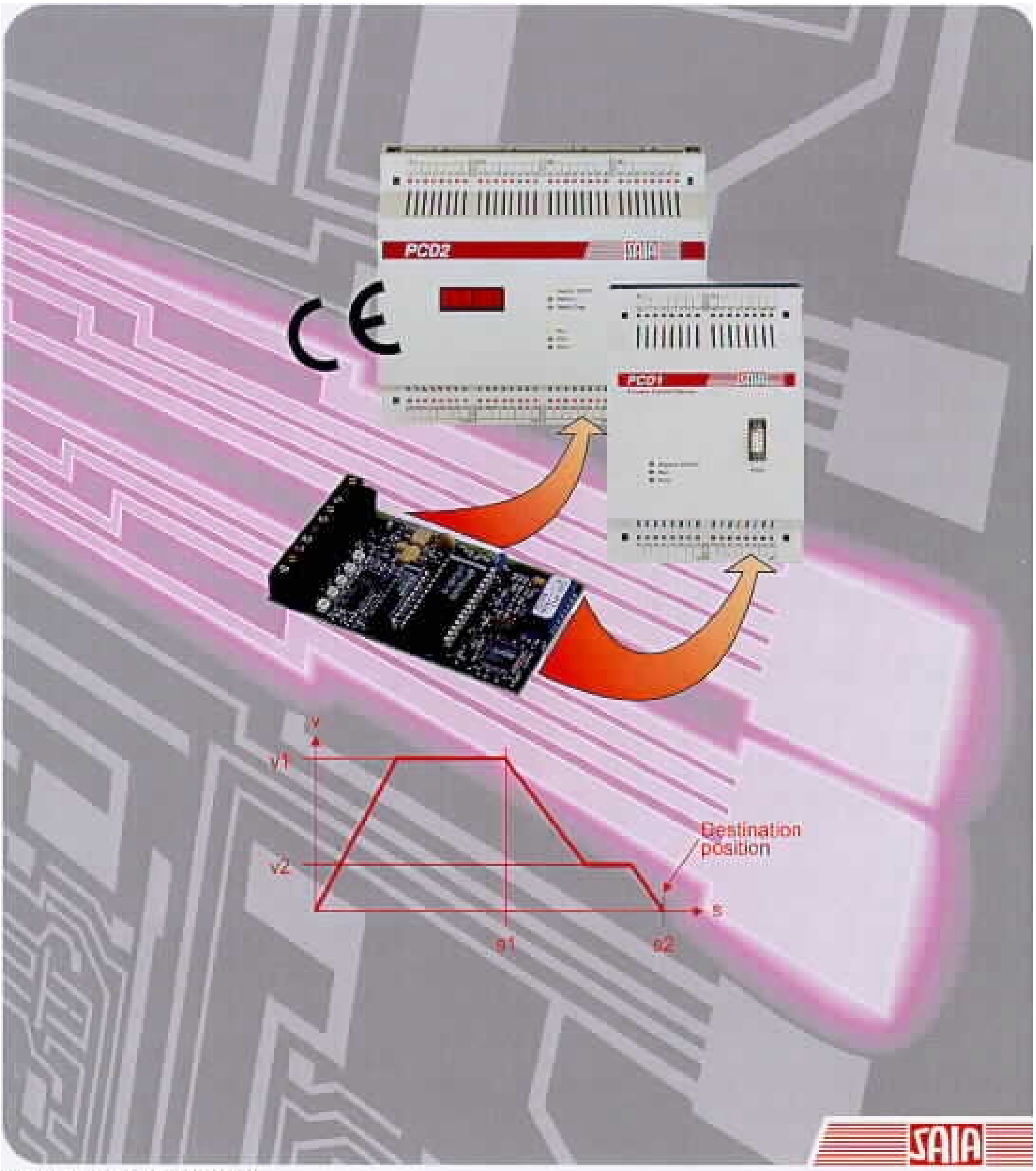
SAIA-Burgess Electronics

SWITCHES • MOTORS • CONTROLLERS

SAIA®PCD

Process Control Devices

# PCD2.H31x Motion control modules for servo drives



English edition 26/762 E2

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

BA: Electronic Controllers Telephone 026 / 672 72 72
Telefax 026 / 672 74 99

## SAIA-Burgess Companies

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

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

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

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

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

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

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

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

## Representatives

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

**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

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

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

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

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

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

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

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

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

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

## After sales service

**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

# Motion control modules for servo drives

# PCD2.H31x

SAIA-Burgess Electronics Ltd. 1999.  All rights reserved.
Edition 26/762 E2 - 11.99

Subject to technical changes

# Updates

**Manual : PCD2.H31x - Motion control modules for servo drives - Edition E2**

| Date | Chapter | Page | Description |
|---|---|---|---|
| 12.05.2000 | Appendix A | A-36 | divers corrections |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Contents

Notes

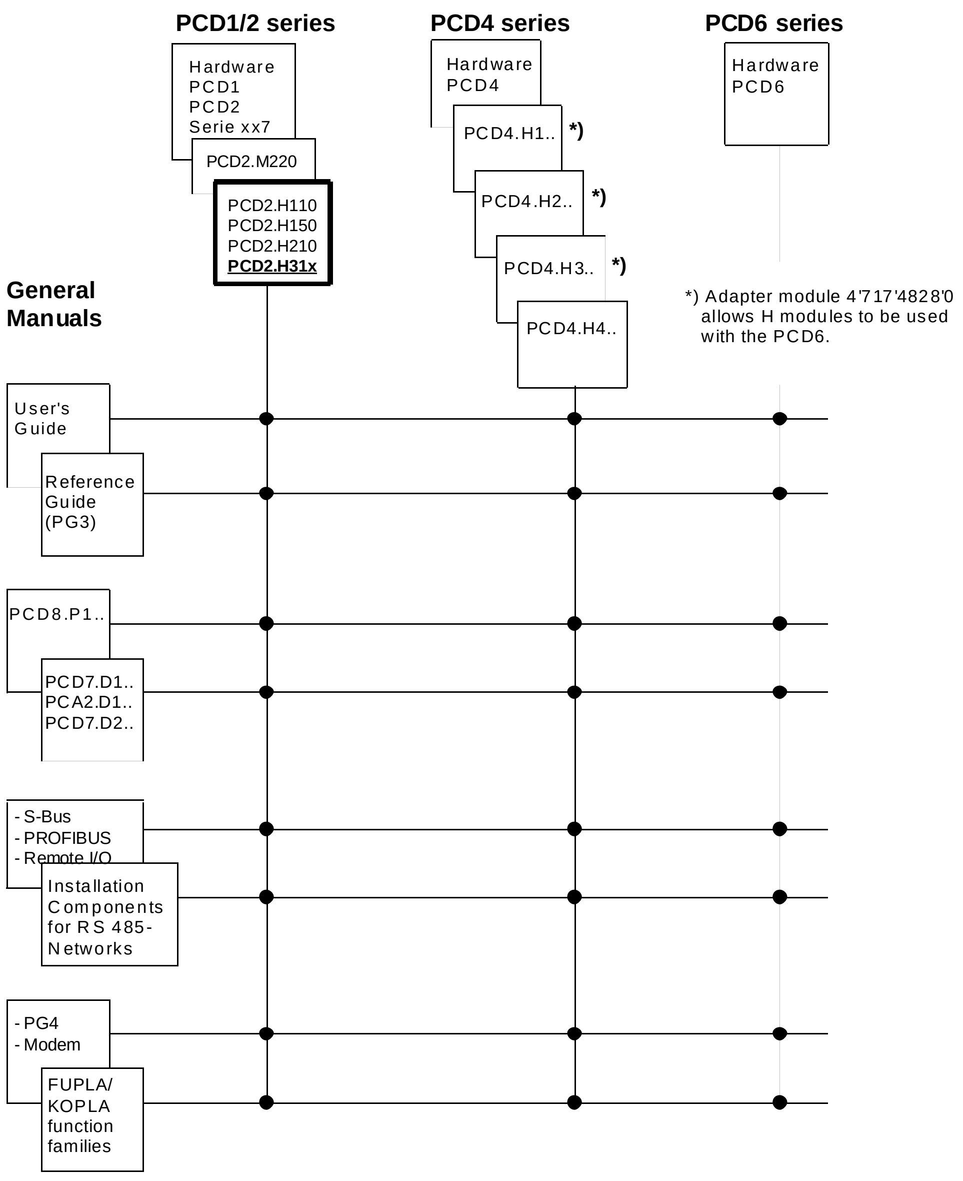
**Important note:**

Many detailed manuals have been created to ensure perfect operation of the SAIA® PCD. These are for use by technically qualified staff, who may also have successfully completed our workshops.

The diverse performance features of the SAIA® PCD best become apparent only by following closely all the information and guidelines contained in these manuals regarding assembly, wiring, programming and commissioning.

Doing this will enable you to join the large group of enthusiastic SAIA® PCD users.

## Summary


PCD1/2 series
PCD4 series
PCD6 series
Hardware PCD1 PCD2 Serie xx7
PCD2.M220
PCD2.H110 PCD2.H150 PCD2.H210 PCD2.H31x
Hardware PCD4
PCD4.H1.. *)
PCD4.H2.. *)
PCD4.H3.. *)
PCD4.H4..
Hardware PCD6
*) Adapter module 4'717'4828'0 allows H modules to be used with the PCD6.
General Manuals
User's Guide
Reference Guide (PG3)
PCD8.P1..
PCD7.D1.. PCA2.D1.. PCD7.D2..
- S-Bus - PROFIBUS - Remote I/O
Installation Components for RS 485-Networks
- PG4 - Modem
FUPLA/ KOPLA function families

## Reliability and safety of electronic controllers

SAIA-Burgess Electronics Ltd. is a company which devotes the greatest care to the design, development and manufacture of its products:

- state-of-the-art technology
- compliance with standards
- ISO 9001 certification
- international approvals: e.g. Germanischer Lloyd, UL,
  Det Norske Veritas, CE mark ...
- choice of high-quality componentry
- quality control checks at various stages of production
- in-circuit tests
- run-in (burn-in at 85°C for 48h)

Despite every care, the excellent quality which results from this does have its limits. It is therefore necessary, for example, to reckon with the natural failure of components. For this reason SAIA-Burgess Electronics Ltd. provides a guarantee according to the "General terms and conditions of supply".

The plant engineer must in turn also contribute his share to the reliable operation of an installation. He is therefore responsible for ensuring that controller use conforms to the technical data and that no excessive stresses are placed on it, e.g. with regard to temperature ranges, overvoltages and noise fields or mechanical stresses.

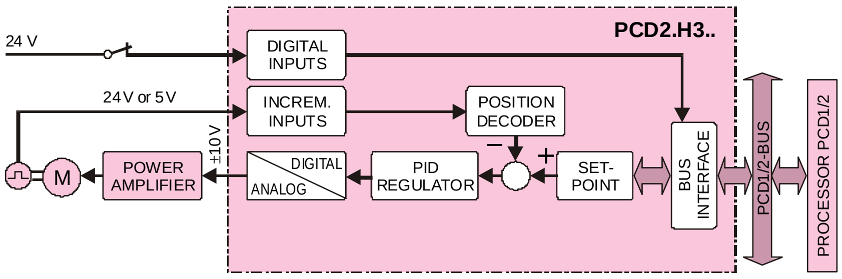
In addition, the plant engineer is also responsible for ensuring that a faulty product in no case leads to personal injury or even death, nor to the damage or destruction of property. The relevant safety regulations should always be observed. Dangerous faults must be recognized by additional measures and any consequences prevented. For example, outputs which are important for safety should lead back to inputs and be monitored from software. Consistent use should be made of the diagnostic elements of the PCD, such as the watchdog, exception organization blocks (XOB) and test or diagnostic instructions.

If all these points are taken into consideration, the SAIA® PCD will provide you with a modern, safe programmable controller to control, regulate and monitor your installation with reliability for many years.

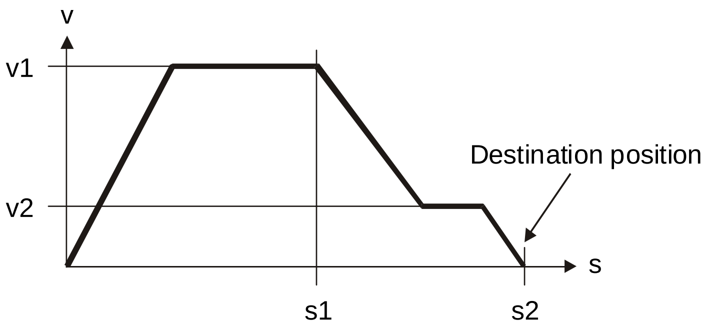
# 1. Introduction

## 1.1 General

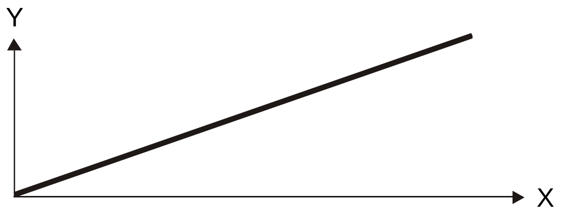
The PCD2.H3.. motion control module is an intelligent I/O module from the PCD2 series. This module is used for positioning an independent axis with a variable speed control drive (servomotor). The servomotor can be any adjustable DC or AC motor having a power stage and incremental shaft encoder for the registration of position and speed.



Block diagram of a servo drive for 1 axis



Position control operation with constant acceleration, delay and slow advance to approach destination position



Coordinated, quasi-synchronous operation of 2 axes

## 1.2 Function and application

The ..H3.. module is used to position a single axis with variable speed control DC or AC servomotors. This requires the drive unit to have a power stage and incremental shaft encoder for capturing position or speed. In a PCD1 system up to 4 motion control modules (4 axes) and in a PCD2 up to 8 ..H3.. modules (8 axes) can be plugged anywhere on the I/O bus and operated together with all base units.

The motion control module contains a single-chip processor that independently directs and PID controls every movement according to parameters supplied by the user program (velocity, acceleration and destination position). This enables each axis to be controlled independently. It is possible to program the linkage of several axes (point-to-point) in co-ordinated, quasi-synchronous mode. Linear motion is thereby achieved with cartesian axes.

PCD2.H31x motion control modules should be plugged into the PCD2.M1xx base unit and not the PCD2.C1xx expansion unit. This is to take account of the relatively high current consumption.

