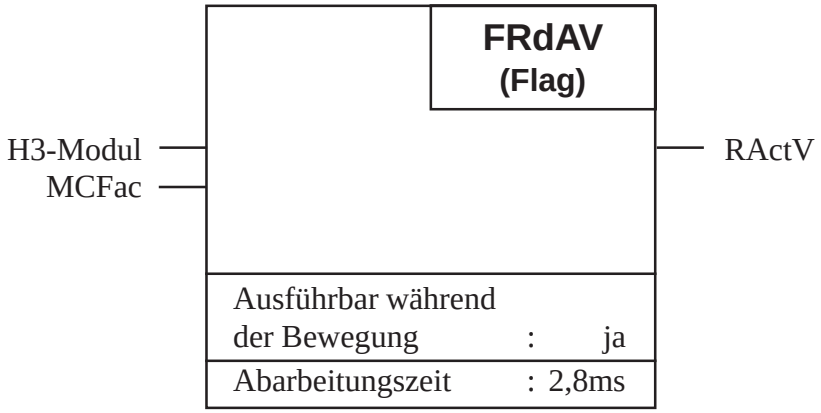
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Das Register "MCFac" wird von der obigen Funktion gelesen und zur Umrechnung der Position von Anzahl Impulsen in ein metrisches Mass benutzt.

# FRdAV

Funktion : - **Rea**d **A**ctual **V**elocity
- Lese Istgeschwindigkeit

FRdAV

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest die Istgeschwindigkeit der Achse vom H3-Modul und kopiert diese in das Register "RActV".
Vom Controller im H3-Modul können jedoch nur die 14 höherwertigen Bit der Istgeschwindigkeit ausgelesen werden. Für kleine Geschwindigkeiten kann deshalb kein vernünftiger Wert gelesen werden und es empfiehlt sich, an Stelle der Ist-, die Sollgeschwindigkeit auszulesen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RActV | **R**egister **Act**ual **V**elocity<br>Register Istgeschwindigkeit<br><br>Wert:<br>$[0..+(2^{30}-1)]/k*22348*10^{-6}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

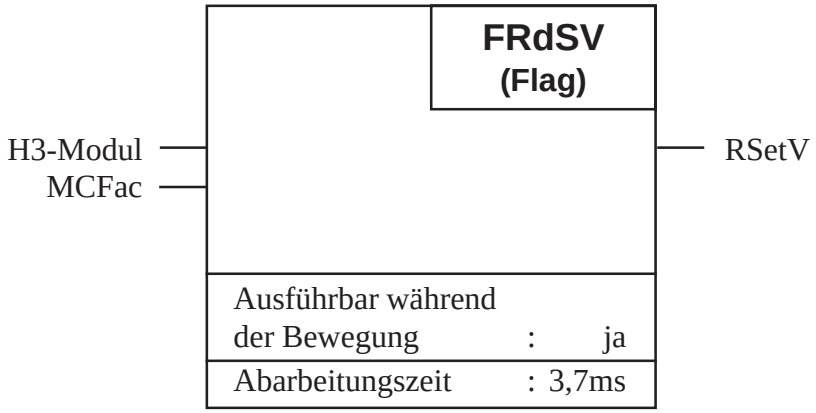
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Das Register "MCFac" wird von der obigen Funktion gelesen und zur Umrechnung der Geschwindigkeit von Anzahl Impulsen/sec. in ein metrisches Mass benutzt.

# FRdSV

Funktion : - **R**ea**d** **S**etpoint **V**elocity
- Lese Sollgeschwindigkeit

FRdSV

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest die momentane Sollgeschwindigkeit vom Profilgenerator und kopiert diese in das Register "RSetV".

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RSetV | **R**egister **Set**point **V**elocity<br>Register Sollgeschwindigkeit<br><br>Wert:<br>$[0..+(2^{30}-1)]/k*22348*10^{-6}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

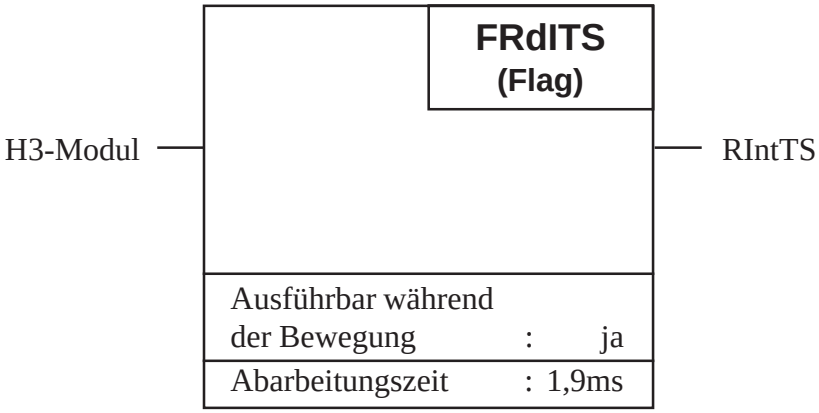
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Das Register "MCFac" wird von der obigen Funktion gelesen und zur Umrechnung der Geschwindigkeit von Anzahl Impulsen/sec. in ein metrisches Mass benutzt.

# FRdITS

Funktion : - **R**ea**d** **I**ntegration **T**erm **S**um
- Lese Integrationssumme

FRdITS

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest den Integrationswert (ki*∑e[n]) des PID-Reglers und kopiert diesen in das Register "RIntS". Die Funktion wird vorallem für die Abstimmung der Regelparameter während der Inbetriebnahme angewendet.

**Beschreibung der Ein- und Ausgänge:**

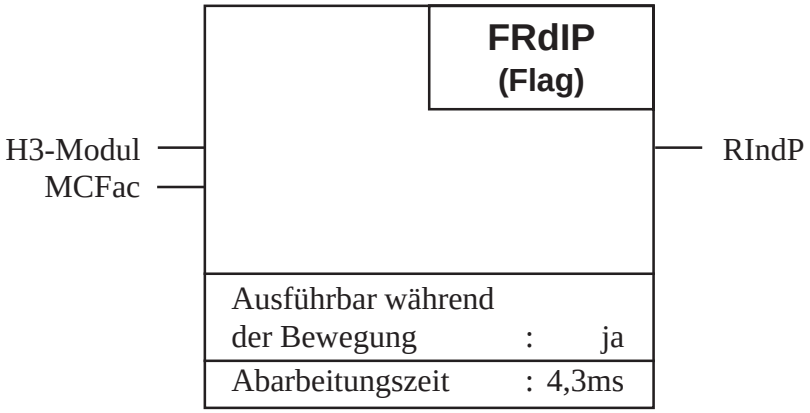
| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RIntTS | **R**egister<br>**Int**egration **T**erm **S**um<br>Register Integrationssumme<br><br>Wert:<br>Der Wert liegt im Bereich, der mit der Funktion "FLdRP" definierten Integrationsgrenze (Reg. "IntL") | nein | R | Integer | -- |

# FRdIP

Funktion : - **R**ea**d** **I**ndex **P**osition
- Lese Indexposition

FRdIP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Die Funktion liest die Indexposition aus dem Indexpositionsregister vom H3-Modul und kopiert diese in das Register "RIndP" (siehe auch Funktion "FSetIP").

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RIndP | **R**egister **Ind**ex **P**osition<br>Register Indexposition<br><br>Wert:<br>$[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

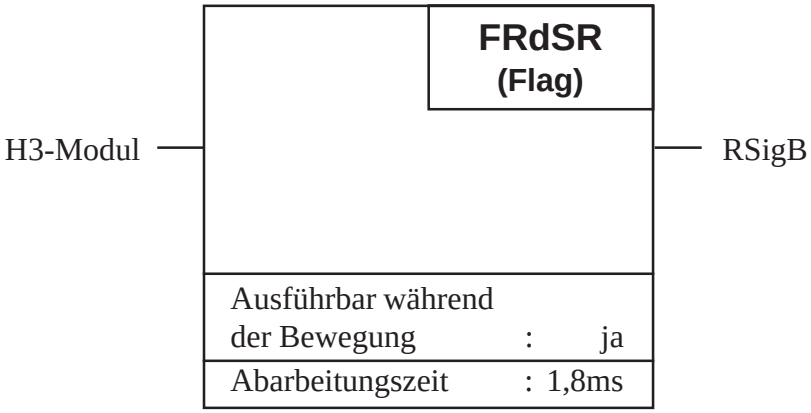
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Das Register "MCFac" wird von der obigen Funktion gelesen und zur Umrechnung der Position von Anzahl Impulsen in ein metrisches Mass benutzt.

# FRdSR

Funktion : - **R**ea**d** **S**ignal **R**egister
- Lese Signalregister

FRdSR

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion kann das Signalregister einer Achse vom H3-Modul gelesen werden.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| RSigB | **R**egister **Sig**nalisation **B**its<br>Register Signalisationsbit | nein | R | Integer | siehe folgende Seite |

Die einzelnen Bit im Register "RSigB" haben die folgende Bedeutung:

Bit 0: wird "1" gesetzt, nachdem die Funktion "SetIP" (setze Index Position) ausgeführt wurde. Das Bit wird zurückgesetzt, nachdem die nächste Indexposition (Indeximpuls) erfasst wurde.

Bit 1: keine Bedeutung

Bit 2 bis 6: zeigen den Zustand der Statusflag (siehe Funktion "FResSF").

Bit 7: wird "1" gesetzt, wenn die Regelung ausgeschaltet ist (Stellgrössenausgang = 0). Die Regelung wird durch folgende Ereignisse ausgeschaltet:
- Einschalten der Speisung
- Nach Abarbeitung des FB "AxInit"
- Bei einer Positionsfehlerüberschreitung (falls so definiert)
- Ausführen der Funktion "MotOff" (Motor Aus Befehl)
- Ausführen der Funktion "FStop" (wenn Stopart so definiert)

Das Bit wird mit dem nächsten Startbefehl ("FStart") zurückgesetzt.

Bit 8: wird "1" gesetzt beim Einschalten der Speisung oder wenn der Stellgrössenausgang mit FB "AxInit" als PWM-Ausgang definiert wird. Das Bit wird zurückgesetzt, wenn der Ausgang mit FB "AxInit" als ± 10V Analogausgang definiert wird.

Bit 9: zeigt die definierte Massnahme bei einem Überschreiten des maximalen Positionsfehlers.
"0" --> nur das Statusflag "ExcEr" wird gesetzt
"1" --> das Statusflag wird gesetzt und die Regelung ausgeschaltet

Bit 10: wird "1" gesetzt, wenn der Generator das errechnete Geschwindigkeitsprofil beendet hat.
Das Bit wird beim nächsten Startbefehl ("FStart") zurückgesetzt.

Bit 11: zeigt die mit Funktion "FSelOM" gewählte Betriebsart an.
"0" --> Positionier-Betrieb
"1" --> Drehzahl-Betrieb
Setzen und Rücksetzen des Bit erfolgt jeweils erst beim nächsten Startbefehl.

Bit 12: zeigt die Drehrichtung im Drehzahlbetrieb an.
"0" --> vorwärts
"1" --> rückwärts
Setzen und Rücksetzen des Bit erfolgt jeweils erst beim nächsten Startbefehl.

Bit 13: keine Bedeutung

Bit 14: zeigt an, dass mit den Funktionen "FLdAA/R" eine neue Beschleunigung geladen wurde.
Das Bit wird beim nächsten Startbefehl zurückgesetzt.

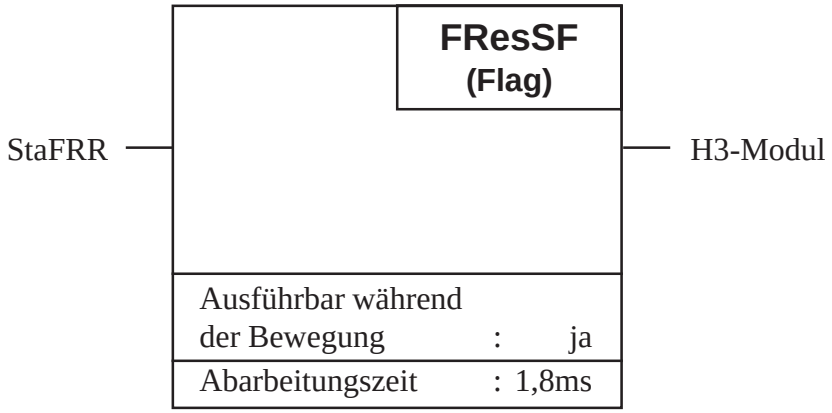
Bit 15 bis 31: keine Bedeutung

# FResSF

Funktion : - **Res**et **S**tatus **F**lag
- Reset Status Flag

FResSF

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

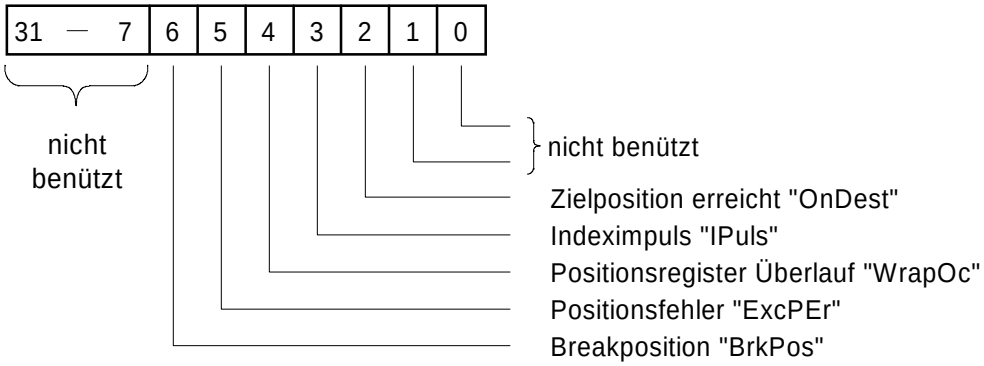
Mit dieser Funktion werden die Statusflag einer Achse zurückgesetzt. Die Flag können einzeln oder alle gleichzeitig zurückgesetzt werden.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| StaFRR | **Sta**tus **F**lag **R**eset **R**egister<br>Rückstellregister<br>für Statusbits | nein | R | Binär | siehe folgende Seite |

Die Funktion liest aus dem Register "StaFRR", welche Statusflag zurückgesetzt werden sollen. Das Flag wird zurückgesetzt, wenn das entsprechende Bit im Register Null ist.

Bedeutung des Registers "StaFRR" :




Bedeutung der Statusflag:

"OnDest" **On Dest**ination / Zielposition erreicht

Wird durch folgende Ereignisse "1" gesetzt:
- der Generator hat das errechnete Geschwindigkeitsprofil beendet. Infolge falsch eingestellter Parameter oder mechanischer Probleme kann es passieren, dass der Motor noch nicht endgültig in der Zielposition ist und das Flag trotztdem gesetzt wird, da der Generator das Sollprofil bereits beendet hat.
- die Regelung ist ausgeschaltet (z.B. nach der Funktion "FMotOff")
- nach einem manuellen Stop (Funktion "FStop")

Das Flag wird bei einem Startbefehl (Funktion "FStart") automatisch zurückgesetzt

"IPuls" **I**ndex **Puls** / Indeximpuls erfasst

Wird "1" gesetzt, wenn ein Indeximpuls erfasst wurde und die Istposition in das Indexpositionsregister im H3-Modul geschrieben wurde (siehe auch Funktion "FSetIP").

"WrapOc" **Wrap**around **Oc**cured / Positionsregister-Überlauf

Wird "1" gesetzt, wenn es einen Überlauf des Positionsregisters gibt. Einen Überlauf kann es im drehzahlgeregelten Betrieb geben.

"ExcPEr" **Exc**essive **P**osition **Er**ror / Positionsfehlerüberschreitung

Wird "1" gesetzt, wenn der Betrag des Positionsfehlers den mit der Funktion "FSetPE" definierten Wert überschreitet.

"BrkPos" **Bre**a**k Pos**ition / Breakposition

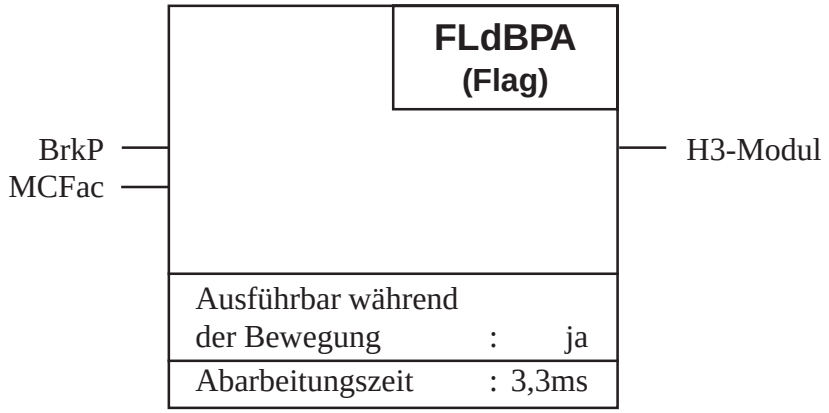
Wird "1" gesetzt, sobald die Istposition die mit Funktion "FLdBPA/R" geladene Breakposition über- oder unterschreitet.

# FLdBPA

Funktion : - **L**oa**d** **B**reak **P**osition **A**bsolute
- Lade Break Position Absolut

FLdBPA

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Breakposition absolut in das H3-Modul geladen. Absolut laden heisst, der Wert ist bezogen auf die Nullposition. Die geladene Position wird vom H3-Modul sofort in das Arbeitsregister übernommen. Wird die Breakposition erreicht, so wird das Statusflag "BrkPos" gesetzt . Das Flag kann mit der Funktion "ResSF" zurückgesetzt werden.
Mit dieser Funktion ergibt sich die Möglichkeit, an einer bestimmten Position eine Meldung zu erhalten, um z.B. die Geschwindigkeit oder die Regelparameter zu ändern.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| BrkP | **Br**ea**k** **P**osition<br>Break Position<br><br>Wert:<br>$[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

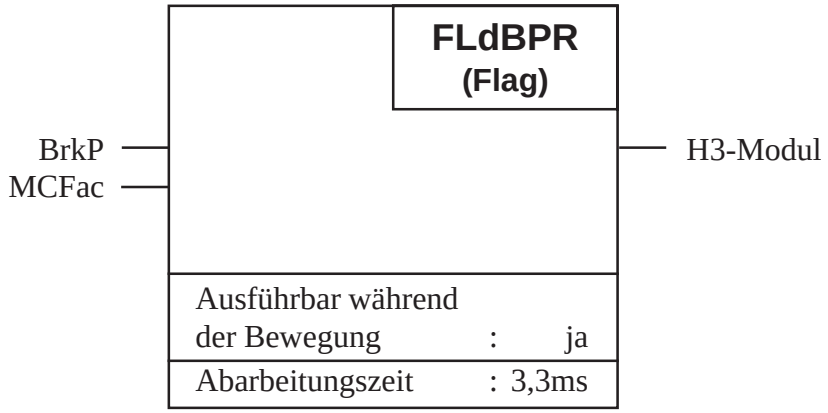
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Es ist zu beachten, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll, für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

# FLdBPR

Funktion : - **L**oa**d** **B**reak **P**osition **R**elative
- Lade Break Position Relativ

FLdBPR

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dieser Funktion wird eine Breakposition relativ in das H3-Modul geladen. Relativ laden heisst, der Wert ist bezogen auf die momentane Zielposition. Dabei ist zu beachten, dass die negative Breakposition addiert mit der Zielposition den gültigen Wertebereich für die Zielposition nicht überschreitet. Die geladene Position wird vom H3-Modul sofort in das Arbeitsregister übernommen. Wird die Breakposition erreicht, so wird das Statusflag "BrkPos" gesetzt . Das Flag kann mit der Funktion "ResSF" zurückgesetzt werden.
Mit dieser Funktion ergibt sich die Möglichkeit, an einer bestimmten Position eine Meldung zu erhalten, um z.B. die Geschwindigkeit oder die Regelparameter zu ändern.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| BrkP | **Br**ea**k** **P**osition<br>Break Position<br><br>Wert:<br>$[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$<br>Einheit: bestimmt durch k | nein | R | Integer | -- |
| MCFac | **M**otion **C**ontrol **Fac**tor<br>Maschinenfaktor | nein | R | Fl.Punkt | 0.. $9{,}223371*10^{18}$ |

**Maschinenfaktor k in Register "MCFac":**

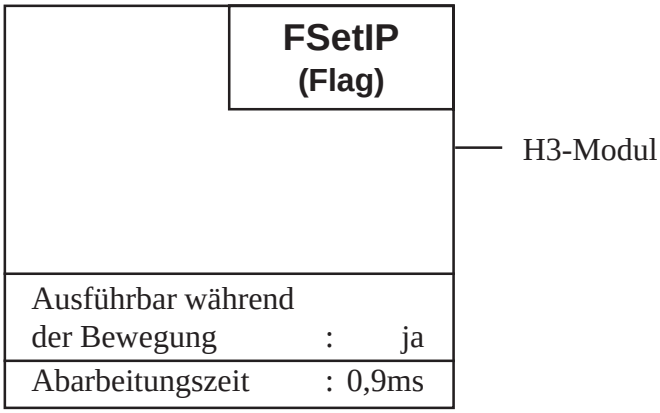
Der Faktor k hat die gleiche Bedeutung wie bei der Funktion "FLdDA". Es ist zu beachten, dass dieser Faktor für die Zielposition, Geschwindigkeit und Beschleunigung vom gleichen Register gelesen wird. Es ist deshalb sinnvoll, für die Eingabe dieser Parameter die gleichen Einheiten zu wählen.

# FSetIP

Funktion : - **S**et **I**ndex **P**osition
- Erfasse Index Position

FSetIP

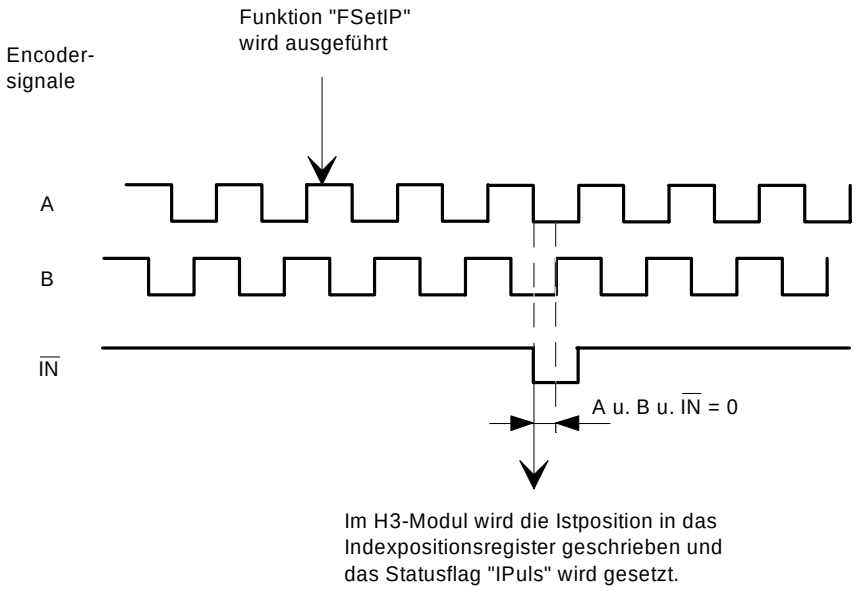
Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Nachdem diese Funktion ausgeführt ist, wird beim nächsten Mal, wenn die Encodersignale A, B und der Indexpulseingang gleichzeitg den Zustand Null haben, die Istposition in das Indexpositionsregister auf dem H3-Modul geschrieben. Wenn die Indexposition erfasst wurde, wird das Statusflag "IPuls" gesetzt. Die Indexposition kann mit der Funktion "FRdIP" gelesen werden vom H3-Modul.