## 1.3 Main characteristics

- Ideal for compact or economical machines
- PID controlled regulation of servomotor position and speed
- Velocity, destination position and PID parameters can be modified, even during motion
- Analogue ±10V output with 12 bits inc. sign bit for triggering the motor power stage
- Digital input for reference switch with the .. H310
- Encoder signal inputs of 24 VDC (source operation) or 5 V/RS 422 (antivalent line driver)
- Other axis-specific inputs and outputs (such as the limit switches, the reference switch for the PCD2.H311 module, ENABLE for the driver) must be monitored or controlled with a standard I/O module (e.g. PCD2.B100, PCD2.E110/E111 or PCD2.A410).

## 1.4 Typical areas of use

- Handling robots
- Automatic placement and assembly machines
- Automatic palletizers
- Packing machines
- Sheet metal working machines
- Automatic drilling machines
- etc.

## 1.5 Programming

The availability of practical function blocks (FBs and FBoxes) means that only the desired parameters need to be entered for the various motion and travel functions. The present detailed manual contains descriptions of each function block, complemented with practical application examples. Programming is via the standard PG4 programming tool from version V2.0.70, either in IL (Instruction List) or graphically (FUPLA).

**Initialization instruction**

| | |
|---|---|
| INIT | FB. When it is called, 11 parameters are given with it. |
| | See section 7.1.2 and appendix A, pages A1/2 |

**Execution instruction**

| | |
|---|---|
| EXEC | FB, always including 3 parameters. Over 30 different instructions can be executed. |
| | See section 7.1.2 and appendix A, pages A5/40 |

**Home instruction**

| | |
|---|---|
| HOME | FB for automatically seeking the reference switch. This FB has 7 parameters. |
| | See section 7.1.2 and appendix A, pages A3/4 |

**Diagnostics and error handling**

Recognition of incorrect FB parameters (diagnostic register)
Timeout monitoring for FB 'Home'.

See chapter 8.

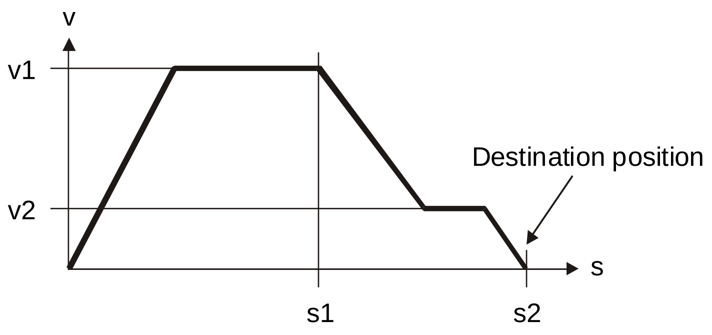
## 1.6 Operating modes

When solving motion control tasks with a servo drive, two different operating modes are fundamentally available:

- position control mode
- speed control mode

In position control mode various parameters are entered (PID factors, acceleration, velocity, etc.) after which a preset destination position is approached in a controlled manner

In speed control mode the desired velocity is attained with a preset rate of acceleration. Travel continues at this velocity in a controlled manner until there is a stop instruction. The desired velocity can be changed even during motion.



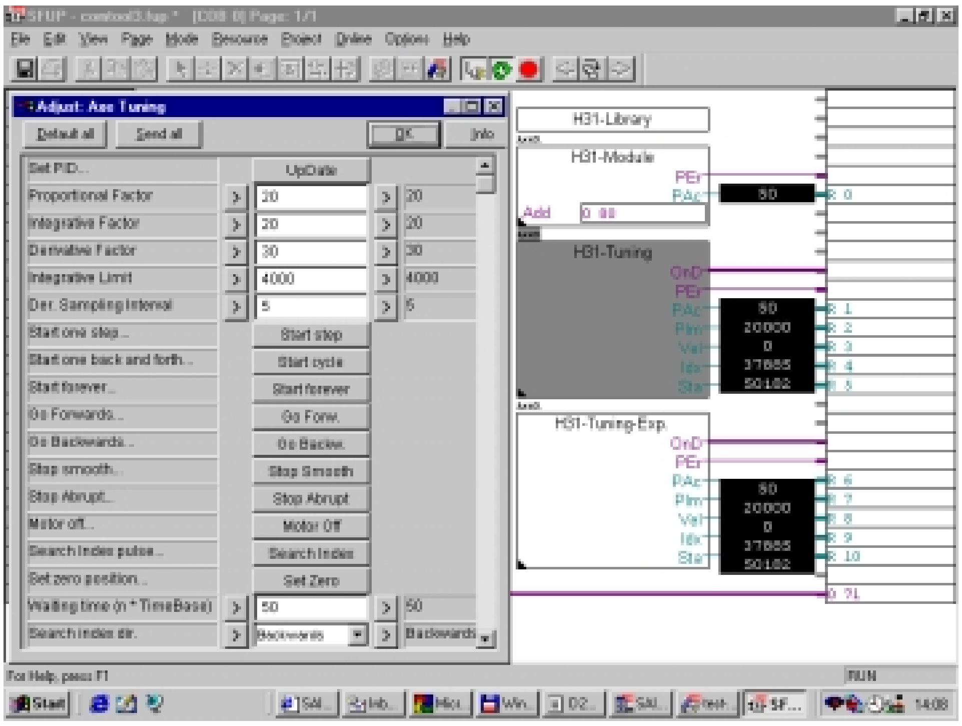
Position control mode with constant acceleration and delay, plus reduced speed on approach to destination position

## 1.7 Commissioning

Convenient commissioning software is available in the form of a FUPLA program. This tool offers a simple method whereby all tests and adjustments required during commissioning can be carried out or checked on-line.

This commissioning tool is available on diskette, order reference: PCD8.H31.

See chapter 10 for more details.



| | | | | |
|---|---|---|---|---|
| Set PID... | | UpDate | | |
| Proportional Factor | > | 20 | > | 20 |
| Integrative Factor | > | 20 | > | 20 |
| Derivative Factor | > | 30 | > | 30 |
| Integrative Limit | > | 4000 | > | 4000 |
| Der. Sampling Interval | > | 5 | > | 5 |
| Start one step... | | Start step | | |
| Start one back and forth... | | Start cycle | | |
| Start forever... | | Start forever | | |
| Go Forwards... | | Go Forw. | | |
| Go Backwards... | | Go Backw. | | |
| Stop smooth... | | Stop Smooth | | |
| Stop Abrupt... | | Stop Abrupt | | |
| Motor off... | | Motor Off | | |
| Search Index pulse... | | Search Index | | |
| Set zero position... | | Set Zero | | |
| Waiting time (n * TimeBase) | > | 50 | > | 50 |
| Search index dir. | > | Backwards | > | Backwards |

Notes

# 2. Technical data

## 2.1 Hardware technical data

**Digital inputs of the PCD2.H310 module**

| | |
|---|---|
| Number of inputs: | 1 encoder A, B, IN<br>1 reference input |
| Input voltage: | typically 24 V |
| "Low" range: | 0 ... +4 V |
| "High" range: | +15 ... +30 V |
| Source operation only (pos. logic) | |
| Input current at 24 VDC: | 6 mA (typical) |
| Circuit type | electronically connected |
| Reaction time | 30 µs |

**Digital inputs of the PCD2.H311 module**

| | |
|---|---|
| Number of inputs: | 1 encoder A, /A, B, /B, IN, /IN<br>(no reference input) |
| Input voltage: | typically 5 V |
| Signal level: | Antivalent inputs according to RS 422 |
| Hysteresis: | max. 200 mV |
| Line termination resistance: | 150 Ω |

**Analogue output for the PCD2.H310/311 modules**

| | |
|---|---|
| Analogue controller output | 12 bit resolution (with sign bit) |
| Short-circuit protection | yes |
| Electrical isolation | no |
| Output voltage *) | ± 10 V, accuracy of adjustment ± 5 mV |
| Logic | positive (positive switching) |
| Minimum load impedance | 3 kΩ |

**5 V supply of 5 V encoder for the PCD2.H311 module**

| | |
|---|---|
| 5 V output | 5 V supply of encoder |
| Short-circuit protection | yes |
| Electrical isolation | no |
| Output voltage | 5 V |
| Max. load current | 300 mA |
| Short-circuit current | 400 mA |

(This current additionally loads the 5 V bus of the PCD1/2)

*) Balancing output voltage is carried out in the factory. You are therefore strongly advised not to adjust the tuning potentiometer.

**Operating conditions**

| | |
|---|---|
| Ambient temperature | Operation: 0 ...+50°C without forced ventilation<br>Storage: -20 ... +85°C |
| Noise immunity | CE mark, compliant with EN 50081-1 and EN 50082-2 |

**LED displays and possibilities for querying from software**

| | |
|---|---|
| Total | 5 |
| LED "A" | Status of encoder input "A" |
| LED "B" | Status of encoder input "B" |
| LED "IN" | Status of index input |
| LED "Ref" | Status of reference switch (H310) |
| LED "Pw 5V" | Encoder 5 V supply (H311) |
| LED "Power" | ± 15 V supply |

Querying from software

| | |
|---|---|
| Input Power (addr. 08) | Enables software monitoring of supplies |
| Input Ref (addr. 11) | Enables the logic status of the reference switch to be queried (H310) |
| Input Pw5V (addr. 11) | Enables software monitoring of the 5 V supply (H311) |
| Input Version (addr. 12) | Enables module type to be queried H310 or H311 (H = H310, L = H311) |

**General**

| | |
|---|---|
| Processor | LM 628 |
| Programming | Based on PCD user program (PG4) and supported by a library of FBs and FBoxes. |

**Ordering details**

| | |
|---|---|
| PCD2.H310 | 1 axis for encoder 24 VDC |
| PCD2.H311 | 1 axis for encoder 5 V/RS 422 |
| PCD9.H31E | Software library with function blocks (FBs) for programming in IL. |
| PCD8.H31 | Commissioning tool in FUPLA |

## 2.2 Electrical specifications

**Internal power consumption** (without encoder)

| | |
|---|---|
| +5 V | typically 125 mA, max. 150 mA |
| Uext | typically 10 mA, max. 15 mA |

**External supply**

**Terminals +/-:** 24 V (19 ... 32 V) smoothed,
max. permissible ripple: 10%

**Digital inputs**

4 or 6 respectively (see section 2.1)

**Analogue output**

1 (see section 2.1)

**Important :**

The maximum current that can be supplied to the 5V is 1600 mA for a PCD2 or 750 mA for a PCD1.

Users of PCD2.H310 and/or PCD2.H311 modules are urged to check the overall current consumption of **all** modules in a PCD2/1 **and** in any C100 or C150 expansion units, and to ensure that this maximum is not exceeded. (The current provided to supply the 5V encoder also comes from this source and must equally be taken into consideration).

When working with an expansion unit, care should be taken to place the PCD2.H31x modules in the base unit and only to plug “normal” I/O modules into the expansion unit, otherwise the potential drop on the connecting cable might take on excessively high values.

## 2.3 Function-specific data

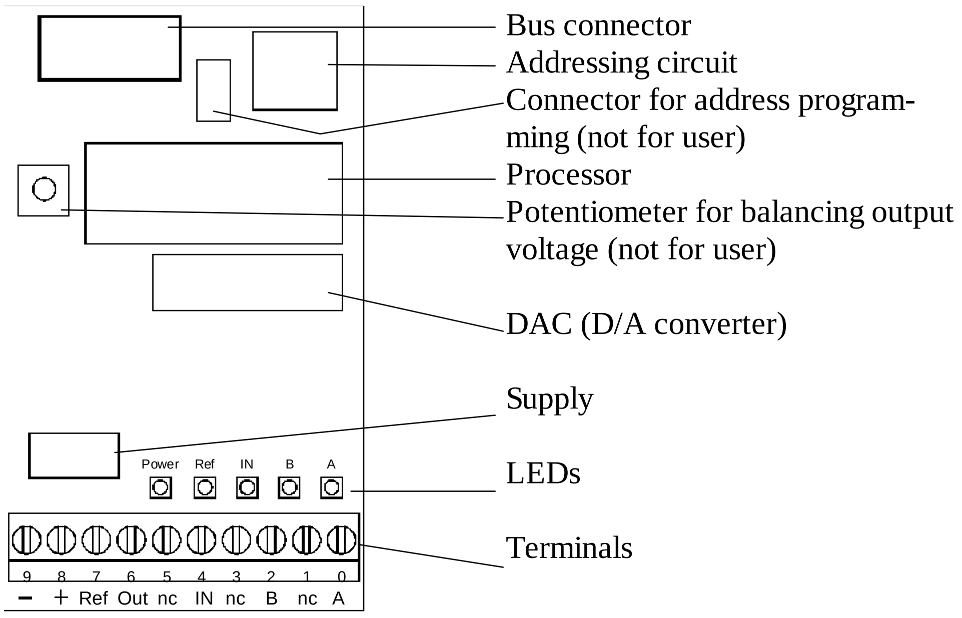
| | |
|---|---|
| Number of systems | 1 |

**Motion parameters**
(31-bit registers are used for destination position, velocity and acceleration, numerical range ± $2^{30}$)

| | |
|---|---|
| Position | Resolution selectable (depending on machine factor) |
| Velocity | Resolution selectable (depending on machine factor) |
| Acceleration | Resolution selectable (depending on machine factor) |
| PID controller | Sample time 341 µs, programmable proportional, integral and differential factors. Sample time for differential part can be programmed separately |
| Counting frequency | up to 50 kHz |

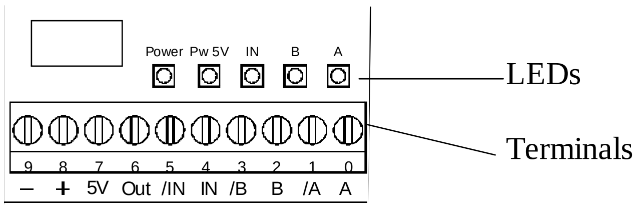
# 3. Presentation

**Assembled module** with terminals PCD2.H310 (24V encoder)


Bus connector
Addressing circuit
Connector for address programming (not for user)
Processor
Potentiometer for balancing output voltage (not for user)
DAC (D/A converter)
Supply
LEDs
Terminals
Power Ref IN B A
9 8 7 6 5 4 3 2 1 0
– + Ref Out nc IN nc B nc A