Untenstehendes Diagramm zeigt den Ablauf im H3-Modul im Zusammenhang mit den Encodersignalen nachdem diese Funktion ausgeführt wurde.

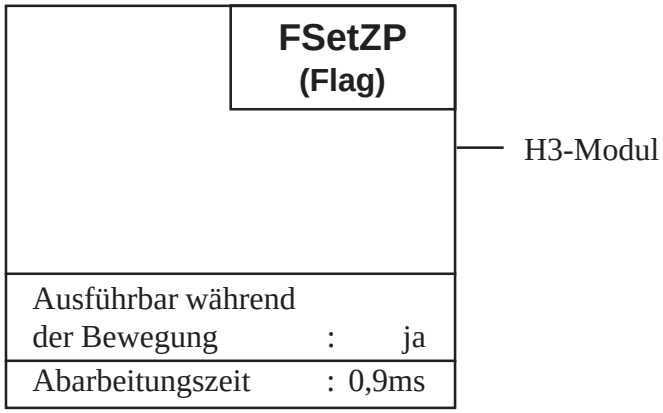


# FSetZP

Funktion : - **Set Z**ero **P**osition
- Setze Nullposition

FSetZP

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

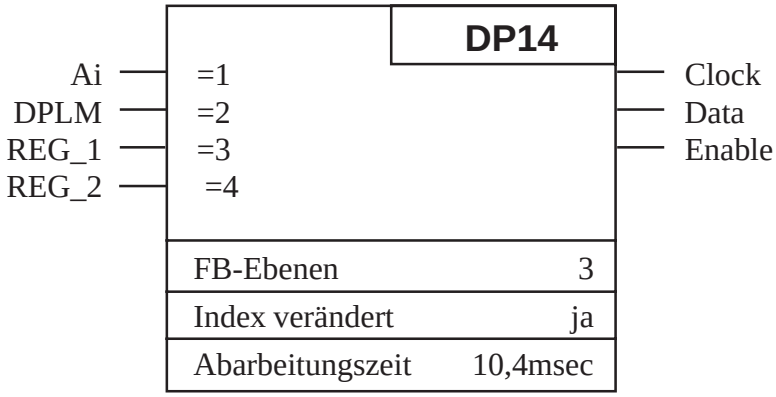
Mit dieser Funktion wird die Istposition als Nullposition definiert. Wird die Funktion während einer Bewegung ausgeführt, so wird die momentane Zielposition nicht beeinflusst, wenn kein Startbefehl "FStart" ausgeführt wird.

# DP14

Funktion : - **D**isplay Contents of Register on PCA2.D**14**
- Registerwerte mit PCA2.D14 anzeigen

DP14

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit diesem Befehl kann ein Registerwert mit 1*10 Digit oder mit 2*6 Digit auf dem Anzeigemodul PCA2.D14 angezeigt werden.
Folgende Anzeigeformate sind möglich:

**1*10 Digits** (1Register)

| | | | | | |
|---|---|---|---|---|---|
| b | v | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 |

b = blank ; v = blank oder "-"
1 .. 10 = Digits

Anzeigebereich = Wertebereich der Register ± 2'147'483'647

**2*6 Digits** (2Register)

| | | | | | |
|---|---|---|---|---|---|
| v | x | x | x | x | x |
| v | x | x | x | x | x |

w = Inhalt des Registers
0 ≤ w ≤ + 999'999 : v = MSD
w > + 999'999 : v = "A"
0 > w ≥ - 99'999 : v = " -"
w < - 99'999 : v = "U"

Anzeigebereich : - 99'999 ... + 999'999

Mit dem Modul PCD4.H320 können 2 Anzeigemodule betrieben werden.

Das Flag DPLM bestimmt die Art der Anzeige:

DPLM = "0" --> 1*10 Digits
DPLM = "1" --> 2* 6 Digits

Parameter: REG_1 : Register für den 1. Anzeigewert
REG_2 : Register für den 2. Anzeigewert

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Ai | Achsennummer i | | K | Integer | 1.. 32 |
| DPLM | Dislpay Modus | | F | Binary | 0, 1 |
| REG_1 | Anzeigewert 1 | | R | Integer | 0.. $10^9$-1 |
| REG_2 | Anzeigewert 2 | | R | Integer | 0.. $10^6$-1 |
| Clock | Ausgang zu D14 | | O | Binary | 0, 1 |
| Data | " " | | O | Binary | 0, 1 |
| Enable | " " | | O | Binary | 0, 1 |

**Programmbeispiel** für die Anzeige der Istposition und Sollgeschwindigkeit von der Achse 1 mit einem PCA2.D14::

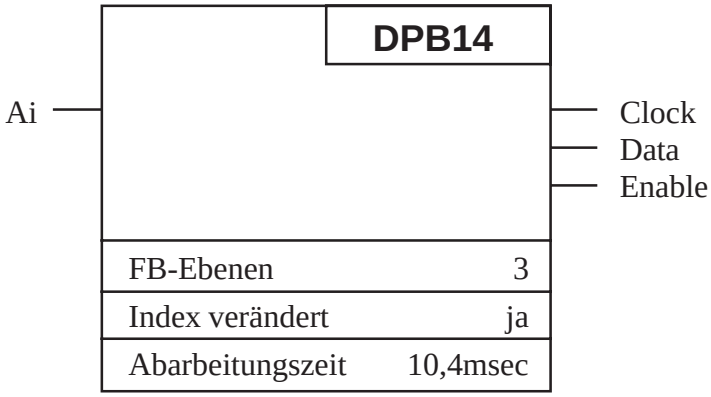
```
 ¦
ACC  H
SET  F    DplM+FA1  ; Display Mode = 2*6 Digits
CFB       DP14      ; Display on PCA2.D14
          1         ; Axis 1
     F    DplM+FA1  ; Display Mode
     R    RActP+RA1 ; Actual Position
     R    RSetV+RA1 ; Setpoint Position
 ¦
```

# DPB14

Funktion : - Clear Display on PCA2.D14
- Löschen der Anzeige auf PCA2.D14

DPB14

Softwarepaket : PCD9.H3E1




**Funktionsbeschreibung:**

Mit dem Befehl "DPB14" werden alle Ziffern der Anzeige PCA2.D14 mit Leerzeichen gefüllt, so dass sie erlöschen.

**Beschreibung der Ein- und Ausgänge:**

| Symbol | Bezeichnung / Funktion | Para-meter | Daten | | |
|---|---|---|---|---|---|
| | | | Typ | Format | Wert |
| Ai | Achsennummer i | | K | Integer | 1.. 32 |
| Clock | Ausgang zu D14 | | O | Binary | 0, 1 |
| Data | " " | | O | Binary | 0, 1 |
| Enable | " " | | O | Binary | 0, 1 |

# 8. Fehlererkennung und -Behandlung

Der H3-Controller meldet einen Fehler durch Setzen des Befehls-Errorflag. Das Flag wird gesetzt, wenn der Controller einen gesendeten Befehl oder Daten nicht interpretieren kann.

Meistens ist ein Programmierfehler des Anwenders die Ursache für einen Befehls-Error, indem z.B. versucht wird einen Wert ins H3-Modul zu laden, der ausserhalb seines erlaubten Wertebereiches liegt.

Fehler-Beispiele:

- Mit der Funktion “FLdVA” (lade Geschwindigkeit absolut) wird eine negative Geschwindigkeit geladen.

- Der Maschinenfaktor k wurde nicht im Fliesspunkt-Format in das Register “MCFac” geladen. —> Kann bei der Umrechnung der Einheit eines Parameters einen Überlauf des Wertebereiches bewirken.

- Ein Regelfaktor > 32767 soll ins H3-Modul geladen werden.

- Es wird versucht, für die erste Bewegung (vor dem ersten Startbefehl) einen Bewegungsparameter relativ zu laden (“FLdDR”, “LdVR”, “FLdAR” und FLdBPR”).

- Eine Störung auf dem PCD4-Bus.

**Behandlung des Befehls-Error**

Bei einem Fehler ignoriert der H3-Controller den gesendeten Befehl und setzt das Befehls-Errorflag. Dieses Flag wird von allen Funktionsblöcken, die mit dem H3-Modul kommunizieren, selbständig überwacht. Der Anwender hat keinen direkten Zugriff auf das Errorflag. Tritt ein Fehler auf, so wird der zuletzt gesendete Befehl maximal zweimal wiederholt. Nach dem dritten aufeinanderfolgenden Fehler wird der Fehlerbehandlungs-FB “ComErS” aufgerufen und nach dessen Abarbeitung mit dem nächsten Schritt im Programm weitergefahren. Der FB "ComErS" befindet sich in der H3FB.SRC-Datei. Er kann vom Anwender beliebig programmiert werden. Im Fehlerfall bestimmt somit der Anwender selbst, welche Massnahmen getroffen werden sollen. Ein Aufruf dieses FB bedeutet in jedem Fall, dass ein Befehl nicht ausgeführt wurde und somit ein korrekter Ablauf des H3-Programmes nicht gewährleistet ist. Für die Inbetriebnahme empfiehlt es sich, mittels einer Division durch Null einen Aufruf des XOB 13 zu provozieren.

Beispiel:

```
FB      ComErS      ; Command Error Stop

DIV     R  0        ;
        K  0        ;     provoziert Aufruf von XOB 13
        R  0        ;
        R  0        ;

EFB
```

Wird im XOB 13 der Befehl “DIAG” programmiert, kann sehr schnell festgestellt werden, wo der Fehler im Anwenderprogramm aufgetreten ist.

Beispiel:

```
XOB 13

DIAG    R  1        ; Diagnoseregister
```

| Ev. den Antrieb abschalten |
|---|

```
HALT

EXOB
```

**Vorgehen zur Lokalisierung des Fehlers**

Im Debugger mit Hilfe der Diagnoseregister feststellen, bei welchem Befehl der Fehler ausgelöst wird.
Siehe dazu auch Beschreibung der PCD-Instruktion “DIAG”.

# 9. Didaktische Anwenderbeispiele

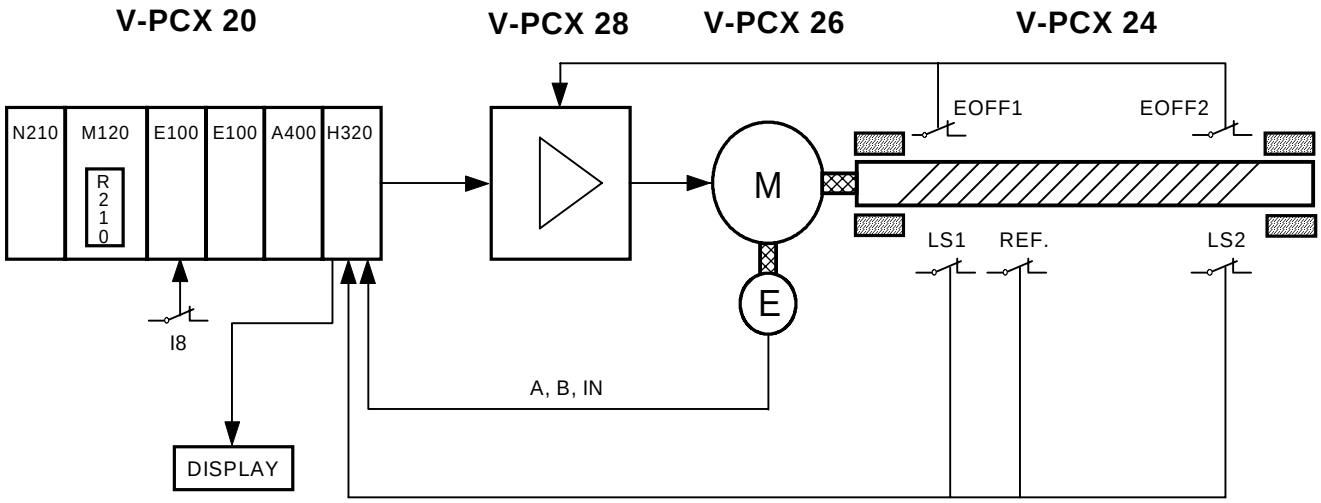
## 9.1 Beispiel 1

Bei dem Beispiel handelt es sich um eine sehr einfache Anwendung mit einer Achse. Es soll zeigen, welche Schritte in welcher Reihenfolge gemacht werden müssen, um eine einfache Bewegung zu fahren .