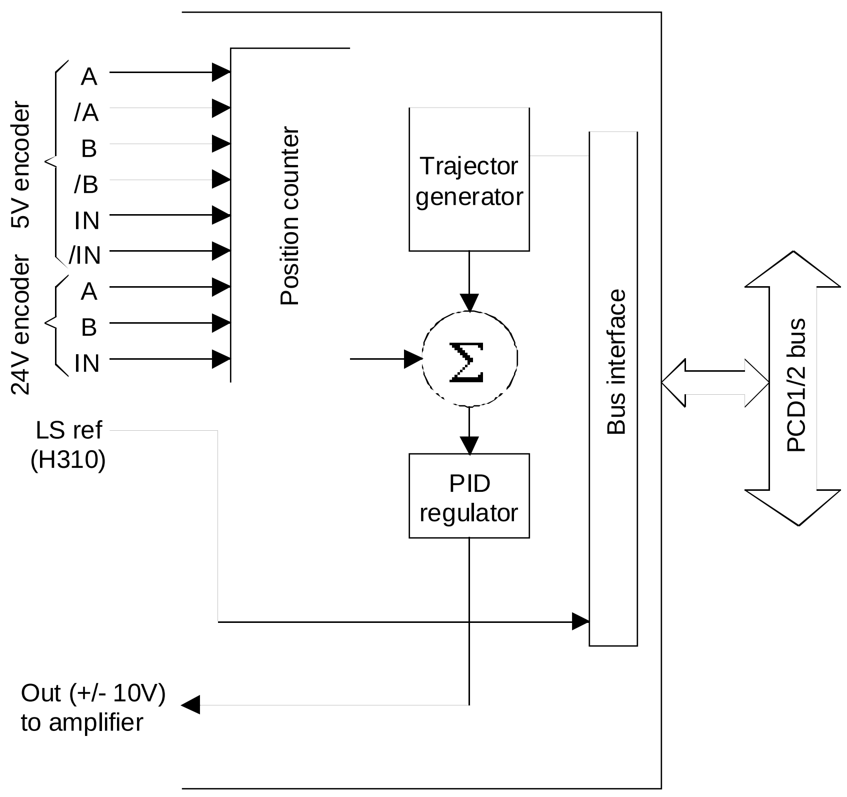
**–** and **+** are the external supply terminals: $V_{ext}$
**Ref** is the digital input for the reference switch
**Out** is the analogue controller output
**A**, **B**, **IN** are the 3 encoder signals
**nc** terminals are not used (not connected)

**Assembled module** with terminals PCD2.H311 (5V encoder)



– and **+** are the external supply terminals: $V_{ext}$
**5V** is the output for the encoder's 5V supply (300 mA max.)
**Out** is the analogue controller output
**A**, **B**, **IN** are the encoder's 3 non-inverted signals
/**A**, /**B**, /**IN** are the encoder's 3 inverted signals

## Simple block diagram

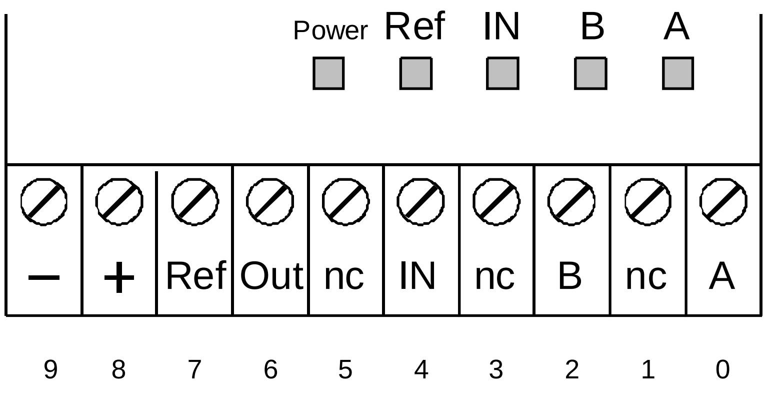


Other axis-specific inputs and outputs (such as limit switches, the PCD2.H311 module’s reference switch, the driver’s ENABLE) must be monitored or controlled with a standard I/O module (e.g. PCD2.B100, PCD2.E110/E111 or PCD2.A410).

# 4. Terminals and meaning of LEDs

**Terminals and LEDs of the PCD2.H310 (24V encoder)**

The picture shows the labelling of the printed circuit board. The I/O connector block has standard 0 .. 9 numbering (right to left)



**Inputs:**

| | | |
|---|---|---|
| Number | | 4 |
| Terminal 0 = | "A": | Encoder signal "A" |
| Terminal 1 = | nc: | not used *) |
| Terminal 2 = | "B": | Encoder signal "B" |
| Terminal 3 = | nc: | not used *) |
| Terminal 4 = | "IN": | Encoder signal "IN" |
| Terminal 5 = | nc: | not used *) |
| Terminal 7 = | "Ref": | Digital input for reference switch |

**Ausgang:**

| | | |
|---|---|---|
| Terminal 6 = | "Out": | Analogue controller output |

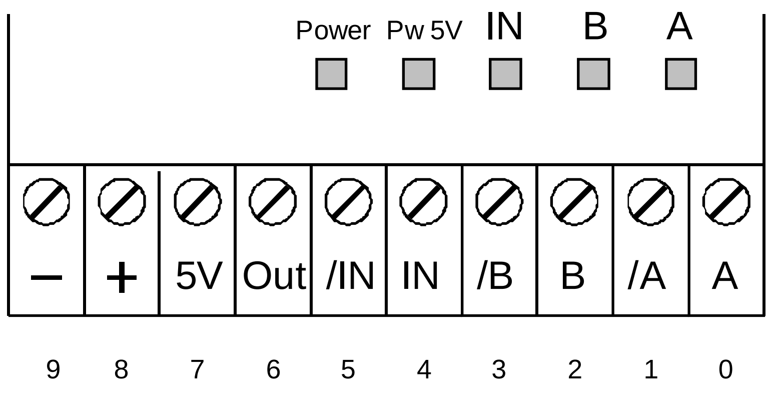
**Speisung:**

| | | |
|---|---|---|
| Terminal 8 = | + | + 24 VDC, smoothed |
| Terminal 9 = | - | GND |

*) Do not wire – not for use as a restart point

## Terminals and LEDs of PCD2.H311 (5V encoder)

The picture shows the labelling of the printed circuit board. The I/O connector block has standard 0 .. 9 numbering (right to left)



**Inputs:**

| | | |
|---|---|---|
| Number | | 6 |
| Terminal 0 = | "A": | Encoder signal "A" |
| Terminal 1 = | "/A": | Encoder signal "/A" |
| Terminal 2 = | "B": | Encoder signal "B" |
| Terminal 3 = | "/B": | Encoder signal "/B" |
| Terminal 4 = | "IN": | Encoder signal "IN" |
| Terminal 5 = | "/IN": | Encoder signal "/IN" |

**Outputs:**

| | | |
|---|---|---|
| Terminal 6 = | "Out": | Analogue controller output |
| Terminal 7 = | "5V": | 5V supply for encoder (max. 300 mA) |

**Supply:**

| | | |
|---|---|---|
| Terminal 8 = | + | + 24 VDC, smoothed |
| Terminal 9 = | - | GND |

## Meaning of LEDs

LED **Ref** shows the logic state of the reference switch.
This LED is "on" when a reference is requested or when the switch has not been wired. (Applies to PCD2.H310 only).

LED **Pw5V** represents the supply of a connected encoder.
This LED is "on" when the 5V are OK (no short-circuit). (Applies to PCD2.H311 only).

LED **Power** shows the presence of ±15V .
This LED is "on" when both 15V supplies are OK.

LEDs **A, B** represent the encoder inputs.
These LEDs are "on" when the logic state is "H".

LED **IN** represents the encoder's index input.
This LED is "on" in the active state (negative logic with H310).

**Caution:** This module includes components that are sensitive to electrostatic discharges.

Notes

# 5. Function description

## 5.1 Operating modes

Two fundamentally different operating modes are available :

- **position control operation**
- **speed control operation**

**Position control operation** (Initiation with instruction: 'StartMot')

Motion control instructions follow the pattern below:

1. Position and parameter entry for velocity profile
2. Start motion control
3. Await signal for "**Destination position reached**".

In position control mode various parameters are entered (PID factors, acceleration, velocity, etc.) after which a preset destination position is approached in a controlled manner. During the motion, velocity, PID factors and destination position can be changed.

**Speed control operation**
(Initiation with instructions 'GoForw' or 'GoBackw')

Pattern of instructions :

1. Parameter entry for the velocity profile
2. Start motion
3. Stop motion by entering a **stop instruction**

In speed control mode the desired velocity is attained with a defined rate of acceleration. Travel continues at this velocity in a controlled manner until there is a stop instruction. The desired velocity can be changed even during motion.

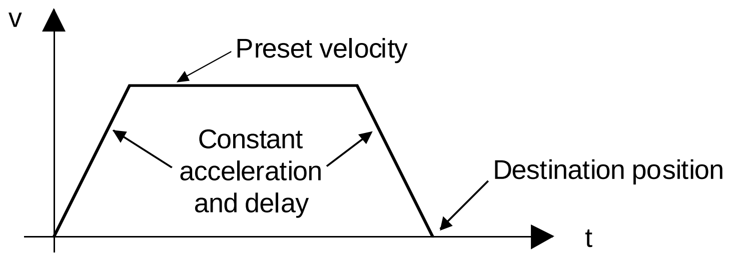
**Function units**

As the block diagram shows (page 3-2) the motion control module essentially consists of the following function units:

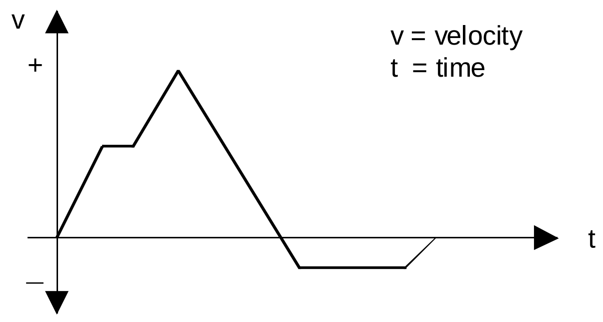
- Generator for velocity profile
- PID controller
- Position decoder and input circuit
- Bus interface (CPLD) to PCD bus
- D/A converter for the analogue controller output

## 5.2 Generator for the velocity profile

The profile generator calculates theoretical speed according to the acceleration and velocity presets and as a function of time in position or speed control mode. In position control operation, the difference between desired and actual positions is fed to the PID controller during motion. Very accurate positioning of the motion is thereby achieved.



Standard velocity profile



Velocity profile with desired velocity and destination position changed during motion.

Velocity and destination position can be changed at any point along the motion and the controller will accelerate or brake accordingly at the defined rate of acceleration. Acceleration and braking ramps are symmetrical.

In speed control mode, the controller accelerates to the user-defined desired speed and runs on at a constant velocity until there is a stop command.

Operating principle of speed control:

The destination position is continuously augmented (according to the required velocity). The difference between destination and actual position (captured with the encoder) is again conveyed to the PID controller. The latter immediately compensates for speed fluctuations (caused by whatever disturbance) by making the controller output greater or smaller.

If the motor fails to reach the desired speed (e.g. due to a blocked rotor) the difference between the destination and actual positions is very large. This generates a position error message, which can trigger an interrupt or automatic motor stop. The maximum allowable position error is a programmable value.

## 5.3 PID controller

The PID controller can be used to help the motor to approach the destination position accurately and keep it in that position, as the controller remains active until a stop command occurs.
The controller utilizes the following algorithm:

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

| | | |
|---|---|---|
| where: | U(n) | $\rightarrow$ Controller output for the motor |
| | e(n) | $\rightarrow$ Position error at the n'th sampling |
| | n | $\rightarrow$ Sampling for the integral portion |
| | n' | $\rightarrow$ Sampling for the differential portion |
| | kp | $\rightarrow$ Proportional factor |
| | ki | $\rightarrow$ Integral factor |
| | kd | $\rightarrow$ Differential factor |

User programmable parameters:

- Control factors kp, ki, kd
- Differential sampling time
- Integration limit (IL) for the integral portion

**Control factors kp, ki and kd** can be changed during motion.

The **sampling time for the proportional and for the integral factor** is 341 µs. This means, that the value of the controller output is refreshed any 341 µs.

The **sampling time of the differential portion** can be adjusted in steps of 341 µs (max. 256*341 µs). For operation at slow speeds, a greater sampling time should be selected.

**Integration limit IL** : limitation applies to the amount of the expression

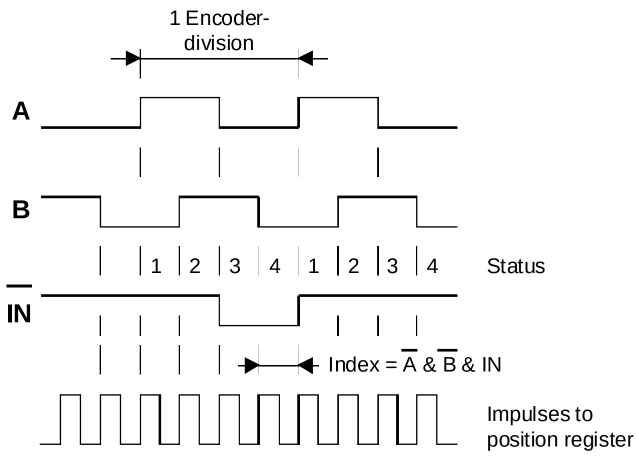
$$ki * \sum_{N=0}^{n} e(n)$$

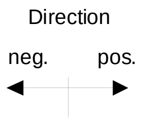
## 5.4 Position decoder and input circuit

**Position and velocity capture**
The precise position and/or speed of the motor is captured with an incremental shaft encoder. The following encoder signals can be connected:

| | |
|---|---|
| PCD2.H310: | A, B, IN (terminals)<br>24V signals in source mode |
| PCD2.H311: | A, /A; B, /B; IN, /IN (terminals<br>5V RS422 inputs (antivalent line driver) |

**Inputs A, B, /IN :**
Status diagram of signals A, B, /IN at position decoder :





| 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 |

**Inputs A, B:**

Whenever signals A or B change status (0 → 1 and 1 → 0) the internal position register is incremented or decremented by 1. In this way the quadruple resolution of encoder division is achieved. Correspondingly, the entry for destination position must also be multiplied by four when working with encoder impulses.

For the position decoder, signals must demonstrate the exact sequence illustrated in the figure above.

**Input IN:**

With modules for 24V encoders (type H310) input IN  can be used as an index impulse input (zero signal from the encoder) or reference point input.

- Use as index impulse input:

  Whenever all 3 encoder signals have a status of zero and the "SetIdx-Pos" (Set Index Position) function block has been called, the absolute motion position is written to the index position register .

- Use as reference point input:

  A reference switch can be connected, e.g. to define the zero position.