**Aufgabenstellung**

Hardware:

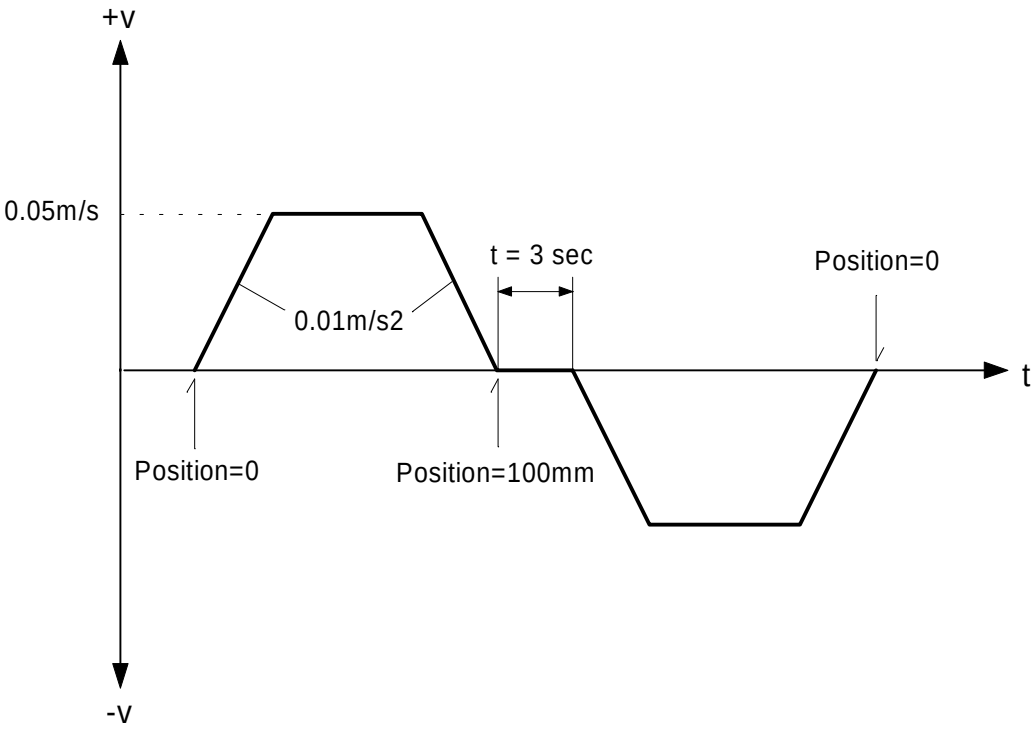
Das Beispiel basiert auf der Hardware der Workshopmodelle V-PCX 20, V-PCX 24, V-PCX 26 und V-PCX 28.




Daten der Achse:

- Encoder 500 Imp./Umdr.
- Spindel 2mm Steigung
- Eingabe der Zielposition in 1/10mm Auflösung
- Maximale Geschwindigkeit des Antriebes 0.1m/s
- Maximale zulässige Beschleunigung 0.05m/s$^2$

Folgender Bewegungsablauf soll ausgeführt werden:




Mit der Taste, angeschlossen am Eingang 8, soll die Bewegung gestartet werden.
Es wird angenommen, dass nach dem Einschalten der Steuerung die Nullposition bereits definiert wurde und deshalb keine Referenzfahrt notwendig ist. In diesem ersten Beispiel soll auch auf die Überwachung der Endschalter sowie auf eine Anzeige mit dem Display-Modul PCA2.D14 verzichtet werden.

## Lösung

Die in der Aufgabenstellung erwähnte Hardware und das Softwarepaket PCD9.H3E1 stehen zur Verfügung. Wir gehen davon aus, dass die gesamte Hardware bereits funktionsfähig installiert wurde. Die in der Folge beschriebenen Schritte sollten bei der Programmierung beachtet werden.

**a) Installation der Software**

Die zwei H3-Dateien H3DEF.SRC und H3FB.SRC werden in das Arbeitsverzeichnis kopiert. Als erstes soll die H3-Installation in der H3DEF.SRC-Datei wie folgt konfiguriert werden:

```
FMAH3   EQU   48          ; Basisadresse des H3-Moduls
IMode   EQU   6           ; Ausgangsport-Initialisierung (Analog)
MNA     EQU   1           ; 1 Achse eingesetzt
BAF     EQU   200         ; Basisadresse Flag
BAR     EQU   200         ; Basisadresse Register
BAC     EQU   40          ; Basisadresse Zähler
BAFB    EQU   900         ; Basisadresse Funktionsblöcke
RA1     EQU   0*NoRfeA    ; Registerblock-Konstante Achse 1
FA1     EQU   0*NoFfeA    ; Flagfeld-Konstante Achse 1
```

Da nicht mit externen Symbolzuweisungen gearbeitet werden soll, wird das Symbol

```
PUBLSYM          EQU     0
```

definiert.

Alle folgenden Symbolzuweisungen sind unberührt zu belassen.

In der H3FB.SRC-Datei wird das Symbol

```
EXTNSYM     EQU     0
```

definiert.

Durch diese Definition wird beim Assemblieren die Symboldefinitionsdatei H3DEF.SRC mit der Direktive $INCLUDE automatisch eingebunden.
Das Einbinden der Definitionsdatei in die Anwender- und die H3FB.SRC-Datei hat den Vorteil, dass bereits nach dem Assemblieren in der Listdatei die Absolutadressen zur Verfügung stehen. Diese werden benötigt, wenn beim Test des Programmes allenfalls ein Wert mit dem Debugger angezeigt werden soll.
Würde hingegen mit externen Symbolzuweisungen gearbeitet, so wären die Absolutadressen erst verfügbar nachdem das DOC-File generiert wurde.

Am Ende der H3FB.SRC-Datei befindet sich der FB “ComErS”, welcher bei einem wiederholten Auftreten eines “Command Errors” aufgerufen wird. In diesem FB kann der Anwender bestimmen, welche Massnahmen im Fehlerfall ergriffen werden sollen. Für die Inbetriebnahme empfiehlt es sich, mittels einer Division durch Null den Aufruf des XOB 13 zu provozieren. Im XOB 13 kann dann der Befehl “DIAG” programmiert werden. Auf diese Weise kann sehr schnell festgestellt werden, wo der Fehler im Anwenderprogramm aufgetreten ist. Für nähere Details siehe Kpitel 8. Fehlererkennung.

```
FB      ComErS   ; Command Error Stop
DIV     R  0
        K  0     ; Division durch 0
        R  0
        R  0
EFB
```

Ausgenommen diese zwei Anpassungen darf die H3FB.SRC-Datei nicht verändert werden.
Im nächsten Schritt wird die Datei assembliert, um festzustellen, ob die genannten Anpassungen korrekt vorgenommen wurden. Ist die Assemblierung erfolgreich verlaufen, muss die Datei während des ganzen Programmiervorganges nicht mehr assembliert werden und kann später nur noch mit dem Anwenderprogramm gelinkt werden.

**b) Anwenderprogramm**

Bei der Erstellung des Programmes halten wir uns an die in Kapitel 7.2 beschriebene Programmstruktur:

**1. Initialisierung im XOB 16**

In einem ersten Schritt müssen die Werte für die folgenden Initialisierungs-Register bestimmt werden:
(Die Initialisierungsregister werden vom FB “AxInit” gelesen)

**- Bewegungs Control Wort “MCW”**

In diesem Register ist die Stop-Betriebsart und die Massnahme bei einem Positionsfehler definiert.

Stopart: siehe dazu Funktion “FStop”
In diesem 1. Beispiel spielt die Stopart zwar noch keine Rolle, da im Programm kein manueller Stopbefehl vorgesehen ist.
Dennoch definieren wir Bit 10 = “1” —> Stop mit definierter Verzögerung

Betriebsart: siehe dazu Funktion “FSelOM”
Wir arbeiten im Positionierbetrieb —> Bit 11 und 12 = “0”

Positionsfehler: siehe dazu Funktion “FSetPE”
Im Falle eines Positionsfehlers soll nur das Statusflag “ExcPEr” gesetzt werden.
—> Wert für Bit 0 bis 7 = 1BH (Hexadezimal)

Register “MCW”

Bit 31 13 12 11 10 9 8 7 0

| nicht benützt | 0 | 0 | 1 | 0 | 0 | 1BH |
|---|---|---|---|---|---|---|

—> Register “MCW laden mit dem Wert 41BH (Hexadezimal)

**- Positionsfehler “PosEr”** —> siehe Funktion “FSetPE”

In diesem 1. Beispiel ist die Angabe für den Positionsfehler unbedeutend, da das Statusflag “ExcPEr” mit dem Anwenderprogramm nicht überwacht wird (um beispielsweise den Antrieb abzuschalten).
Weil bei der Initialisierung der Positionsfehler trotzdem in das H3-Modul geladen wird und im erlaubten Wertebereich liegen muss, definieren wir, dass die maximale Differenz zwischen Soll- und Istposition eine Umdrehung betragen soll, damit das Statusflag “ExcPEr” gesetzt wird.

—> Register “PosEr” laden mit dem Wert 2000 = 4 * 500 Imp./Umdr.
(Flankenauswertung der Encoderimpulse)

**- PID-Faktoren** —> siehe Funktion “FLdRP”

Der Einfachheit halber wird an einem anderen Beispiel das Vorgehen zur Bestimmung der PID-Faktoren gezeigt. Wir gehen davon aus, dass die Faktoren bereits ermittelt wurden.
Die Register müssen mit den folgenden Werten geladen werden:

Proportionalfaktor “KProp”: 150

Integralfaktor “KInt”: 50

Differentialfaktor “KDer”: 50

Integrationsgrenze “IntL”: 500

Abtastzeit D-Anteil “Sampl”: 15 (5.46ms = 16*0,341ms)

**- Maschinenfaktor k "MCFac":** —> siehe Funktion "FLdDA"

Mit dem Maschinenfaktor wird die Einheit der Ein- und Ausgabewerte für eine Position, Geschwindigkeit und Beschleunigung bestimmt.
In der Aufgabenstellung wird für die Eingabe der Zielposition eine Auflösung von 1/10mm verlangt.

$$\text{—> } k = \frac{4 * In}{s} = \frac{4 * 500\ \text{Imp./Umdr.}}{20 * 1/10\text{mm/Umdr.}} = 100\ \text{Imp. pro } 1/10\text{mm}$$

—> Register "MCFac" laden mit dem Wert 100 (Fliesspunkt Format)

**- Geschwindigkeit "Veloc"** —> siehe Funktion "FLdVA"

Da bei vielen Anwendungen oft nur mit einer Geschwindigkeit gefahren wird, wird schon bei der Initialisierung eine absolute Geschwindigkeit geladen.
In unserem Beispiel soll die Geschwindigkeit 0.05m/s betragen.

—> Register "Veloc" laden mit dem Wert 500 (Einheit 1/10 mm/s)

**- Beschleunigung "Accel"** —> siehe Funktion "FLdAA"

Beschleunigung laut Aufgabenstellung : 0.01m/s$^2$

—> Register "Accel" laden mit dem Wert 100 (Einheit 1/10 mm/s$^2$)

Die jetzt ermittelten Werte müssen im XOB 16 vor dem Aufruf des FB "AxInit" in die Initialisierungsregister geladen werden.

**2. Zyklische Achsenbearbeitung**

Im COB 0 wird nur der FB "AxHndlg" zur Bearbeitung der Achse aufgerufen.