**Modules for the connection of 5V encoder signals**

The PCD2.H311 module is used. Shielded cable must be used for connection to the 5V encoder:

- max. cable length: 20m
- min. line cross section: 0.25 mm$^2$

e.g. cable types PCD2.K271 or PCD2.K273

**Reference switch**

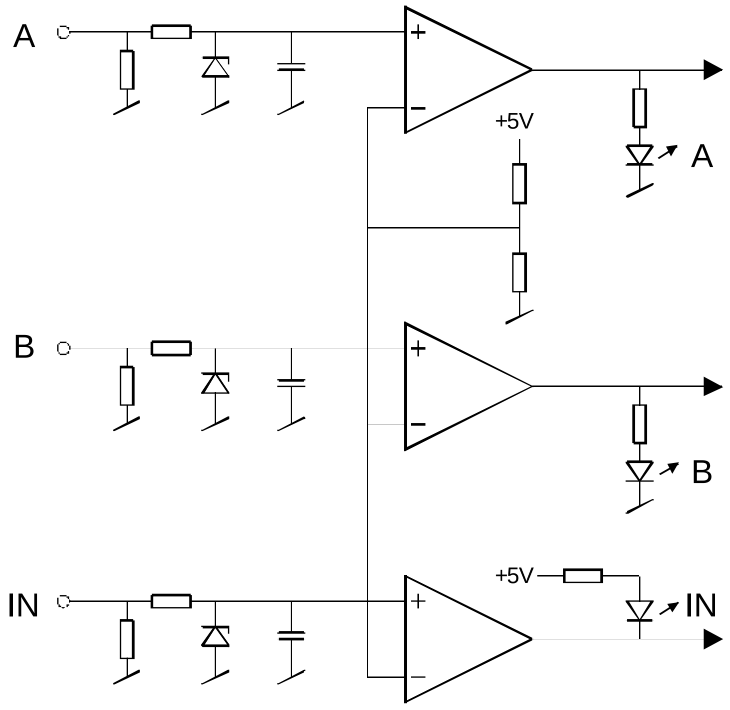
Only the PCD2.H310 has the 'Ref' input (24 VDC, source mode). The signal for this switch is carried directly to the PCD2 bus. This means that the input should be monitored from the user program to trigger any necessary action.

With the PCD2.H311, any reference switch should be wired to a "normal" digital I/O module.

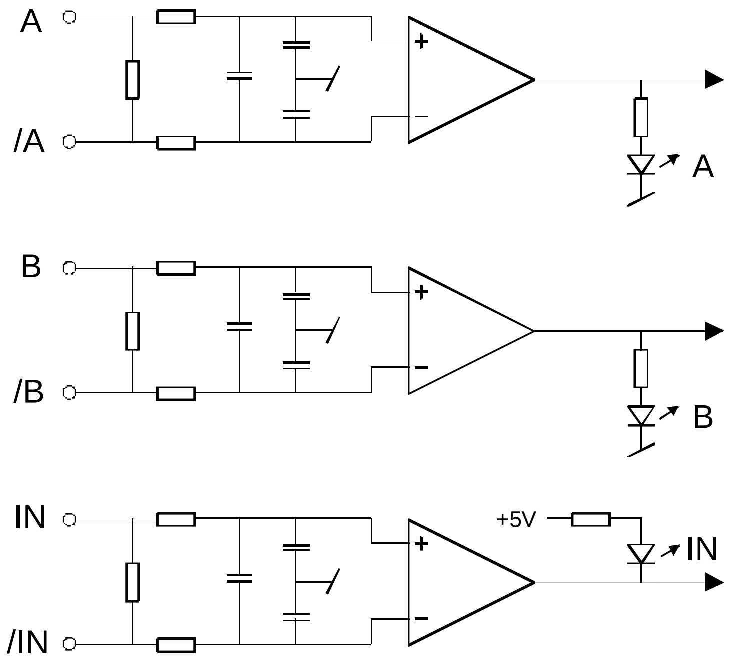
The 'reference switch' is used in combination with the 'limit switches' for the 'Home' FB. In both modules (H310 and H311) these limit switches should be carried to a digital I/O module.

## Input wiring diagrams and connections

### PCD2.H310



### PCD2.H311

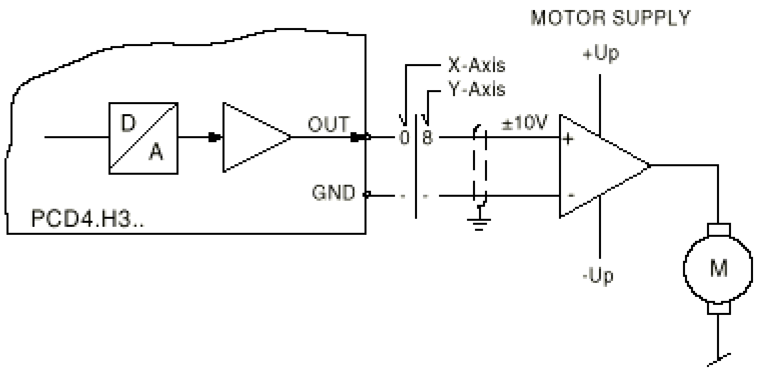


## 5.5 D/A converter

Both PCD2.H310 and H311 modules have an analogue output for the motor’s controller output.

A 12 bit D/A converter has been used.

**Analogue output connection:**



## 5.6 Complementary information: homing (FB Home)

With the 'Home' FB, independent homing can take place. Seven parameters are required for definition of the home travel. The use of the 'Home' FB is described in section 12.1.

The axis to be referenced must have been initialized (FB Init). (The following description refers to the figure on the next page). For successful use of the 'Home' FB, the axis and reference switch must be located between 'LS1' and 'LS2'.

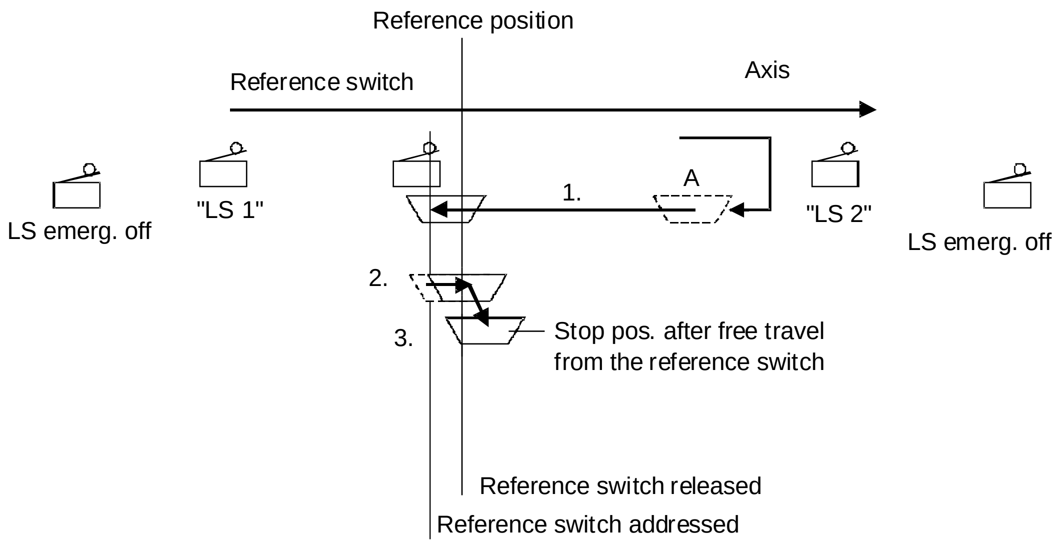
1. The search for the reference switch is undertaken at the velocities defined in parameter 5 (Vmax). The search direction is defined in parameter 2. If the reference switch is not found and the axis meets a limit switch, the search direction is inverted.

2. The digital input to which the reference switch is wired is defined in parameter 7. For a PCD2.H310 module this address is the module base address plus 11. For a PCD2.H311 module any address can be chosen ("normal" digital input module).

3. When the reference switch has been found free travel commences. The direction of free travel is defined in parameter 3; the velocity for free travel is Vmin (parameter 4).

4. When the reference switch has been released, the axis travels as far as the next encoder index signal and stops. This position is now defined as the zero position.

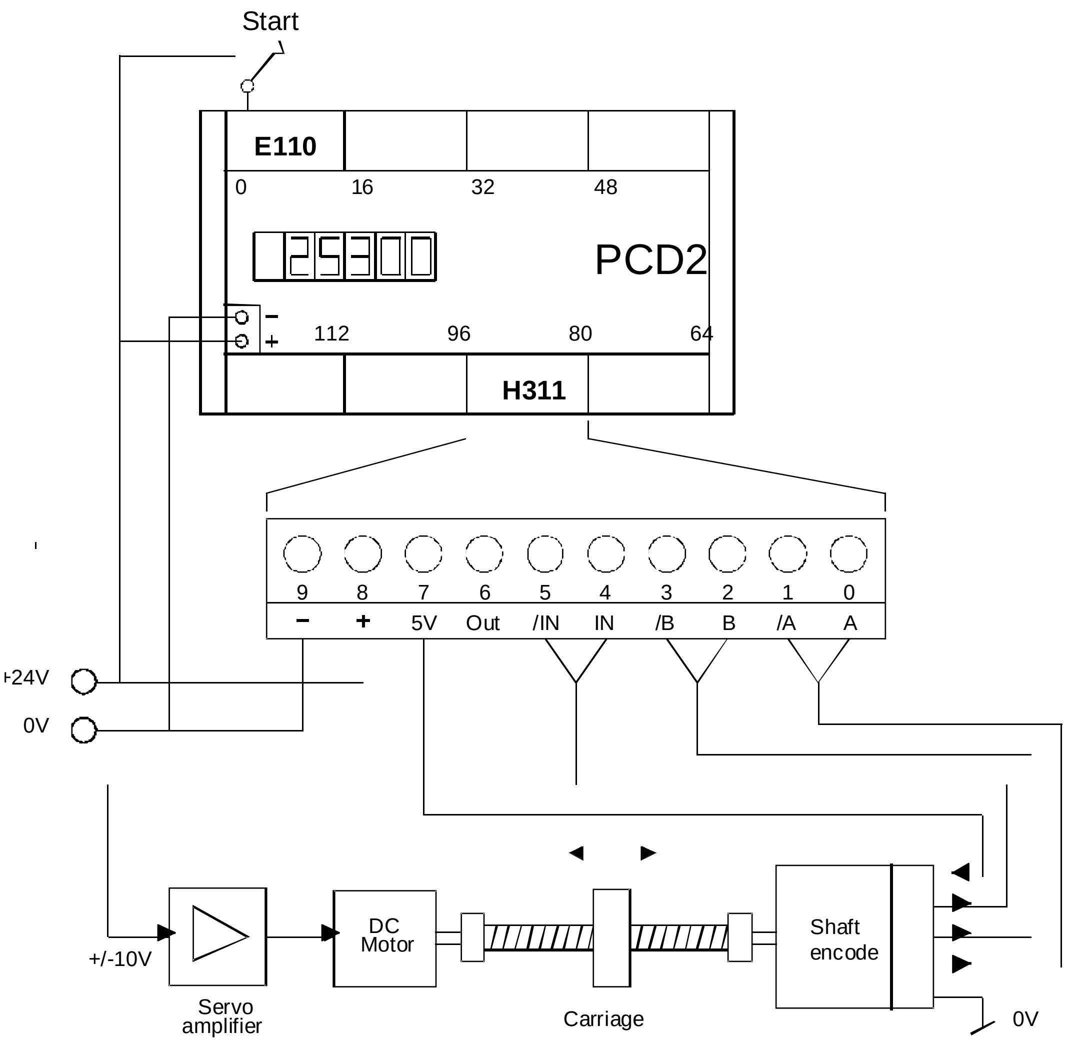
5. The module is configured with the original settings (from FB Init) and FB Home is exited.


Reference position
Reference switch
Axis
A
1.
"LS 1"
"LS 2"
LS emerg. off
LS emerg. off
2.
3.
Stop pos. after free travel from the reference switch
Reference switch released
Reference switch addressed

Instructions:

- Several modules can be referenced on one PCD at the same time.
- The 'fEndHome_x' flag must be =L for the start of the homing procedure.
- Flags 'fLS1_x' and 'fLS2_x' are representations of the limit switches. The logic states of both flags are the result of querying both digital inputs to which these LSs are connected.
- Break contacts should be used for the switches.
- During the homing procedure, limit switches are switched off internally and are only used for reversing the direction of travel.
- If no reference switch is found, the error flag 'fHomeErr_x' (x = module no.) is set and the 'Home' FB is exited.
- When the 'Home' FB has finished (successfully or broken off by an error) the 'fEndHome_x' flag is set automatically.
- Since the 'Home' FB is only exited when homing has been successfully completed or an error has been detected, the FB can be broken off with a timeout (parameter 6). Its value corresponds to the time in seconds after which the 'Home' FB will be abandoned. In this case, as well as the error flag 'fHomeErr_x', the diagnostic register 'rDiag' is loaded with code 6 (for parameter 6) in the third byte (for FB Home). Parameter checking takes place as described in chapter 8.

# 6. Quick start



Minimal arrangement for position control using a PCD2.H310 or PCD2.H311, not including reference switch and limit switches.

The individual elements are:

- PCD1 or PCD2, equipped at least with 1 PCD2.H310/311 with F510/530 display
  1 PCD2.E110
- Motion model with DC motor, spindle and incremental shaft encoder
- DC Motor with gears: approx. 500 rev/min at 10 VDC
  Spindle gradient: 1 mm/revolution
  Incremental shaft encoder: 1000 signals per revolution
- 4-quadrant servo amplifier
  e.g. with Op-Amp LM 12 (National Semiconductor)
  for details see: http://www.national.com/pf/LM/LM12.html
- PG4 from version V2.0.70 and FBs PCD9.H31E

## 6.1 Entry with programming in IL

The basic program shown below is proposed as the simplest way of commissioning a regulated position controller:

A properly written user program should not contain any wait loops. Despite this, our first example has been designed with wait loops to demonstrate the main instructions for driving a PCD2.H31x. In practice a GRAFTEC or (in future) a FUPLA structure should always be chosen for this type of program. (See second example and chapter 10).

**Task:** After switching on the PCD's 'Start' input, the carriage is to travel first in one direction and then, after a pause, back again to the starting position. It is assumed that, at the start, the carriage will be located roughly in the middle of the axis.

The purpose of this example is to demonstrate the principle and possible user program structures. Above all note the beginning of the program with its assembler directives '$include ' and '$group'.

The meaning and definition of individual parameters require some knowledge of control technique and are explained in chapter 9: 'Installation and commissioning'.

This first example (with wait loops) is entitled 'Intro-1.src'. The same example shown subsequently in GRAFTEC is called 'Intro-2.sfc'

The FBs (IL for PG4 from version V2.0.70) are on the PCD9.H31E diskette. To install the FBs on the PC, follow the instructions in chapter 7 below and the README.TXT file. This file can also be found on the diskette.