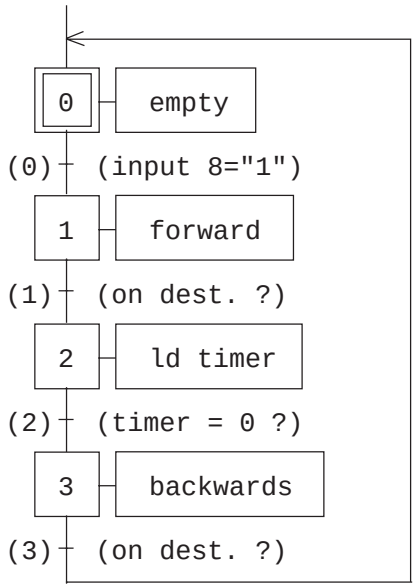
**3. Definition des Fahrprogrammes**

Das Fahrprogramm wird entsprechend dem vorgegebenen Geschwindigkeit/Zeit-Profil in einer GRAFTEC-Struktur (SB 0) programmiert

Auf den folgenden Seiten ist die Source-Datei (BSP01.SRC) des Anwenderprogramms für das Beispiel wiedergegeben.

```
; Demo programm for the motion control module PCD4.H3..
; ======================================================
; Name  : BSP01.SRC
; U. Jäggi 21.08.90

$ include H3DEF.SRC

      ;********************** Cold-Start (Initialisation)
      XOB         16
      ;----------------------- Cold-Start Definitions
      ;----------------------- loading of the initialisation registers

      Ld       R  MCW+RA1     ;Motion Control Word
                  41BH        ;Stop smothly, Position mode
                              ;only statusflag (Pos.error)
      Ld       R  PosEr+RA1   ;Position Error
                  2000        ;4 * 500 pulses
      Ld       R  KProp+RA1   ;Proportional factor
                  150
      Ld       R  KInt+RA1    ;Integral factor
                  50
      Ld       R  KDer+RA1    ;Derivative factor
                  50
      Ld       R  IntL+RA1    ;Integration Limit
                  500
      Ld       R  SampI+RA1   ;Sampling Interval
                  15          ;5.46ms
      Ld       R  MCFac+RA1   ;Motion Control Factor
                  100.0       ;100 Imp./1/10mm
      Ld       R  Veloc+RA1   ;Velocity
                  500         ;0.05m/s
      Ld       R  Accel+RA1   ;Acceleration
                  100         ;0.01m/s2

      ;-----------------------
      CFB         AxInit      ;Axis Initialisation
                  1           ;X axis
                  RA1
                  FA1
                  IMode       ;Initialisation mode: Analog/PWM
      ;-----------------------  End XOB 16
      EXOB

      ;**********************  Cyclic program
      COB         0
                  0
      ;-----------------------
      CFB         AxHndlg     ;Axis Handling
                  1           ;X axis
                  RA1
                  FA1
      ;-----------------------
      CSB         0           ;Call of the motion control program
      ;----------------------- ;End cyclic program
      ECOB
```

```
*** SAIA PCD GRAFTEC EDITOR $113 ***                          PAGE: 1
FILE: BSP01.GLS  (29.08.90  10.51)          PRODUCED: 29.08.90  10.51
***   SAIA AG   -   CH-3280 MURTEN   ***
SB-NUMBER:  0
PAGE-NB:  0
```



```
;********************** Motion control program
SB       0
;-----------------------
IST      0              ;empty
         I  3           ;on dest. ?
         O  0           ;input 8="1"
EST
;----------------------
ST       1              ;forward
         I  0           ;input 8="1"
         O  1           ;on dest. ?
ld       r  DestP+RA1   ;Destination Position
            1000        ;100mm = 1000*1/10mm
set      f  FLdDA+FA1   ;Load Destination Absolute
set      f  FStart+FA1  ;Start motion
EST
;----------------------
ST       2              ;ld timer
         I  1           :on dest. ?
         O  2           ;timer = 0 ?
ld       t  0           ;timer 0
            30          ;3s
EST
;----------------------
ST       3              ;backwards
         I  2           ;timer = 0 ?
         O  3           ;on dest. ?
ld       r  DestP+RA1   ;Destination Position
            0           ;Position 0
set      f  FLdDA+FA1   ;Load Destination Absolute
set      f  FStart+FA1  ;Start motion
EST
;----------------------
 TR      0              ;input 8="1"
         I  0           ;empty
         O  1           ;forward
sth      i  8           ;motion free ?
ETR
;----------------------
 TR      1              ;on dest. ?
         I  1           ;forward
         O  2           ;ld timer
sth      f  OnDest+FA1  ;On Destination ?
ETR
;----------------------
 TR      2              ;timer = 0 ?
         I  2           ;ld timer
         O  3           ;backwards
stl      t  0           ;timer = 0 ?
ETR
;----------------------
 TR      3              ;on dest. ?
         I  3           ;backwards
         O  0           ;empty
sth      f  OnDest+FA1  ;On Destination ?
ETR
;----------------------
ESB
```

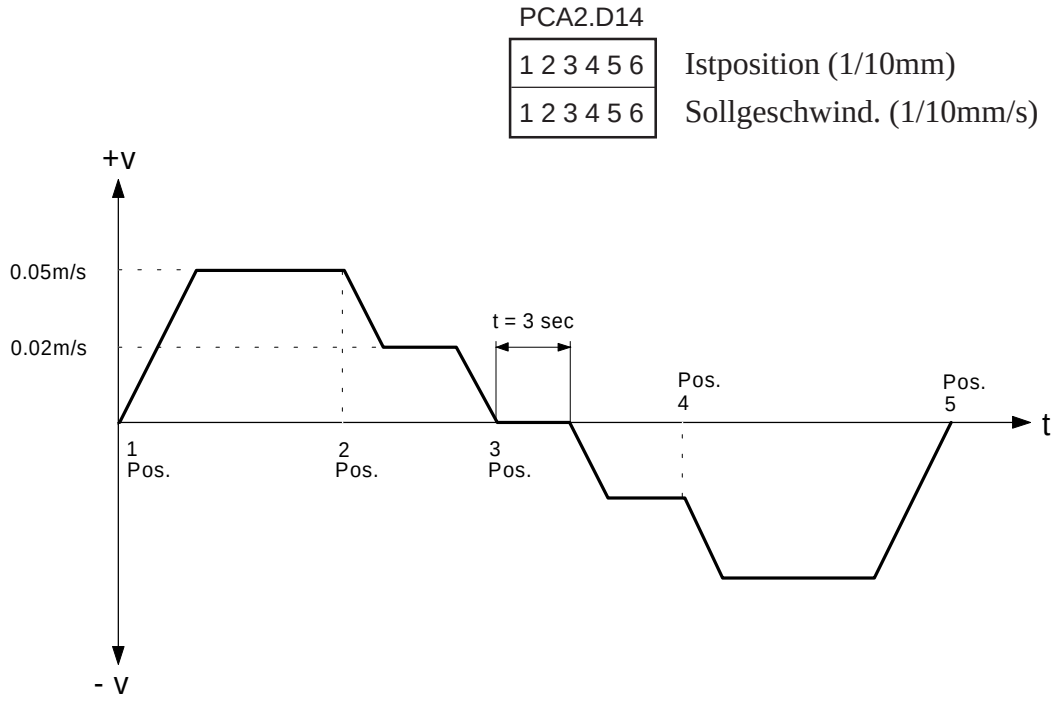
## 9.2 Beispiel 2

Das Beispiel ist aufbauend auf dem Beispiel 1. Die Erweiterung beinhaltet eine zusätzliche Geschwindigkeitsänderung während der Bewegung und die Anzeige auf dem Anzeigemodul PCA2.D14.

**Aufgabenstellung**

Hardware: Workshopmodelle wie Beispiel 1.

Folgender Bewegungsablauf soll ausgeführt werden:




| | | |
|---|---|---|
| Position 1: | 0mm | (Startposition) |
| Position 2: | 70mm | (Breakposition) |
| Position 3: | 100mm | (Zielposition) |
| Position 4: | 70mm | (Breakposition) |
| Position 5: | 0mm | (Zielposition) |

- Geschwindigkeit für langsam und schnell rückwärts gleich wie für Vorwärtsbewegung.
- Die Beschleunigung soll konstant 0.01m/s$^2$ betragen.
- Mit der Taste angeschlossen am Eingang 8 soll die Bewegung gestartet werden.
- Auf der Anzeige PCA2.D14 soll oben die Istposition und unten die Sollgeschwindigkeit angezeigt werden.

## Lösung

### a) Installation der Software

Wir gehen davon aus, dass die Hard- und Software wie bei Beispiel 1 bereits installiert ist.
(Die Dateien H3DEF.SRC und H3FB.SRC können ohne Änderung übernommen werden.)

### b) Anwenderprogramm

#### 1. Initialisierung im XOB 16

Kann ohne Änderung von Beispiel 1 übernommen werden .

#### 2. Zyklische Achsenbearbeitung

Als Erweiterung zu Beispiel 1 wird hier mit den Funktionen “FRdAP” und “FRdSV” die Istposition und Sollgeschwindigkeit vom H3-Modul gelesen und mit dem Funktionsblock “DP14” auf dem Displaymodul PCA2.D14 angezeigt .

#### 3. Definition des Fahrprogrammes

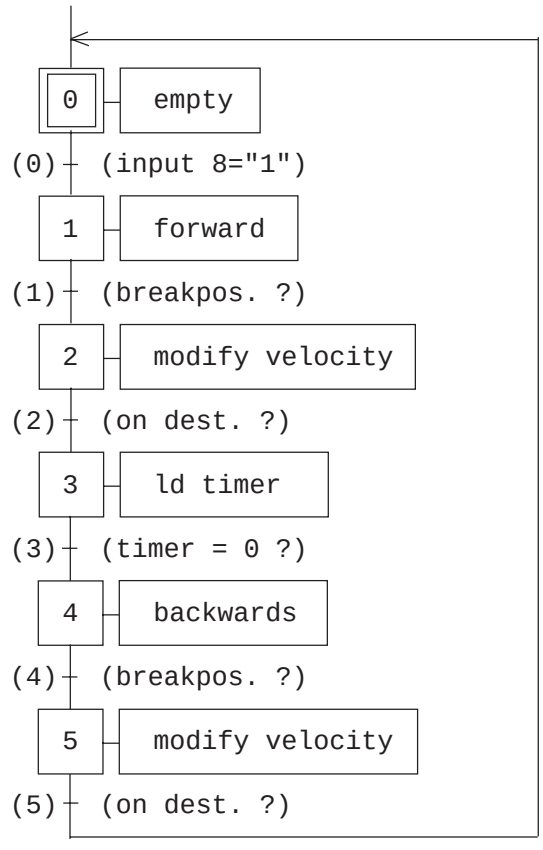
Definition des Fahrprogrammes entsprechend dem vorgegebenen Geschwindigkeit/Zeit-Profil in einer GRAFTEC-Struktur (SB 0). Soll während der Bewegung an einer bestimmten Position die Geschwindigkeit geändert werden, so kann dies durch das Laden einer Breakposition und Abfragen des Statusflag “BrkPos” realisiert werden. Dabei ist zu beachten, dass nach dem Laden der Breakposition das Statusflag “BrkPos” mit der Funktion “FResSF” zuerst zurückgesetzt werden muss.
Siehe dazu auch Beschreibung der Funktionen “FLdBPA” und “FResSF”.