The number of modules (1) and the address of the PCD2.H311 module (address 80) must be entered in the file D2H310_B.MBA:

```
NbrModules   EQU   1     ; No. of H31x modules used (0...16)

BA_1         EQU   80    ; Base address of module 1
```

This file (D2H310_B.MBA) must be located in the project directory for the example, i.e. the file should be copied manually from the diskette into the current project directory.

### 6.1.1 Entry-level example in IL with wait loop: INTRO-1.SRC

```
$include D2H310_B.equ
$group H310

        xob    16

        LD   R 1000
               4.0         ; Mechanical factor
        LD   R 1001
               8000        ; Initial absolute speed
        LD   R 1002
               10000       ; Initial absolute acceleration

        CFB    Init        ; Initialization FB
             K 1           ;  Module number
               250         ;  Proportional factor (regulator)
               0           ;  Integrative factor (regulator)
               0           ;  Derivative factor (regulator)
               4000        ;  Integrative limit value
               5           ;  Derivative term sampling interval
               500         ;  Position tolerance
               0           ;  Behaviour in case of position error
             R 1000        ;  Mechanical factor register
             R 1001        ;  Initial velocity register
             R 1002        ;  Initial acceleration register

        exob
; ---------------------------------------------------------------

        cob    0
               0

start:  sth  i 0

        CFB    Exec        ; Executable FB
             K 1           ;  Module number
               RdActPos    ;  Command: Read Actual Position
             R 90          ;  Actual Position register
        DSP  R 90          ; Display register

        jr   l start       ; If Start is not done, wait

        LD   R 100         ; Target Position
               20000       ;  Absolute value

        CFB    Exec        ; Executable FB
             K 1           ;  Module number
               LdDestAbs   ;  Command: Load Absolute Destination
             R 100         ;  Absolute Destination register

        CFB    Exec        ; Executable FB
             K 1           ;  Module number
               StartMot    ;  Command: Start motion
               rNotUsed    ;  Dummy register
```

```
pos1:   CFB     Exec       ; Executable FB
             K 1           ;  Module number
               RdActPos    ;  Command: Read Actual Position
             R 90          ;  Actual Position register
        DSP  R 90          ; Display register

        CFB     Exec       ; Executable FB
             K 1           ;  Module number
               RdStatRg    ;  Command: Read Status Register
             R 0           ;  Value of Status Register

        STH     fOnDest_1  ; Position reached?
        jr   l pos1        ; if no - wait (loop)

        ld   t 0           ; load timer for pause
               50          ;  5 sec
pause1: sth  t 0           ; pause elapsed?
        jr   h pause1      ; if no - wait (loop)

        LD   R 100         ; Target Position
               0           ;  Absolute value

        CFB     Exec       ; Executable FB
             K 1           ;  Module number
               LdDestAbs   ;  Command: Load Absolute Destination
             R 100         ;  Absolute Destination register

        CFB     Exec       ; Executable FB
             K 1           ;  Module number
               StartMot    ;  Command: Start motion
               rNotUsed    ;  Dummy register

pos2:   CFB     Exec       ; Executable FB
             K 1           ;  Module number
               RdActPos    ;  Command: Read Actual Position
             R 90          ;  Actual Position register
        DSP  R 90          ; Display register

        CFB     Exec       ; Executable FB
             K 1           ;  Module number
               RdStatRg    ;  Command: Read Status Register
             R 0           ;  Value of Status Register

        STH     fOnDest_1  ; Position reached?
        jr   l pos2        ; if no - wait (loop)

        ld   t 0           ; load timer for pause
               50          ;  5 sec
pause2: sth  t 0           ; pause elapsed?
        jr   h pause2      ; if no - wait (loop)

        ecob

$endgroup
```

Description of programs:

The directive '$include' is used to integrate file 'D2H310_B.equ'. (This file in turn integrates the 'D2H310_B.mba' file with information about the number of H31x modules and their base addresses. This happens automatically. The user has nothing to do with it.)

The directive '$group H310' declares the program code up to '$endgroup' as belonging to PCD2.H31x.

In XOB 16 (coldstart block) module initialization takes place. Before FB INIT the 3 registers for machine factor, velocity and acceleration should be loaded. The choice of individual values is explained in chapter 9: 'Installation and commissioning'.

FB INIT is then called. Choosing the 11 parameters is also described in chapter 9.

The actual motion program occurs in COB 0.

Start is awaited with PCD input 0. To enable current position to be captured even during this phase and appear on the display, the instruction 'RdActPos' is incorporated in the wait loop. This means that, if the start condition is not met, the position will be read continuously.

The absolute destination position is loaded into PCD register R 100. This value is transferred to the module with 'LdDestAbs'. The 'StartMot' command starts the motion. (20 mm traverse)

In a program loop the position is continuously read with 'RdActPos' and output to the display via PCD register R 90. Motion takes place according to the parameters selected in FB INIT, i.e. the motion is travelled in the best way, controlled by the module itself, uninfluenced by the user program, up to the destination point. To allow the program sequence to continue correctly, it is necessary to determine when motion has concluded. This is done by querying the 'fOnDest_x'  flag (fOnDest_1 for module no. 1 in our case). However, before this flag can be queried, it must be activated with the 'RdStatRg' instruction.

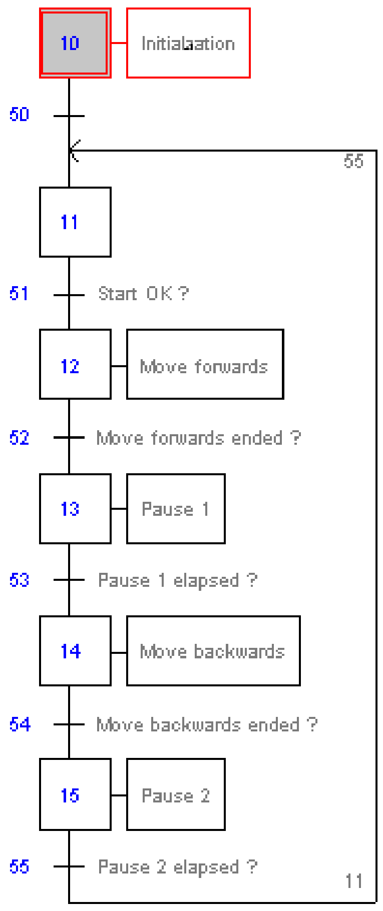
The program loop that reads or displays current position and activates or queries the position flag is in a constant cycle until the destination position has been reached.

When the destination position has been reached a pause of, e.g. 5 seconds is loaded and waited through. The new destination position (zero) is then loaded and motion travels back to the starting point.

In order to know the real, current position even during pauses (stabilization of final position), it would also be necessary to incorporate the commands 'RdActPos' and 'DSP R 90' into wait loops for these pauses.

### 6.1.2 Entry-level example in GRAFTEC: INTRO-2.SFC

Same example as under section 6.1.1, but in a proper GRAFTEC structure without any program jumps or wait loops. Individual steps and transitions have been edited in IL.



**Program code for "intro-2.sfc"**

(To obtain this representation, file "intro-2.sfc should be renamed as “intro-2.src").

```
  SB     0
;------------------------------
  IST    10         ;Initialization
         O 50

$include D2H310_B.equ
$group H310

  LD   R 1000
         4.00       ; Mechanical factor
  LD   R 1001
         8000       ; Initial absolute speed
  LD   R 1002
         10000      ; Initial absolute acceleration

  CFB    Init       ; Initialization FB
       K 1          ;  Module number
         250        ;  Proportional factor (regulator)
         0          ;  Integrative factor (regulator)
         0          ;  Derivative factor (regulator)
         4000       ;  Integrative limit value
         5          ;  Derivative term sampling interval
         500        ;  Position tolerance
         0          ;  Behaviour in case of position error
       R 1000       ;  Mechanical factor register
       R 1001       ;  Initial velocity register
       R 1002       ;  Initial acceleration register
  EST    ;10

;------------------------------
  ST     11
       I 50
       I 55         ;Pause 2 elapsed ?
       O 51         ;Start OK ?
  EST    ;11

;------------------------------
  ST     12         ;Move forwards
       I 51         ;Start OK ?
       O 52         ;Move forwards ended ?

  LD   R 100        ; Target Position
         20000      ;  Absolute value

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         LdDestAbs  ;  Command: Load Absolute Destination
       R 100        ;  Absolute Destination register

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         StartMot   ;  Command: Start motion
         rNotUsed   ;  Dummy register
   EST   ;12
```

```
;--------------------------------
  ST      13          ;Pause 1
       I 52           ;Move forwards ended ?
       O 53           ;Pause 1 elapsed ?

  ld   t 0
         50
  EST    ;13

;--------------------------------
  ST      14          ;Move backwards
       I 53           ;Pause 1 elapsed ?
       O 54           ;Move backwards ended ?

  LD   R 100          ; Target Position
         0            ;  Absolute value

  CFB    Exec         ; Executable FB
       K 1            ;  Module number
         LdDestAbs    ;  Command: Load Absolute Destination
       R 100          ;  Absolute Destination register

  CFB    Exec         ; Executable FB
       K 1            ;  Module number
         StartMot     ;  Command: Start motion
         rNotUsed     ;  Dummy register
  EST    ;14

;--------------------------------
  ST      15          ;Pause 2
       I 54           ;Move backwards ended ?
       O 55           ;Pause 2 elapsed ?

  ld   t 0
         50
  EST    ;15

;--------------------------------
  TR      50
       I 10           ;Initialization
       O 11
  ETR    ;50

;--------------------------------
  TR      51          ;Start OK ?
       I 11
       O 12           ;Move forwards

  CFB    Exec         ; Executable FB
       K 1            ;  Module number
         RdActPos     ;  Command: Read Actual Position
       R 90           ;  Actual Position register
  DSP    R 90         ; Display register

  sth  i 0

  ETR    ;51
```

```
;--------------------------------
  TR     52         ;Move forwards ended ?
       I 12         ;Move forwards
       O 13         ;Pause 1

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         RdActPos   ;  Command: Read Actual Position
       R 90         ;  Actual Position register
  DSP  R 90         ; Display register

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         RdStatRg   ;  Command: Read Status Register
       R 0          ;  Value of Status Register

  STH    fOnDest_1  ; Position reached?
  ETR    ;52

;--------------------------------
  TR     53         ;Pause 1 elapsed ?
       I 13         ;Pause 1
       O 14         ;Move backwards

  stl  t 0
  ETR    ;53

;--------------------------------
  TR     54         ;Move backwards ended ?
       I 14         ;Move backwards
       O 15         ;Pause 2

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         RdActPos   ;  Command: Read Actual Position
       R 90         ;  Actual Position register
  DSP  R 90         ; Display register

  CFB    Exec       ; Executable FB
       K 1          ;  Module number
         RdStatRg   ;  Command: Read Status Register
       R 0          ;  Value of Status Register

  STH    fOnDest_1  ; Position reached?
  ETR    ;54

;--------------------------------
  TR     55         ;Pause 2 elapsed ?
       I 15         ;Pause 2
       O 11

  stl  t 0

$endgroup
  ETR    ;55

  ESB    ;0
```

**Explanatory notes to the program**

Knowledge of the PG4 in general and GRAFTEC in particular is assumed.

During assembly, sequential block SB 0 is called automatically from a COB.

The course of the GRAFTEC program can be viewed online.

Initialization of the H310 module takes place in IST 10. In the arrangement chosen, this IST is only processed when the SB is called for the first time, like XOB 16. It makes sense if initialization of the H310 module takes place in the IST of whichever SB deals with the module, so that the whole program routine stays together. XOB 16 is preferred for carrying out initializations that apply to the whole PCD.

In ST 12 and ST 14 the absolute destination position is loaded via PCD register R 100 into the H310 module and motion is started.

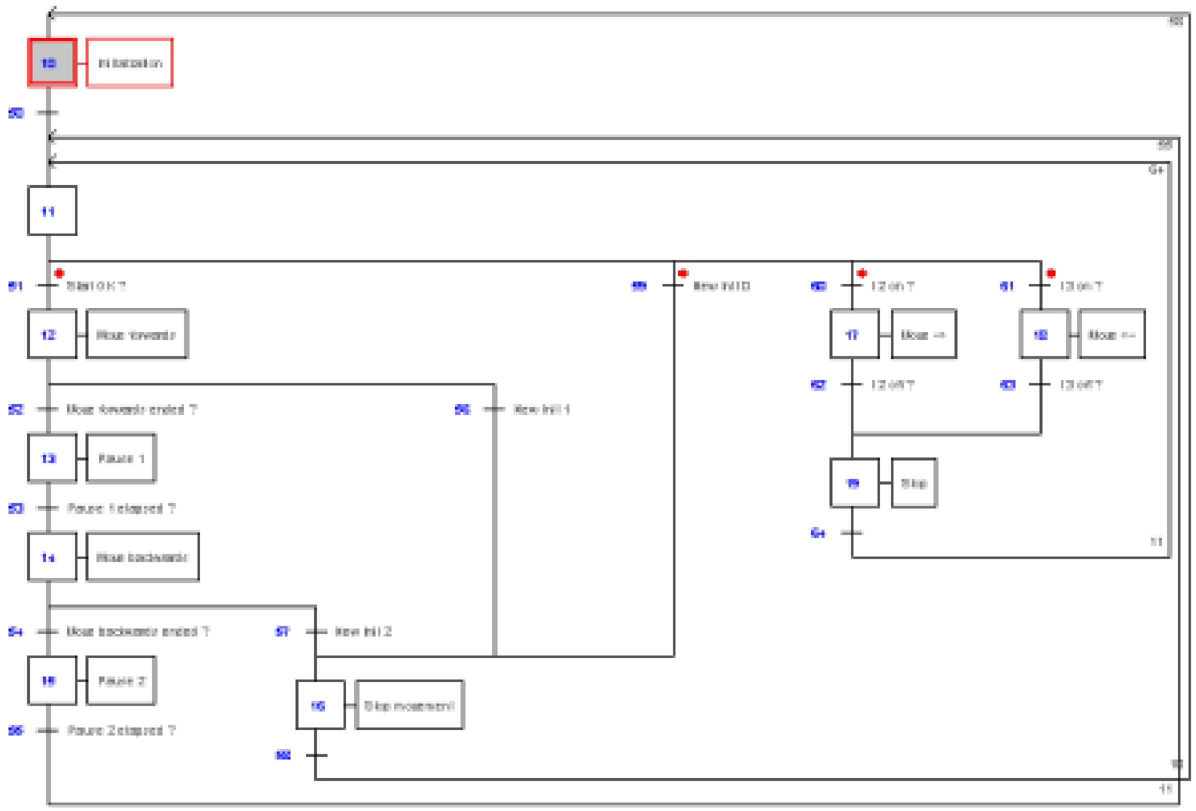
In TR 52 and TR 54 the end of a motion is queried from software so that the program can be released to continue its course. Motion itself is controlled directly by the module. Before querying the continuation condition (sth fOnDest_1) the current position is read and output to the display, and the 'fOnDest_1' flag is refreshed with 'RdStatRg'. In accordance with GRAFTEC rules, for every unfulfilled TR (e.g. destination position not yet reached) the program returns to the calling COB and continues working. At the next program cycle the unfulfilled TR is processed again **<u>in full</u>**. This ensures that the position is read and displayed automatically each time and that the 'fOnDest_1' flag is refreshed.

To be correct, the position in an experimental set-up should also be read and displayed during pauses, so that carriage status can be viewed even during stabilization.

### 6.1.3 Simple commissioning program *)

To try out the different parameters for achieving optimal motion, the following program "test-par.sfc" is proposed. (It has been derived from the previous example "intro-2.sfc").

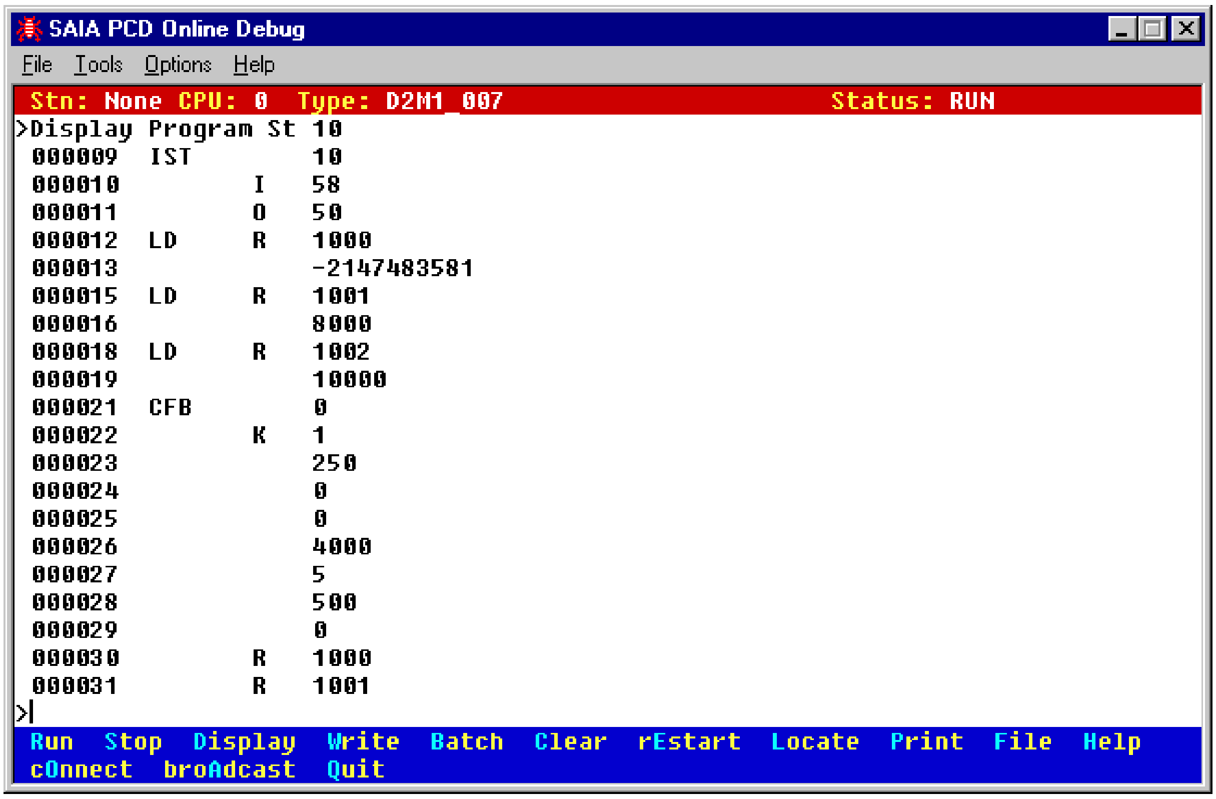
**Function:**



| | |
|---|---|
| I 0: | Start of backwards and forwards motion (as "intro-2.sfc") |
| I 1: | Adoption of new parameters, which were changed online in the debugger. By activating I 1, motion can also be stopped abruptly. |
| I 2: | From the resting position, the carriage can be moved forward with the chosen parameters for as long as I 2 remains switched on. |
| I 3: | From the resting position, the carriage can be moved backward with the chosen parameters for as long as I 3 remains switched on. |

Parameters can be modified online in the debugger. To identify the absolute addresses of parameters, the IST should be displayed or even printed
out with <Display> <Program> <Step> <10> <CR>.

*) A more convenient commissioning program (commissioning tool) in FUPLA is presented in chapter 10.



```
Stn: None CPU: 0  Type: D2M1_007                  Status: RUN
>Display Program St 10
 000009  IST         10
 000010        I     58
 000011        O     50
 000012  LD    R     1000
 000013              -2147483581
 000015  LD    R     1001
 000016              8000
 000018  LD    R     1002
 000019              10000
 000021  CFB         0
 000022        K     1
 000023              250
 000024              0
 000025              0
 000026              4000
 000027              5
 000028              500
 000029              0
 000030        R     1000
 000031        R     1001
>|
```

```
LD    R 1000
        4.0          ; Mechanical factor
LD    R 1001
        8000         ; Initial absolute speed
LD    R 1002
        10000        ; Initial absolute acceleration

CFB     Init         ; Initialization FB
      K 1            ;  Module number
        250          ;  Proportional factor (regulator)
        0            ;  Integrative factor (regulator)
        0            ;  Derivative factor (regulator)
        4000         ;  Integrative limit value
        5            ;  Derivative term sampling interval
        500          ;  Position tolerance
        0            ;  Behaviour in case of position error
      R 1000         ;  Mechanical factor register
      R 1001         ;  Initial velocity register
      R 1002         ;  Initial acceleration register
EST     ;10
```

For example, if initial absolute acceleration is to be increased from 10 000 to 30 000, absolute program line 19 should be processed:

<Write> <Program> <19> <CR> <30000> <CR> <Esc>

When motion is resting: Switch PCD input I 1 on and off. This means that the modified parameters are adopted. By switching on I 0 the new behaviour can be tried out, etc.

## 6.2 Entry with programming in FUPLA

In preparation

Notes

# 7. Programming

The PCD is programmed to use the PCD2.H.. counting and motion control modules via the PCD user program with the standard PG4 programming tool from version V2.0.70. (For applications with the older PG3 programming tool, use FBs for the PCD_4_.H3.. module).

Programming takes place either in IL (Instruction List) with FBs (Function Blocks) or in FUPLA with FBoxen (in preparation). FBs are available on diskette under reference PCD9.H31E.

Since motion control tasks are always sequential processes, it is preferable if user programs are written in GRAFTEC, editing the individual steps and transitions in IL with FBs or, in FUPLA, FBoxes. However, user programs can also be written in straight BLOCTEC or FUPLA.

## 7.1 Programming in IL with FBs

### 7.1.1 The IL package (Installation of FB)

This diskette has order reference PCD9.H31E. It contains the following directories:

- APPSDIR : containing all helps
- FB : containing the .SRC and .EQU files of the H31x
- FBOX : containing FBoxes for the H31x
- PG3_FB : containing all FB files for the PG3
- PG4_FB : containing examples and the .MBA file
- Readme : containing general information

This package is provided for use with SAIA PG4 from version V2.0.70. Consult the 'Readme' file for all other PG4 versions. (The package also contains FBs for use with the earlier PG3, see 'Readme').

FBoxes for FUPLA are not yet available.

**Installation of package for the PG4**

The easiest method of installation is with the PG4 'Setup Extra Files' program:

Insert diskette PCD9.H31E into drive A:
<Start> <Programs> <SAIA PG4> <Setup Extra Files>. FBs and the 'Help' file are installed on the hard disk in directory 'PG4'.

The following files are installed:

| | | |
|---|---|---|
| D2H310_B.SRC | FB source code | read-only file |
| D2H310_B.EQU | FB definitions | read-only file |

These 2 files are copied from the diskette into PG4 directory ...\PG4\FB.

| | |
|---|---|
| FB_LIB.HLP | FB library data |
| D2H310_B.HLP | FB help file |

This file is located in directory A:\APPSDIR and is copied into the PG4 directory ...\PG4.

The file **D2H310_B.MBA** (module base addresses) must be copied **manually** from the diskette, directory PG4_FB, into the relevant project directory.

File **'D2H310_B.MBA'** is important for the user and is shown below:

File: **D2H310_B.MBA** (MBA = Module Base Address)

```
;
; This file can be modified by the user
;
; Base addresses defined by the user
; ---------------------------------
$group H310
NbrModules      EQU     1       ; No. of H310 modules used (0...16)

;
; Module base addresses (only the used modules must be defined)

BA_1            EQU    32       ;Base address of module 1
BA_2            EQU     0       ;Base address of module 2
BA_3            EQU     0       ;Base address of module 3
BA_4            EQU     0       ;Base address of module 4
BA_5            EQU     0       ;Base address of module 5
BA_6            EQU     0       ;Base address of module 6
BA_7            EQU     0       ;Base address of module 7
BA_8            EQU     0       ;Base address of module 8
BA_9            EQU     0       ;Base address of module 9
BA_10           EQU     0       ;Base address of module 10
BA_11           EQU     0       ;Base address of module 11
BA_12           EQU     0       ;Base address of module 12
BA_13           EQU     0       ;Base address of module 13
BA_14           EQU     0       ;Base address of module 14
BA_15           EQU     0       ;Base address of module 15
BA_16           EQU     0       ;Base address of module 16
$endgroup
```

The number of PCD2.H31x modules should be specified. The hardware base addresses of PCD2.H31x modules utilized should then be entered.

Since the '.mba' file does not appear in the Project Manager, a text editor is required for any adjustments, e.g. SEDIT32.

Modules should be numbered consecutively, starting from 'BA_1'. For example, if 3 x H310 modules are used in a project, use 'BA_1', 'BA_2' and 'BA-3'. Module sockets can be assigned as desired, for example:

```
NbrModules      EQU     3       ; No. of H310 modules used (0...16)
;
; Module base addresses (only the used modules must be defined)

BA_1            EQU    64       ;Base address of module 1
BA_2            EQU   208       ;Base address of module 2
BA_3            EQU   112       ;Base address of module 3
BA_4            EQU     0       ;Base address of module 4
BA_5            EQU     0       ;Base address of module 5
```

The base addresses of registers, flags and FBs are assigned automatically and can be consulted in the resource list under 'View' - 'Resource List'.

Arrangement of files and procedure when writing a user program. The project created is to be entitled "TEST-H3" and the actual user program should have the name "move-01.sfc".

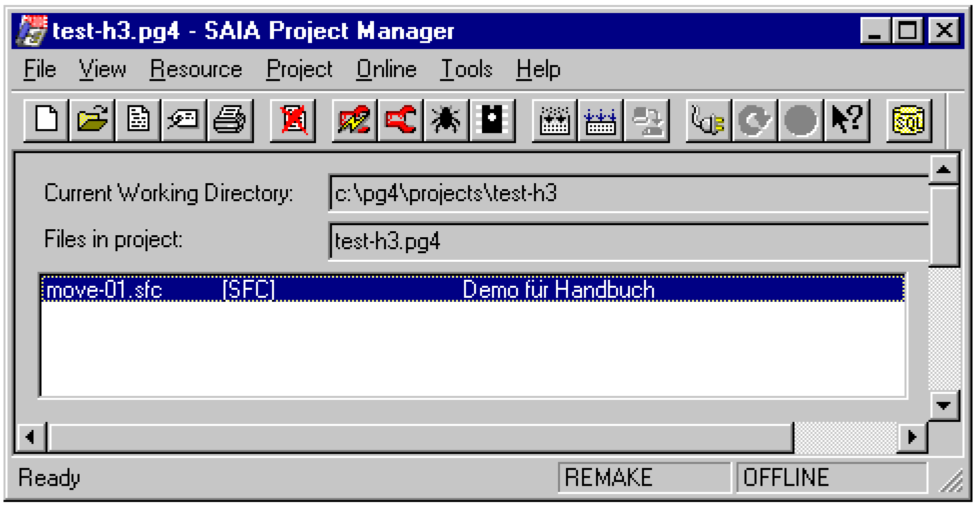
```
C:\PG4 \FB          \D2H310_b.equ
                    \D2H310_b.src
       \...
       \FBOX        \...
       \GALEP3      \...
       \PROJECTS    \FUP_E              (Demo example PG4)
                    \GRAF_E             (Demo example PG4)
                    \TEST-H3    \D2H310_b.mba
                                \move-01.sfc
       \...
       \D2H310_b.hlp
```

The user program for the H310 section has the following presentation:

```
$include D2H310_b.equ
$group H310

XOB     16


PCD-Code


ecob
$endgroup
```

If the program is written in GRAFTEC, the assembler directives "$include" and "$group" will usually be located in the first step (ST), normally the initial step (IST). "$endgroup" comes at the end of the last transition (TR).

If everything has been correctly installed, the user program edited and all parameters defined, it is possible with 'Project' - 'Build' to process the program and load it into the PCD.


test-h3.pg4 - SAIA Project Manager

Current Working Directory: c:\pg4\projects\test-h3

Files in project: test-h3.pg4

move-01.sfc [SFC] Demo für Handbuch

Ready REMAKE OFFLINE

### 7.1.2 Individual FBs

The whole package basically consists of 2 (3) FBs with parameters:

| | | |
|---|---|---|
| • INIT | Initialization | FB with 11 parameters |
| • EXEC | Execution | FB with 3 parameters |
| • HOME | Home position | FB with 7 parameters |

Calling FB "INIT" always has the following presentation:
(values entered serve as examples only)

```
CFB        init    ; Intitialization of a PCD2.H31x module
         k 1       ; Par. 1:  Module number (k 1 - k 16)
           250     ; Par. 2:  Proportional factor
           150     ; Par. 3:  Integral factor
           10      ; Par. 4:  Differential factor
           4000    ; Par. 5:  Integration limit
           5       ; Par. 6:  Sample time of D factor
           500     ; Par. 7:  Position tolerance
           0       ; Par. 8:  Behaviour under position error
         r 1000    ; Par. 9:  Mechanical factor *)
         r 1001    ; Par. 10: Velocity *)
         r 1002    ; Par. 11: Acceleration  *)
```

*) Registers for parameters 9, 10 and 11 should be loaded with the correct values before processing FB INIT.