Auf den folgenden Seiten ist die Source-Datei (BSP02.SRC) des Anwenderprogramms für das Beispiel wiedergegeben.

```
; Demo programm for the motion control module PCD4.H3..
; ======================================================
; Name  : BSP02.SRC
; U. Jäggi 27.08.90

$ include H3DEF.SRC

        ;********************** Cold-Start (Initialisation)
        XOB         16
        ;----------------------- Cold-Start Definitions
        ;----------------------- loading of the initialisation registers

        Ld       R  MCW+RA1     ;Motion Control Word
                    41BH        ;Stop smothly, Position mode
                                ;only statusflag (Pos.error)
        Ld       R  PosEr+RA1   ;Position Error
                    2000        ;4 * 500 pulses
        Ld       R  KProp+RA1   ;Proportional factor
                    150
        Ld       R  KInt+RA1    ;Integral factor
                    50
        Ld       R  KDer+RA1    ;Derivative factor
                    50
        Ld       R  IntL+RA1    ;Integration Limit
                    500
        Ld       R  SampI+RA1   ;Sampling Interval
                    15          ;5.46ms
        Ld       R  MCFac+RA1   ;Motion Control Factor
                    100.0       ;100 Imp./1/10mm
        Ld       R  Veloc+RA1   ;Velocity
                    500         ;0.05m/s
        Ld       R  Accel+RA1   ;Acceleration
                    100         ;0.01m/s2

        ;----------------------
        CFB         AxInit      ;Axis Initialisation
                    1           ;X axis
                    RA1
                    FA1
                    IMode       ;Initialisation mode: Analog/PWM
        ;----------------------  End XOB 16
        EXOB
```

```
;**********************  Cyclic program
COB         0
            0
;-----------------------
CFB         AxHndlg     ;Axis Handling
            1           ;X axis
            RA1
            FA1
;-----------------------
SET      F  FRdAP+FA1   ;Read Actual Position
SET      F  FRdSV+FA1   ;Read Setpoint Velocity
;-----------------------
SET      F  DplM+FA1    ;Diplay Mode = 2*6 digits
CFB         DP14        ;Display on PCA2.D14
            1           ;Axis 1
         F  DplM+FA1    ;Display Mode
         R  RActP+RA1   ;Actual Position
         R  RSetV+RA1   ;Setpoint Velocity
;-----------------------
CSB         0           ;Call of the motion control program
;----------------------- ;End cyclic program
ECOB
```

```
*** SAIA PCD GRAFTEC EDITOR $113 ***                              PAGE: 1
FILE: BSP01.GLS  (29.08.90  10.46)          PRODUCED: 29.08.90  10.52
***   SAIA AG   -   CH-3280 MURTEN   ***
SB-NUMBER:  0
PAGE-NB:  0
```



```
;********************** Motion control program
SB       0
;-----------------------
IST      0              ;empty
         I  5           ;on dest. ?
         O  0           ;input 8="1"
EST
;----------------------
ST       1              ;forward
         I  0           ;input 8="1"
         O  1           ;breakpos. ?
ld       r  DestP+RA1   ;Destination Position
            1000        ;100mm = 1000*1/10mm
set      f  FLdDA+FA1   ;Load Destination Absolute
ld       r  BrkP+RA1    ;Break Position
            700         ;70mm
set      f  FLdBPA+FA1  ;Load Break Position Absolute
ld       r  StaFRR+RA1  ;Status Flag Reset Register
            0           ;reset all Status Flag
set      f  FResSF+FA1  ;Reset Status Flag
set      f  FStart+FA1  ;Start motion
EST
;----------------------
ST       2              ;modify velocity
         I  1           :breakpos. ?
         O  2           ;on dest. ?
ld       r  Veloc+RA1   ;Velocity
            200         ;0.02m/s
set      f  FLdVA+FA1   ;Load Velocity Absolute
set      f  FStart+FA1  ;Update Velocity
EST
;----------------------
ST       3              ;ld timer
         I  2           ;on dest. ?
         O  3           ;timer = 0 ?
ld       t  0           ;timer 0
            30          ;3 sec.
EST
;----------------------
ST       4              ;backwards
         I  3           ;timer = 0 ?
         O  4           ;breakpos. ?
ld       r  DestP+RA1   ;Destination Position
            0           ;Position 0
set      f  FLdDA+FA1   ;Load Destination Absolute
set      f  FResSF+FA1  ;Reset Status Flag
set      f  FStart+FA1  ;Start motion
EST
;----------------------
ST       5              ;modify velocity
         I  4           :breakpos. ?
         O  5           ;on dest. ?
ld       r  Veloc+RA1   ;Velocity
            500         ;0.05m/s
set      f  FLdVA+FA1   ;Load Velocity Absolute
set      f  FStart+FA1  ;Update Velocity
EST
```

```
;----------------------
TR       0              ;input 8="1"
         I  0           ;empty
         O  1           ;forward
sth      i  8           ;motion free ?
ETR
;----------------------
TR       1              ;breakpos. ?
         I  1           ;forward
         O  2           ;modify velocity
sth      f  BrkPos+FA1  ;Break Position reached ?
ETR
;----------------------
 TR      2              ;on dest. ?
         I  2           ;modify velocity
         O  3           ;ld timer
sth      f  OnDest+FA1  ;On Destination ?
ETR
;----------------------
 TR      3              ;timer = 0 ?
         I  3           ;ld timer
         O  4           ;backwards
stl      t  0           ;timer = 0 ?
ETR
;----------------------
TR       4              ;breakpos. ?
         I  4           ;forward
         O  5           ;modify velocity
sth      f  BrkPos+FA1  ;Break Position reached ?
ETR
;----------------------
TR       5              ;on dest. ?
         I  5           ;modify velocity
         O  0           ;empty
sth      f  OnDest+FA1  ;On Destination ?
ETR
;----------------------
ESB
```

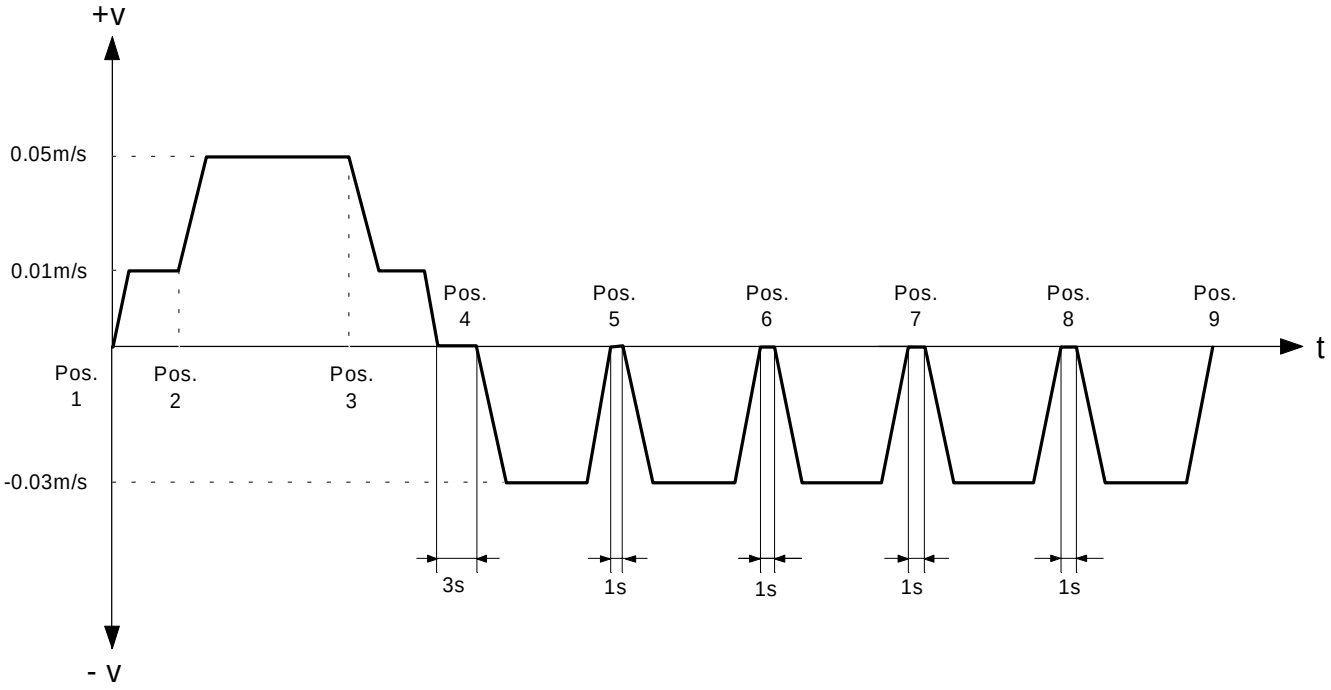
## 9.3 Beispiel 3

Das Beispiel ist aufbauend auf dem Beispiel 2.
Die Erweiterung beinhaltet ein umfangreicheres Fahrprogramm mit einem Automatik- und einem Manuellsteuerteil.

**Aufgabenstellung**

Hardware: Workshopmodelle wie Beispiel 1.

Im Automatikbetrieb soll folgender Bewegungsablauf ausgeführt werden:




| | | |
|---|---|---|
| Position 1: | 0mm | (Startposition) |
| Position 2: | 30mm | (Breakposition |
| Position 3: | 110mm | (Breakposition) |
| Position 4: | 150mm | (Zielposition) |
| Position 5: | 120mm | (Zielposition) |
| Position 6: | 90mm | (Zielposition) |
| Position 7: | 60mm | (Zielposition) |
| Position 8: | 30mm | (Zielposition) |
| Position 9: | 0mm | (Zielposition) |

Bedingungen:

- Die Beschleunigung soll konstant 0.01m/s$^2$ betragen.
- Zu jedem beliebigen Zeitpunkt soll mit einem Schalter zwischen Automatik- und Manuellbetrieb umgeschaltet werden können.
- Nachdem das Automatikprogramm mit einer Taste gestartet wurde, läuft es endlos bis in den Manuellbetrieb umgeschaltet oder mit der Stoptaste abgebrochen wird.
- Die Achse soll mit einer Taste zu einem beliebigen Zeitpunkt gestoppt werden können.

Manuellbetrieb:

- Durch Tastendruck soll mit einer Geschwindigkeit von ±0.02m/s in die postive respektive negative Bewegungsrichtung gefahren werden. Sobald die Taste losgelassen wird, soll mit der definierten Stopart gestoppt werden.
- Durch Tastendruck soll die Istposition als Nullposition definiert werden.

Belegung der Eingänge:

| | |
|---|---|
| Eingang 0 | —> Umschaltung Automatik/Manuellbetrieb (0/1) |
| Eingang 8 | —> Start Automatikprogramm |
| Eingang 9 | —> Manuell vorwärts |
| Eingang 10 | —> Manuell rückwärts |
| Eingang 11 | —> Definiere Nullposition |
| Eingang 15 | —> Stop |

## Lösung

### a) Installation der Software

Wir gehen davon aus, dass die Hard- und Software wie bei Beispiel 2 bereits installiert ist.
(Die Dateien H3DEF.SRC und H3FB.SRC können ohne Änderung übernommen werden.)

### b) Anwenderprogramm

#### 1. Initialisierung im XOB 16

Kann ohne Änderung von Beispiel 2 übernommen werden .

**2. Zyklische Achsenbearbeitung**

Als Erweiterung zu Beispiel 2 wird hier zusätzlich die Betriebsartensteuerung realisiert. Damit zu einem beliebigen Zeitpunkt zwischen Automatik- und Manuellbetrieb umgeschaltet werden kann, wird bei Betätigen des Schalters an Eingang 0 das GRAFTEC-Programm bei Step 0 neu gestartet. Das bedeutet, bei der Umschaltung von Automatik- in den Manuellbetrieb wird die momentane Bewegung noch zu Ende geführt bis der Antrieb stoppt.
Bei Betätigen der Stoptaste an Eingang 15 wird der Antrieb gestoppt und anschliessend das GRAFTEC-Programm neu gestartet bei Step 0.

**3. Definition des Fahrprogrammes**

Definition des Fahrprogrammes entsprechend der Aufgabenstellung in einer GRAFTEC-Struktur (SB 0).

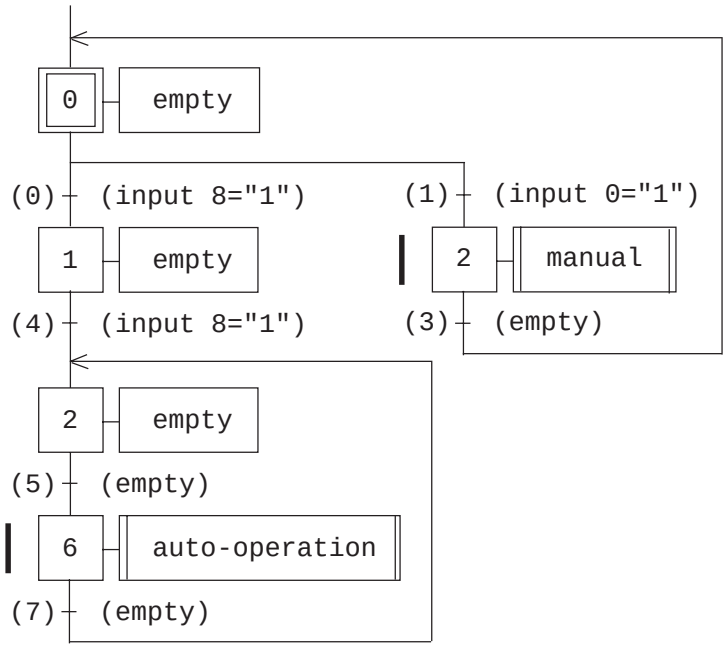
Automatikprogramm:
Da für das Anfahren der Positionen 5 bis 9 jedes Mal die gleiche Strecke mit der gleichen Geschwindigkeit gefahren wird, können diese Bewegungen in einer Schleife (Zielposition wird relativ geladen) programmiert werden.

Auf den folgenden Seiten ist die Source-Datei (BSP03.SRC) des Anwenderprogramms für das Beispiel wiedergegeben.

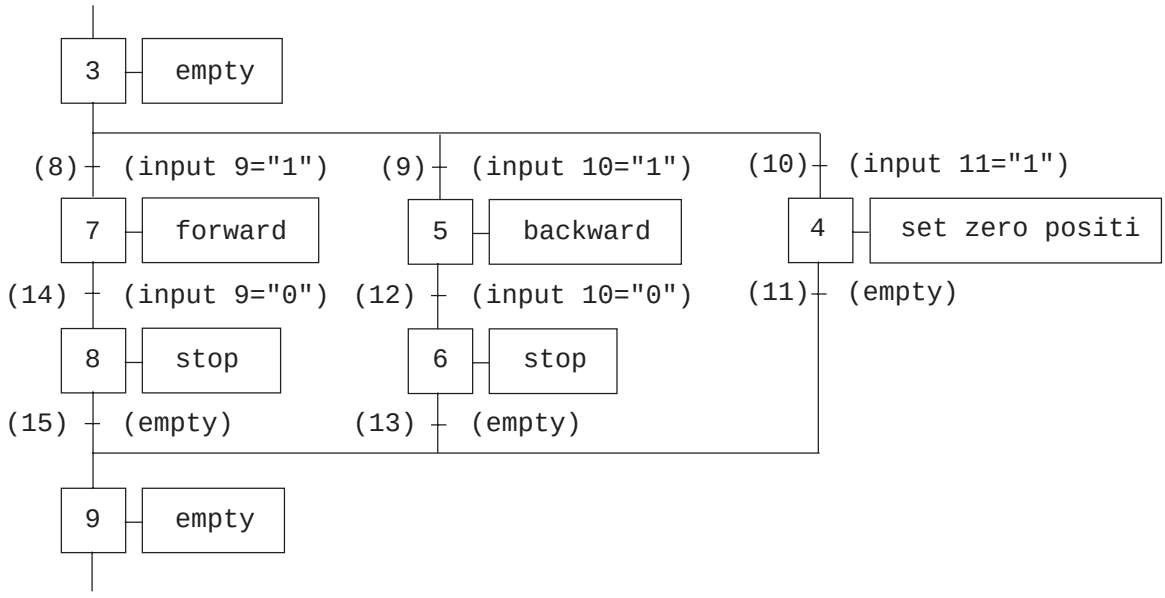
```
; Demo programm for the motion control module PCD4.H3..
; ======================================================
; Name  : BSP03.SRC
; U. Jäggi 27.08.90

$ include H3DEF.SRC

        ;********************** Cold-Start (Initialisation)
        XOB         16
        ;----------------------- Cold-Start Definitions
        ;----------------------- loading of the initialisation registers

        Ld       R  MCW+RA1     ;Motion Control Word
                    41BH        ;Stop smothly, Position mode
                                ;only statusflag (Pos.error)
        Ld       R  PosEr+RA1   ;Position Error
                    2000        ;4 * 500 pulses
        Ld       R  KProp+RA1   ;Proportional factor
                    150
        Ld       R  KInt+RA1    ;Integral factor
                    50
        Ld       R  KDer+RA1    ;Derivative factor
                    50
        Ld       R  IntL+RA1    ;Integration Limit
                    500
        Ld       R  SampI+RA1   ;Sampling Interval
                    15          ;5.46ms
        Ld       R  MCFac+RA1   ;Motion Control Factor
                    100.0       ;100 Imp./1/10mm
        Ld       R  Veloc+RA1   ;Velocity
                    500         ;0.05m/s
        Ld       R  Accel+RA1   ;Acceleration
                    100         ;0.01m/s2

        ;-----------------------
        CFB         AxInit      ;Axis Initialisation
                    1           ;X axis
                    RA1
                    FA1
                    IMode       ;Initialisation mode: Analog/PWM

        ;-----------------------  End XOB 16
        EXOB
```

```
        ;**********************  Cyclic program
        COB         0
                    0
        ;-----------------------
        CFB         AxHndlg     ;Axis Handling
                    1           ;X axis
                    RA1
                    FA1
        ;-----------------------
        SET      F  FRdAP+FA1   ;Read Actual Position
        SET      F  FRdSV+FA1   ;Read Setpoint Velocity
        ;-----------------------
        SET      F  DplM+FA1    ;Diplay Mode = 2*6 digits
        CFB         DP14        ;Display on PCA2.D14
                    1           ;Axis 1
                 F  DplM+FA1    ;Display Mode
                 R  RActP+RA1   ;Actual Position
                 R  RSetV+RA1   ;Setpoint Velocity
        ;-----------------------

        CSB         0           ;Call of the motion control program

        ;-----------------------
        STH      I  0           ;Manual operation mode
        DYN      F  0
        RSB      H  0           ;Restart SB
                    0           ;at Step 0
        STL      I  0           ;Automatic operation mode
        DYN      F  1
        RSB      H  0           ;Restart SB
                    0           ;at Step 0
        STH      I  15          ;Stop switch
        DYN      F  15
        JR       L  END
        SET      F  FStop+FA1   ;Stop X axis
        RSB         0           ;Restart SB
                    0           ;at Step 0

        ;-----------------------;End cyclic program
END:    ECOB
```

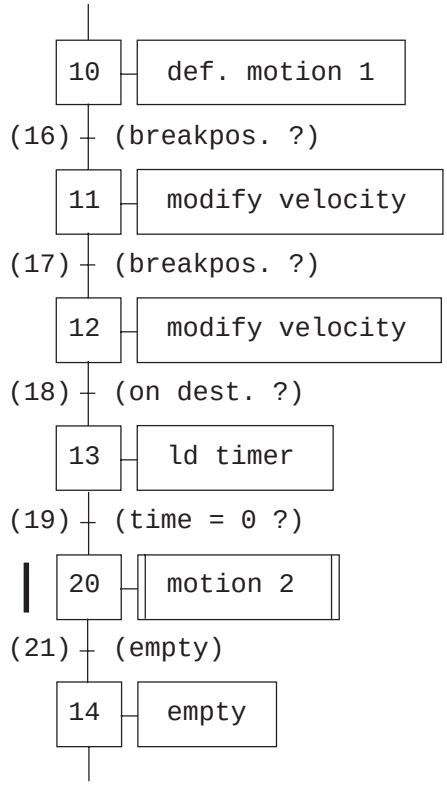
```
*** SAIA PCD GRAFTEC EDITOR $113 ***                          PAGE: 1
FILE: BSP03.GLS  (29.08.90  11.08)          PRODUCED: 29.08.90  11.11
***   SAIA AG   -   CH-3280 MURTEN   ***
SB-NUMBER:  0
PAGE-NB:  0
```