Calling FB 'EXEC' has the following presentation for some typical examples:

```
CFB        exec
         k 1         ; Par. 1:  Module number   (k 1 – k 16)
           LdDestRel ; Par. 2:  Function (instruction)
         r 777       ; Par. 3:  Value (from source register)

CFB        exec
         k 1         ; Par. 1:  Module number   (k 1 - k 16 )
           start     ; Par. 2:  Function (instruction)
           rNotUsed  ; Par. 3:  not used

CFB        exec
         k 1         ; Par. 1:  Module number   (k 1 - k 16)
           RdActPos  ; Par. 2:  Function (instruction)
         r 1000      ; Par. 3:  Value (in dest. register)
```

Three parameters must always be specified, even if only 2 are required for a function. For the third parameter, specify 'rNotUsed', or any register.

A list with all instructions follows on the next page.

## Instructions (functions) for FB 'Exec' (Parameter 2):

| No. | Symbol | Instruction | Page |
|---|---|---|---|
| 01 | StartMot | Start Motion | A-7 |
| 02 | StopUrg | Stop motion in Urgency | A-8 |
| 03 | Stop | Stop motion | A-9 |
| 04 | MotOff | Motor regulation Off | A-10 |
| 05 | RdActPos | Read Actual Position | A-11 |
| 06 | RdActVel | Read Actual Velocity | A-12 |
| 07 | RdIntSum | Read Integration Sum | A-13 |
| 08 | RdIndexRg | Read Index Register | A-14 |
| 09 | RdStatRg | Read Status Register | A-15 |
| 10 | RdTargPos | Read Target Position | A-16 |
| 11 | RdTargVel | Read Target Velocity | A-17 |
| 12 | GoForw | Go Forwards | A-18 |
| 13 | GoBackw | Go Backwards | A-19 |
| 14 | SgStpFor | Single Step Forwards | A-20 |
| 15 | SgStpBak | Single Step Backwards | A-21 |
| 16 | LdDestAbs | Load Destination Absolute | A-22 |
| 17 | LdDestRel | Load Destination Relative | A-23 |
| 18 | LdVelAbs | Load Velocity Absolute | A-24 |
| 19 | LdVelRel | Load Velocity Relative | A-25 |
| 20 | LdAccAbs | Load Acceleration Absolute | A-26 |
| 21 | LdAccRel | Load Acceleration Relative | A-27 |
| 22 | LdPropG | Load Proportional Gain | A-28 |
| 23 | LdIntG | Load Integrative Gain | A-29 |
| 24 | LdDerG | Load Derivative Gain | A-30 |
| 25 | LdSampInt | Load derivative Sampling Interval | A-31 |
| 26 | LdIntLim | Load Integrative Limit | A-32 |
| 27 | ActRegFact | Activate Regulation Factors | A-33 |
| 28 | LdBrkPtAbs | Load Breakpoint Absolute | A-34 |
| 29 | LdBrkPtRel | Load Breakpoint Relative | A-35 |
| 30 | ResStatRg | Reset Status Register | A-36 |
| 31 | SetIdxPos | Set Index Position | A-37 |
| 32 | SetZero | Set Zero position | A-38 |
| 33 | MotConf | Motion Configuration | A-39 |
| 34 | SetPosTol | Set Position Tolerance | A-40 |

The figure in the first column (0 - 34) is the absolute value of parameter no. 2 in FB 'Exec'. This figure can be used to interpret the function of FB 'Exec' when viewing the user program in the debugger.

Calling FB "HOME" always has the following presentation:
(values entered serve as examples only)

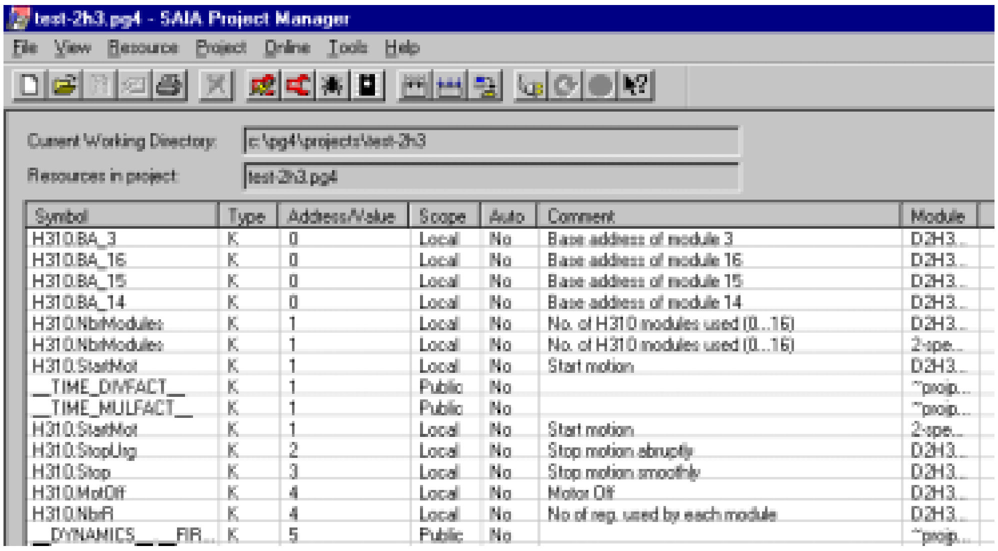
```
CFB         home    ; Intitialization of reference position
          k 1       ; Par. 1:  Module number (k 1 - k 16)
            1       ; Par. 2:  Search direction
            0       ; Par. 3:  Free travel direction
          r 990     ; Par. 4:  Minimum velocity
          r 991     ; Par. 5:  Maximum velocity
            1000    ; Par. 6:  Timeout
          i 7       ; Par. 7:  Reference input
```

Elements that can be queried by the user:

| Element | Description |
|---|---|
| fHomeErr_x | When error with 'Home', element = H.<br>(Timeout, home position not found) |
| fLS1_x | H on arrival at limit switch 1 |
| fLS2_x | H on arrival at limit switch 2 |
| fEndHome_x | Always = H, except during home procedure |
| fBrkPt_x | H, when breakpoint reached |
| fOnDest_x | H, when destination position reached |
| | |
| fPosErr_x | H, in case of major position error |
| | |
| fPar_Err | Parameter error (outside range) |
| fTimeout | Read/write (with hardware problem) |
| | |
| Ref_1 | Representation of reference input |
| | |

'_x'    corresponds to the module number

Effective element addresses should be taken from the resource list: from Project Manager 'View' - 'Resource List'. A complete, detailed list appears.


test-2h3.pg4 - SAIA Project Manager
File View Resource Project Online Tools Help

Current Working Directory: c:\pg4\projects\test-2h3
Resources in project: test-2h3.pg4

| Symbol | Type | Address/Value | Scope | Auto | Comment | Module |
|---|---|---|---|---|---|---|
| H310.BA_3 | K | 0 | Local | No | Base address of module 3 | D2H3... |
| H310.BA_16 | K | 0 | Local | No | Base address of module 16 | D2H3... |
| H310.BA_15 | K | 0 | Local | No | Base address of module 15 | D2H3... |
| H310.BA_14 | K | 0 | Local | No | Base address of module 14 | D2H3... |
| H310.NbrModules | K | 1 | Local | No | No. of H310 modules used (0...16) | D2H3... |
| H310.NbrModules | K | 1 | Local | No | No. of H310 modules used (0...16) | 2-spe... |
| H310.StartMot | K | 1 | Local | No | Start motion | D2H3... |
| __TIME_DIVFACT__ | K | 1 | Public | No | | ~proj... |
| __TIME_MULFACT__ | K | 1 | Public | No | | ~proj... |
| H310.StartMot | K | 1 | Local | No | Start motion | 2-spe... |
| H310.StopUrg | K | 2 | Local | No | Stop motion abruptly | D2H3... |
| H310.Stop | K | 3 | Local | No | Stop motion smoothly | D2H3... |
| H310.MotOff | K | 4 | Local | No | Motor Off | D2H3... |
| H310.NbrR | K | 4 | Local | No | No of reg. used by each module | D2H3... |
| __DYNAMICS___FIR... | K | 5 | Public | No | | ~proj... |

A clear list with absolute addresses, which ought to be adequate for de-bugging, is supplied by the project .map file:

```
SAIA PCD LINKER SP 2.0.83 FILE: test-2h3.pcd
LINKED: 06/30/99 10:57 PAGE 3

FOR SAIA'S INTERNAL USE ONLY

SYMBOL            TYPE VALUE         DEFINED   REFERENCED

H310.Exec         FB   1             D2H310_B  2-speed
H310.fBrkPt_1     F    7517          D2H310_B
H310.fEndHome_1   F    7516          D2H310_B
H310.fHomeErr_1   F    7513          D2H310_B
H310.Flag_base    F    7502..7528    D2H310_B  2-speed
H310.fLS1_1       F    7514          D2H310_B
H310.fLS2_1       F    7515          D2H310_B
H310.fOnDest_1    F    7518          D2H310_B
H310.fPar_Err     F    7502          D2H310_B
H310.fPosErr_1    F    7528          D2H310_B
H310.fTimeout     F    7512          D2H310_B
H310.Home         FB   2             D2H310_B
H310.Init         FB   0             D2H310_B  2-speed
H310.rDiag        R    3500          D2H310_B
H310.Reg_base     R    3500..3524    D2H310_B  2-speed
H310.rIniVel_1    R    3524          D2H310_B
H310.rMecFac_1    R    3522          D2H310_B
H310.rSampInt_1   R    3523          D2H310_B

Linkage complete. 0 errors, 0 warnings.
```

## Assignment of addresses on the bus

Each module occupies 16 addresses as inputs (readable) and 16 addresses as outputs (writeable).

| Bit no. | DATA In (read) | DATA Out (write) |
|---|---|---|
| 0 | Bus data (LSB) | Bus data (LSB) |
| 1 | Bus data | Bus data |
| 2 | Bus data | Bus data |
| 3 | Bus data | Bus data |
| 4 | Bus data | Bus data |
| 5 | Bus data | Bus data |
| 6 | Bus data | Bus data |
| 7 | Bus data (MSB) | Bus data (MSB) |
| 8 | Supply *) | Write (WR LM628) |
| 9 | - | Read (RD LM628) |
| 10 | - | Port select (PS LM628) |
| 11 | Ref. Switch **)/ Pw5V ***) | - |
| 12 | Version ****) | - |
| 13 - 15 | - | - |

| | |
|---|---|
| *) | Monitoring ±15V (H: Supply OK; L: Supply not OK). |
| **) | Ref Switch. For H310 only. (H: RS active; L: RS not active). |
| ***) | Monitoring 5V. H311 only. (H. 5V OK; L: 5V not OK). |
| ****) | Module version (H: H310; L: H311). |

## 7.2 Programming in FUPLA with FBoxes

in preparation

## 7.3 Programming in GRAFTEC with FBoxes

in preparation

# 8. Error handling and diagnosis

## 8.1 Definition error checked by assembler

The following definition errors in file D2H310_b.MBA are checked during assembly:

- If the number of modules (NbrModules) is < 1, no code is assembled and the following warning is written in the 'Make' window:

**"Remark: No H310 used (NbrModules = 0 in D2H310_B.MBA)"**

- If the number of modules (NbrModules) is > 16, no code is assembled and the following error message is written in the 'Make' window:

**"Error : more than 16 Modules H310 defined (NbrModules = 0...16)"**

- If an incorrect instruction code is used for FB 'Exec' (e.g. RdIdenti instead of RdIdent), the assembler reports an error:

**"Symbol not defined 'H310.RdIdenti'"**
(where the expression 'H310' is generated by $group h310)

- If the definition $group H310 is missing, the assembler reports:

**"Symbol not defined"**

for every instruction and every register/flag used in the program.

## 8.2 Error handling in Run

### 8.2.1 Incorrect parameter

In FB 'Exec' the instruction code only is checked. Parameters 1 (module no.) and 3 (source/destination register) are not checked, to avoid making the execution time longer.

In FBs 'Init' and 'Home' the values of all parameters are checked to ensure they are within the premitted range. If a parameter lies outside a range it is brought to the minimum value, the error flag 'fPar_Err' is set and the diagnostic register 'rDiag' is loaded with the relevant error code.

The 'fPar_Err' flag is not reset within FBs. This should take place in XOB 16 or the 'Init' step.

The error code is made up as follows:

```
rDiag   bit 31 . . . . . . . 24 23 . . . . . . . 16 15 . . . . . . .  8 7 . . . . . . . 0
            \  Reserve  /      \  FB no.  /     \  Par. no.  /   \ Mod. no./
                                (Init = FB 1)
                                (Exec = FB 2)
                                (Home = FB 3)
```

Example: If the sample time (parameter 6) in FB 'Init' of module 2 is incorrectly defined (>255), 'rDiag' is loaded with the hex value 00 01 06 02.

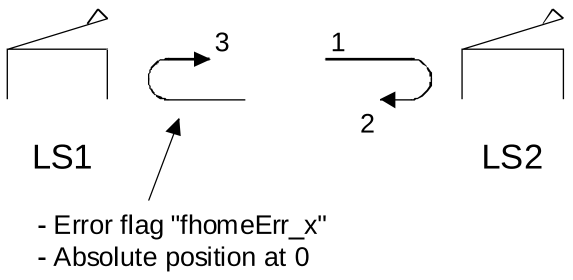
At each incorrect parameter, the diagnostic register is overwritten and always contains the last error. It should therefore be evaluated as soon as the 'fPar_Err' flag signals a range error. The absolute addresses of 'rDiag' and 'fPar_Err' can be seen in the 'project.MAP' file (see section 7.1.2, page 7-8). This can be userful during commissioning with the debugger to locate an error:

- Run until flag 'fPar_Err' = H
- Display register 'rDiag' hex
- Delete flag 'fPar_Err'

### 8.2.2 Error during homing

If it has not been possible to find the reference position (e.g. due to a faulty reference switch), the 'fHomeErr_x' error flag is set, motion is stopped, the absolute position is zeroed and FB 'Home' is broken off.

Reference switch absent or incorrectly wired:



If FB 'Home' has been broken off because the specified timeout has elapsed, the diagnostic register 'rDiag' is additionally loaded with code 6 as the parameter number (the timeout is parameter 6).

The 'fHomeErr_x' flag is defined for each module (_x is the module no.) and is reset at the start of FB 'Home'. This flag should be queried each time FB 'Home' is called, to ensure that the axis is correctly referenced:.

Example:

```
CFB       Home            ; Homing axis 3
        k 3               ;  Module number
          0               ;  search direction
        r 1010            ;  min. speed
        r 1011            ;  max. speed
          50              ;  timeout
        i 64              ;  input reference switch

STH       fHomeErr_3      ; Query home error flag
                          ;  of axis 3
CFB     h Errorhandl      ; Call (user specific)
                          ;  FBs, in case fHomeErr_3 = H

CFB        Exec           ; Motion 1
```

Notes

# 9. Installation and commissioning

## 9.1 Introduction

The following description shows the procedure for commissioning a servodrive with the PCD2.H3.. motion control module. To guarantee fault-free operation of the H3 module, during commissioning the steps described below should be executed in the same sequence.

**Selection criteria for a servo-drive with the PCD2.H3.. module**

A motion control unit always comprises the following parts:

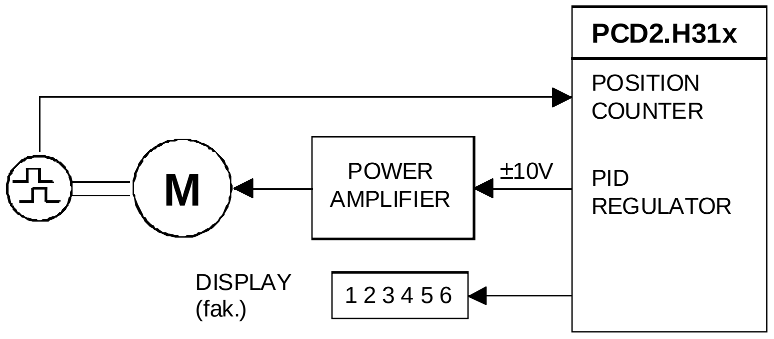
- A motion controller for setting the motion control parameters (position, velocity and acceleration) and for position control. This task is performed by the PCD1/2 with the H3 module.
- A servo-amplifier for triggering the servomotor.
- A servomotor for converting electrical into mechanical energy.
- A position transmitter, in the form of an incremental shaft encoder.
- A mechanical drive unit.

Selection of servo-amplifier and servomotor:

Regardless of whether a DC or AC servodrive is used, special attention should be paid to the following points.

- Amplifier and motor must match each other (power, voltage and current).
- For precise position and speed control, a four-quadrant power output stage is required with integral speed governor.
- The greatest possible governing range for amplifier and motor speed, so that the necessary torque can be applied even at low speeds.
- The H3 module supplies an analogue ±10V signal as the speed set-point.
- For position capture, the H3 module needs an incremental shaft encoder that supplies at least two square-wave signals in phase quadrature.

**Block diagram of a motion control drive with the H3 module**



## 9.2. Installation and wiring

When installing the PCD2 system particular attention should be paid to the points listed below. Before powering up the system, a visual check of the installation and wiring should be made as follows:

- Has the overall PCD1/2 system suffered any damage during transportation or assembly?
- Is the H3 module plugged into the space provided on the PCD1/2 bus and is the wiring complete on the relevant bus module?
- **Wiring and connection of user supply:**
  The H3 module must have a smoothed, +24 V DC supply voltage (19..32 V, max. ripple 10%) at the "+" and "-" terminals.
- **System earth**
  For fault-free operation, a perfect earth to divert external noise voltage is indispensable. The PCD1/2 system should be connected to the "GND" terminal (24 VDC -) with the largest possible cross-section on the earth rail of the control box. It should further be ensured that all earth lines have been laid without loops.
- **Laying the cables**
  Regulations prescribe the laying of high-voltage cables and control cables in separate cable channels.
- **Motor controller output (± 10V analogue)**
  Check connections as described in manual, chapter 5.5. The cable must be shielded.
- **Enable** for the driver is triggered through a "normal" digital output of the PCD1/2.
- **Encoder connections**
  For special attention with the 5V encoder:
  - Cable must be shielded and lines must be in twisted pairs.
  - Max. cable length 20m, min. conductor cross-section 0.25mm$^2$. In addition, check that encoder is correctly mounted (no slippage in coupling), check type and technical data (impulses per revolution).
- **Ref** (for PCD2.H310) wire directly
- **LS1 / LS2,** for PCD2.H311 also **Ref,** wire to "normal" digital inputs of PCD1/2.

## 9.3 Commissioning drive without motion control module

The drive alone (power stage and motor) is commissioned first, without the motion controller. The following measures are necessary:

- Release connection of the H3 module’s controller output "Out" (± 10V) to the power stage at terminals 6 and "-" (minus) of the H3 module.
- The PCD1/2 and H3 module have been switched off. If this is not possible, execute point 9.4.1 first (switching on the supply voltages).
- The drive’s emergency stop limit switches should be set to prevent any damage arising from uncontrolled axis motion.

Commissioning of the power stage and motor can now take place according to the supplier’s instructions.

## 9.4 Drive with motion control module

### 9.4.1 Switching on the supply voltages

This step can also take place before commissioning the power stage and motor, however, it is necessary to ensure that the axis cannot make any uncontrolled movements (power stage off-voltage, controller release interrupted).

When the supply voltage of the PCD1 or PCD2 and H310 or H311 module is powered up for the first time (programming unit disconnected) the control LEDs should be noted

on the PCD1:

| **Meaning** | **LED** | **Behaviour** |
|---|---|---|
| 24 VDC | yellow | 'Supply voltage present': must be on |
| Run | yellow | 'CPU in Run': not on, as no program has been loaded |
| Error | yellow | should not be on |

on the PCD2:

| **Meaning** | **LED** | **Behaviour** |
|---|---|---|
| 24 VDC | yellow | 'Supply voltage present': must be on |
| Battery | red | 'Battery' (fail): should not be on |
| Watch Dog | yellow | Watch Dog inactive: not on (as no program has been loaded). |
| Run | yellow | 'CPU in Run': not on, as no program has been loaded |
| Halt | red | 'CPU in Halt' is on, as no program is present |
| Error | yellow | should not be on |

on the motion control module PCD2.H310:

| **Meaning** | **LED** | **Behaviour** |
|---|---|---|
| Power | yellow | shows the presence of ± 15V: must be on |
| Ref (H310) | yellow | Reference switch |
| | | |

on the motion control module PCD2.H311:

| **Meaning** | **LED** | **Behaviour** |
|---|---|---|
| Power | yellow | shows the presence of ± 15V: must be on |
| Pw 5V | yellow | 5V supply for encoder: must be on |
| | | |

### 9.4.2 Preparing a basic user program

For commissioning a drive with the H3 module, a basic user program should be loaded into the PCD1/2 so that all checks and adjustments can be made. The "simple commissioning program" in section 6.1.3 can be used for initial trials. A comprehensive tool (Commissioning Tool), including limit and reference switches, is described in chapter 10.

Preparation of the user program comprises the following steps:

- Installation of the PCD9.H31E programming package, according to section 7.1.1.
- Opening a new project.
- Copying in the D2H310_B.MBA file.
- Adjusting the D2H310_B.MBA file with the number of H31x modules and their base addresses.
- Definition of the machine data.
- Adjusting (parameters in FB INIT and path to travel back and forth) and loading the commissioning program

**Evaluation of 'fPar_Err' flag**

Evaluation of this flag is particularly recommended for commissioning. This flag is only present once for all axes in a project.

After the flag has responded it is possible to view the cause of the error in the PCD register 'rDiag'. See section 8.2.1.

The user is responsible for resetting the 'fPar_Err' flag.

### 9.4.3 Determining machine data

Before commissioning for the first time, various parameters for FB 'Init' should be determined, some of which are already fixed by the machine, whereas other are recommended as test values.

Position tolerance → see FB 'Init', par. 7, manual page A-1/2

Recommended value 1 encoder revolution

When an encoder has 500 imp./revolution, this parameter has the value 2000 = 4 * 500 imp./revolution (evaluation of encoder impulse edges).

PID factors → see FB 'Init' manual page A-1/2

Recommended start values:

| | |
|---|---|
| Proportional factor (Parameter 2): | 10 |
| Integral factor (Parameter 3): | 0 |
| Differential factor (Parameter 4): | 0 |
| Integration limit (Parameter 5): | 30000 |
| Sample time D portion (Parameter 6): | 15 (= 5.61 msec.) |

Machine factor FB 'Init', PCD register for parameter 9 (floating-point format)

This factor results from the resolution of the encoder used, the spindle gradient and any gears.

Important: The unit of measurement chosen here (m, cm, mm, 1/10mm, 1/100mm, µm) must also be used for the velocity (par. 10), acceleration (par. 11) and all values relating to path, break-point, velocity and acceleration throughout the user program for this axis.

Assume: $k = \frac{4 \times In}{s}$ [impulses/unit of measurement]

where In: impulses/revolution (encoder resolution)
s: path/revolution (spindle gradient and gears)

| | | |
|---|---|---|
| Example: | In | = 1000 impulses per revolution |
| | s | = 2 mm (spindle gradient) (no gears) |

$$k = \frac{4 \times 1000}{2} = 2000 \text{ impulses/for 1 mm}$$

$$= 2 \text{ impulses for 1 } \mu\text{m}$$

If the chosen unit of measurement is 1 mm, a value of 2000.0 (floating-point format) should be written in the PCD register for parameter 9. If the chosen unit of measurement is 1 µm, a value of 2.0 (floating-point format) should be entered.

Velocity FB 'Init', PCD register for parameter 10

When commissioning for the first time, it is advisable to work with a low velocity. The value should be entered as an integer in the PCD register provided, using the same unit of measurement as the machine factor.

Recommendation: 0.05 m/s $\rightarrow$ 50 mm/s or 50000 µm/s

Ensure that the chosen drive can actually achieve the velocity of its parameter, otherwise the controller continuously detects a deviation from the target velocity and adds up these deviations. This will then lead to an incorrect braking ramp or to an abrupt halt of the motion.

Acceleration FB 'Init', PCD register for parameter 11

When commissioning for the first time, it is advisable to work with a low acceleration. The value should be entered as an integer in the PCD register provided, using the same unit of measurement as the machine factor.

Recommendation: 0.01 $m/s^2$ $\rightarrow$ 10 $mm/s^2$ or 10000 $\mu m/s^2$

Before starting any motion, check encoder function by manually moving the axis (amplifier powered off). See also section 9.4.5.

### 9.4.4 Direction, path measurement (encoder)

To check direction of rotation and path measurement, the following pre-conditions must be met:

- The connection between the H3 module’s controller output (± 10V) and the power stage has been released at terminals 6 and GND for the H3 module’s axis.
- The PCD1/2 has been switched on (commissioning program loaded) and is in "Run".
- The programming unit is connected and the commissioning software has been started. All the necessary adjustments have been made in the "Configure" menu. These settings must agree with definitions in the D2H310_B.MBA file.

  For the following tests, a display of actual position is viewed.

**Checking the direction:**

1. Rotate drive shaft in a positive direction
   → actual position must increment

2. Rotate drive shaft in a negative direction
   → actual position must decrement

If this is possible, the drive shaft can be turned by hand. Otherwise it must be moved by applying a set value (using a voltage source of ± 0.5 .. 10V) to the power stage.

Definition of positive and negative directions:

- Positive direction corresponds to the direction moved when a positive setpoint voltage is applied (0...+10V) at the power stage.
- Negative direction corresponds to the direction moved when a negative setpoint voltage is applied (0...-10V) at the power stage.
- If behaviour is the reverse, both phase signals A and B of the encoder should be exchanged.

**Checking path measurement:**

Turn drive shaft through one revolution and observe display of actual position.

A path value corresponding to the machine factor k (parameter 9 in FB 'Init') must be displayed as the actual position. If the display is incorrect, recheck the calculation and entry of machine factor 'k'.

**Checking the index signal IN:**

This check is only necessary if the index signal (also called zero impulse) is evaluated by the encoder to define the zero position.

Procedure:
(All the following actions are carried out manually by commands from the commissioning program.)

1. Observe display of index position register → any value is displayed.
2. Execute the instruction 'SetIdxPos'.
   → When the next index impulse is captured, the actual position is written in the index position register.
3. Turn the axis until the actual position is written in the index position register.
   The position must be written to the register once within an encoder revolution.
4. Execute the instruction "SetIdxPos" again and check whether, during a revolution, the current position is written once only to the register.

If the check produced another result, this may be due to the following:

- Faulty encoder
- Encoder signals A, B and IN are not in the order required at the H3 module’s position decoder input.

If the order of encoder signals is not known, it must be established with the help of an oscilloscope.

### 9.4.5 PID controller

To check and tune the PID controller, the following preconditions must be met:

- With the controller powered off, the control circuit is closed by re-connecting the setpoint cable of the power stage to terminals 6 and GND of the H3 module.
- Power up the controller:
  CAUTION: Activate emergency stop if the axis should proceed out of control.
- The programming unit is connected and commissioning software has been started.

Checks:

1. Load destination position 0 and execute start instruction.
   → The H3 module regulator is switched on and the axis is held in its position by the regulator.

2. Load a destination position that is located within the allowed travelling range and start motion.
   → The axis runs towards the fixed destination position.

Incorrect behaviour:

- The axis runs at maximum velocity.

| | |
|---|---|
| Possible cause | • Position control circuit (H3 module) or speed control circuit (power stage) incorrectly poled. This means that, when the setpoint voltage is positive (= positive destination position) motion in the axis is in a negative direction.<br>• The H3 module's analogue output is faulty and supplies a constant, maximum voltage of approx. +12V or –12V to the controller output. |
| Remedy | • Check direction, execute section 9.4.4 once again and make the necessary changes to poling.<br>• Change the H3 module. |

- The axis runs at a constantly slow velocity (drift).

| | |
|---|---|
| Possible cause | • The setpoint is not reaching the power stage, regulation is not active in the H3 module. |
| Remedy | • Check wiring.<br>• Check whether, after the 'StartMot' instruction, the profile generator in the H3 module outputs a target velocity (instruction 'RdTargVel'), which means that the regulator in the H3 module is working correctly. |

- The axis starts running briefly, and stops again.

| | |
|---|---|
| Possible cause | • Position error monitoring has responded and stopped the drive. (The motor cannot follow the motion defined) |
| Remedy | • Correct possible mechanical problems.<br>• Adjust velocity and acceleration.<br>• Increase max. value for position error. |

- The axis runs jerkily to an incorrect destination position.

| | |
|---|---|
| Possible cause | • Loose mechanical connection of encoder and motor (coupling). |
| Remedy | • Correct possible mechanical problems. |

When approaching the destination position a persistent position error is noticeable. This is accounted for by the permanent control deviation in a straight P controller (I and D factors are 0).

In a subsequent step, therefore, control factors must be determined which achieve the desired quality of regulation (accuracy and hardness).

The set-up rules indicated below have arisen from experience in practical applications and tests.

**Setting the proportional factor (kp):**

1. Set a small proportional factor (experience recommends 10). Integral and differential factors must be 0 ($\