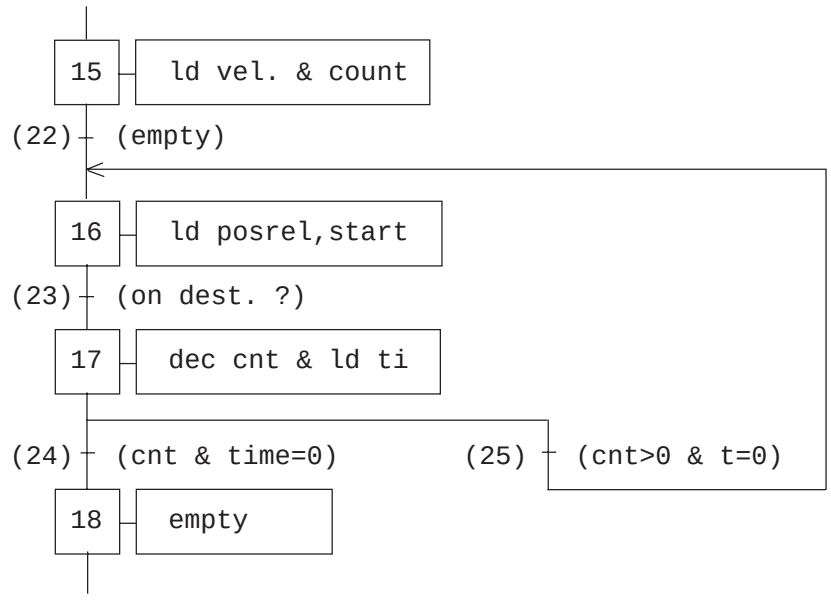

PAGE-NB:  2   manual



PAGE-NB: 6 auto-operation




PAGE-NB: 20 motion 2



```
;********************** Motion control program
SB       0
;-----------------------
IST      0              ;empty
         I  3           ;empty
         O  0           ;input 0="0"
EST
;----------------------
ST       1              ;empty
         I  0           ;input 0="0"
         O  4           ;input 8="1"
EST
;-----------------------
ST       2              ;empty
         I  4           ;input 8="1"
         I  7           ;empty
         O  5           ;empty
EST
;----------------------
ST       3              ;empty
         I  1           ;input 0="1"
         O  8           ;input 9="1"
         O  9           ;input 10="1"
         O  10          ;input 11="1"
EST
;-----------------------
ST       4              ;set zero position
         I  10          ;input 11="1"
         O  11          ;empty
set      f  FSetZP+FA1  ;set zero position X axis
EST
;-----------------------
ST       5              ;backward
         I  9           ;input 10="1"
         O  12          ;input 10="0"

ld       r  Veloc+RA1   ;velocity
            200         ;0.02m/s
set      f  FLdVA+FA1   ;load velocity
set      f  FBackW+FA1  ;backward X axis
EST
;-----------------------
ST       6              ;stop
         I  12          ;input 10="0"
         O  13          ;empty
set      f  FStop+FA1   ;stop X axis
EST
;-----------------------
ST       7              ;forward
         I  8           ;input 9="1"
         O  14          ;input 9="0"

ld       r  Veloc+RA1   ;velocity
            200         ;0.02m/s
set      f  FLdVA+FA1   ;load velocity
set      f  FForw+FA1   ;forward X axis
EST
```

```
;-----------------------
ST      8             ;stop
        I  14         ;input 9="0"
        O  15         ;empty
set     f  FStop+FA1  ;stop X axis
EST
;-----------------------
ST      9             ;empty
        I  15         ;empty
        I  13         ;empty
        I  11         ;empty
        O  3          ;empty
EST
;-----------------------
ST      10            ;def. motion 1
        I  5          ;empty
        O  16         ;breakpos. ?
ld      r  Veloc+RA1  ;Velocity
           100        ;0.01m/s
set     f  FLdVA+FA1  ;Load Velocity Absolute
ld      r  DestP+RA1  ;Destination Position
           1500       ;150mm
set     f  FLdDA+FA1  ;Load Destination Absolute
ld      r  BrkP+RA1   ;Break Position
           300        ;30mm
set     f  FLdBPA+FA1 ;Load Break Position Absolute
ld      r  StaFRR+RA1 ;Status Flag Reset Register
           0          ;reset all Status Flag
set     f  FResSF+FA1 ;Reset Status Flag
set     f  FStart+FA1 ;Start motion
EST
;-----------------------
ST      11            ;modify velocity
        I  16         ;breakpos. ?
        O  17         ;breakpos. ?
ld      r  Veloc+RA1  ;Velocity
           500        ;0.05m/s
set     f  FLdVA+FA1  ;Load Velocity Absolute
ld      r  BrkP+RA1   ;Break Position
           1100       ;110mm
set     f  FLdBPA+FA1 ;Load Break Position Absolute
set     f  FResSF+FA1 ;Reset Status Flag
set     f  FStart+FA1 ;Start motion
EST
;-----------------------
ST      12            ;modify velocity
        I  17         ;breakpos. ?
        O  18         ;on dest. ?
ld      r  Veloc+RA1  ;Velocity
           100        ;0.01m/s
set     f  FLdVA+FA1  ;Load Velocity Absolute
set     f  FStart+FA1 ;Start motion
EST
```

```
;----------------------
ST      13              ;ld time
        I  18           :on dest. ?
        O  19           ;time = 0 ?
ld      t  0
           20
EST
;-----------------------
ST      14              ;empty
        I  21           ;empty
        O  7            ;empty
EST

;----------------------
ST      15              ;ld vel. & counter
        I  19           :time = 0 ?
        O  22           ;empty
ld      r  Veloc+RA1    ;Velocity
           300          ;0.03m/s
set     f  FLdVA+FA1    ;Load Velocity Absolute
ld      c  100
           5
EST
;----------------------
ST      16              ;ld  posrel,start
        I  22           :empty
        I  25           :cnt>0 & t=0
        O  23           ;on dest. ?
ld      r  DestP+RA1    ;Destination Position
           -300         ;-30mm
set     f  FLdDR+FA1    ;Load Destination Relative
set     f  FStart+FA1   ;Start motion
EST
;----------------------
ST      17              ;dec cnt & ld timer
        I  23           :on dest. ?
        O  24           ;cnt & time=0
        O  25           ;cnt>0 & t=0
dec     c  100
ld      t  0
           10
EST
;----------------------
ST      18              ;empty
        I  24           ;cnt & time=0
        O  21           ;empty
EST
;----------------------
```

```
;----------------------
TR       0              ;input 0="0"
         I  0           ;empty
         O  1           ;empty
stl      i  0           ;auto op. ?
ETR
;----------------------
TR       1              ;input 0="1"
         I  0           ;empty
         O  3           ;empty
sth      i  0           ;manual op. ?
ETR
;----------------------
TR       2              ;manual
         I  3           ;empty
         O  9           ;empty
ETR
;----------------------
TR       3              ;empty
         I  9           ;empty
         O  0           ;empty
ETR
;----------------------
 TR      4              ;input 8="1"
         I  1           ;empty
         O  2           ;empty
sth      i  8           ;start auto op. ?
ETR
;----------------------
TR       5              ;empty
         I  2           ;empty
         O  10          ;def. motion 1
ETR
;----------------------
TR       6              ;auto-operation
         I  10          ;def. motion 1
         O  14          ;empty
ETR
;----------------------
TR       7              ;empty
         I  14          ;empty
         O  2           ;empty
ETR
;----------------------
TR       8              ;input 9="1"
         I  3           ;empty
         O  7           ;forward
sth      i  9           ;manual forward ?
ETR
;----------------------
TR       9              ;input 10="1"
         I  3           ;empty
         O  5           ;backward
sth      i  10          ;manual backwards ?
ETR
```

```
;----------------------
TR       10             ;input 11="1"
         I  3           ;empty
         O  4           ;set zero position
sth      i  11          ;define zero pos. ?
ETR
;----------------------
TR       11             ;empty
         I  4           ;set zero position
         O  9           ;empty
ETR
;----------------------
TR       12             ;input 10="0"
         I  5           ;backward
         O  6           ;stop
stl      i  10          ;end zero pos. ?
ETR
;----------------------
TR       13             ;empty
         I  6           ;stop
         O  9           ;empty
ETR
;----------------------
TR       14             ;input 9="0"
         I  7           ;forward
         O  8           ;stop
stl      i  9           ;end manual forward ?
ETR
;----------------------
TR       15             ;empty
         I  8           ;stop
         O  9           ;empty
ETR
;----------------------
TR       16             ;breakpos. ?
         I  10          ;def. motion 1
         O  11          ;modify velocity
sth      f  BrkPos+FA1  ;Break Position reached ?
ETR
;----------------------
TR       17             ;breakpos. ?
         I  11          ;modify velocity
         O  12          ;modify velocity
sth      f  BrkPos+FA1  ;Break Position reached ?
ETR
;----------------------
TR       18             ;on dest. ?
         I  12          ;modify velocity
         O  13          ;ld time
sth      f  OnDest+FA1  ;on destination ?
ETR
;----------------------
TR       19             ;time = 0 ?
         I  13          ;ld time
         O  15          ;ld vel. & counter
stl      t  0
ETR
```

```
;----------------------
TR       20             ;motion 2
         I  15          ;ld vel. & counter
         O  18          ;empty
ETR
;----------------------
TR       21             ;empty
         I  18          ;empty
         O  14          ;empty
ETR
;----------------------
TR       22             ;empty
         I  15          ;ld vel. & counter
         O  16          ;ld posrel,start
ETR
;----------------------
TR       23             ;on dest ?
         I  16          ;ld posrel,start
         O  17          ;dec cnt & ld timer
sth      f  OnDest+FA1  ;on destination ?
ETR
;----------------------
TR       24             ;cnt & time=0
         I  17          ;dec cnt & ld timer
         O  18          ;empty
stl      c  100
anl      t  0
ETR
;----------------------
TR       25             ;cnt>0 & t=0
         I  17          ;dec cnt & ld timer
         O  16          ;ld posrel,start
sth      c  100
anl      t  0
ETR
;----------------------

ESB
```

# 10. Übersicht der Befehle und Symbole

## Übersicht der Funktionen ausführbar mit FB "AxHndlg" mit Angabe der Ein / Ausgabe-Werte

### Betriebsdefinitionen

| FB Ausführungs Flag | Bezeichnung, Funktion | Abarb-eitungs-Zeit | Ausführb. während Bewegung | Seite | Ein/Ausgabe-Werte | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Symbol | Typ | Format | Bezeichnung |
| FSelOM | Select Operation Mode | 1,9ms | ja | 7-24 | MCW | R | Binär | Motion Control Word |
| FSetPE | Set Position Error | 1,9ms | ja | 7-25 | PosEr<br>MCW | R<br>R | Integer<br>Binär | Position Error<br>Motion Control Word |

### Fahrbefehle

| FB Ausführungs Flag | Bezeichnung, Funktion | Abarb-eitungs-Zeit | Ausführb. während Bewegung | Seite | Ein/Ausgabe-Werte | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Symbol | Typ | Format | Bezeichnung |
| FStart | Start Motion | 1ms | ja | 7-43 | -- | | | |
| FStop | Stop Motion | 2,7ms | ja | 7-44 | MCW | R | Binär | Motion Control Word |
| FMotOff | Motor Off | 1,8ms | ja | 7-46 | -- | | | |
| FSStepF | Single Step Forward | 4,5ms | nein | 7-47 | -- | | | |
| FSStepB | Single Step Backwards | 4,5ms | nein | 7-48 | -- | | | |
| FForw | Forward with def. Velocity | 4,5ms | ja | 7-49 | -- | | | |
| FBackw | Backwards with def. Velocity | 4,5ms | ja | 7-50 | -- | | | |

## Parametereingaben für das Geschwindigkeitsprofil

| FB Ausführungs Flag | Bezeichnung, Funktion | Abarb-eitungs-Zeit | Ausführb. während Bewegung | Seite | Ein/Ausgabe-Werte | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Symbol | Typ | Format | Bezeichnung |
| FLdDR | Load Destination Position Relative | 4,1ms | ja | 7-30 | DestP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Destination Position<br>Motion Control Factor |
| FLdDA | Load Destination Position Absolute | 4,1ms | ja | 7-28 | DestP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Destination Position<br>Motion Control Factor |
| FLdVR | Load Velocity Relative | 4,5ms | ja | 7-34 | Veloc<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Velocity<br>Motion Control Factor |
| FLdVA | Load Velocity Absolute | 4,5ms | ja | 7-32 | Veloc<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Velocity<br>Motion Control Factor |
| FLdAR | Load Acceleration Relative | 7,2ms | bedingt | 7-38 | Accel<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Acceleration<br>Motion Control Factor |
| FLdAA | Load Acceleration Absolute | 7,2ms | bedingt | 7-36 | Accel<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Acceleration<br>Motion Control Factor |
| FLdRP | Load Regulator Parameter | 5,2ms | ja | 7-40 | KProp<br>KInt<br>KDer<br>IntL<br>SampI | R<br>R<br>R<br>R<br>R | Integer<br>Integer<br>Integer<br>Integer<br>Integer | Proportional Factor<br>Integral Factor<br>Derivative Factor<br>Integration Limit<br>Sampling Interval Derivative Term |
| FUpDRP | Up Date Regulator Parameters | 0,9ms | ja | 7-42 | -- | | | |

## Lesebefehle für Daten

| FB Ausführungs Flag | Bezeichnung, Funktion | Abarb-eitungs-Zeit | Ausführb. während Bewegung | Seite | Ein/Ausgabe-Werte | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Symbol | Typ | Format | Bezeichnung |
| FRdAP | Read Actual Position | 4,3ms | ja | 7-51 | RActP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Actual Position<br>Motion Control Factor |
| FRdSP | Read Setpoint Position | 4,3ms | ja | 7-52 | RSetP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Setpoint Position<br>Motion Control Factor |
| FRdAV | Read Actual Velocity | 2,8ms | ja | 7-53 | RActV<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Actual Velocity<br>Motion Control Factor |
| FRdSV | Read Setpoint Velocity | 3,7ms | ja | 7-54 | RSetV<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Setpoint Velocity<br>Motion Control Factor |
| FRdITS | Read Integration Term Sum | 1,9ms | ja | 7-55 | RIntTS | R | Integer | Integration Term Sum |
| FRdIP | Read Index Position | 4,3ms | ja | 7-56 | RIndP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Index Position<br>Motion Control Factor |
| FRdSR | Read Signal Register | 1,8ms | ja | 7-57 | RSigB | R | Binär | Signalisation Bits |

## Hilfsfunktionen

| FB Ausführungs Flag | Bezeichnung, Funktion | Abarb-eitungs-Zeit | Ausführb. während Bewegung | Seite | Ein/Ausgabe-Werte | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Symbol | Typ | Format | Bezeichnung |
| FResSF | Reset Status Flag | 1,8ms | ja | 7-60 | StaFRR | R | Binär | Status Flag Reset Register |
| FLdBPR | Load Break Position Relative | 3,3ms | ja | 7-64 | BrkP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Break Position<br>Motion Control Factor |
| FLdBPA | Load Break Position Absolute | 3,3ms | ja | 7-62 | BrkP<br>MCFac | R<br>R | Integer<br>Fl. Punkt | Break Position<br>Motion Control Factor |
| FSetIP | Set Index Position | 0,9ms | ja | 7-66 | -- | | | |
| FSetZP | Set Zero Position | 0,9ms | ja | 7-67 | -- | | | |

Notizen:

Absender:

Firma
Abteilung
Name
Adresse

Tel.

Datum

An:

SAIA-Burgess Electronics AG
Bahnhofstrasse 18
CH-3280 Murten (Schweiz)
http://www.saia-burgess.com

GB: Electronic Controllers

Handbuch PCD4.H3xx
Positioniermodule für Servoantriebe

Falls Sie Vorschläge zu SAIA® PCD zu machen oder Fehler in diesem Handbuch gefunden haben, sind wir Ihnen für einen kurzen Bericht dankbar.

Ihre Vorschläge: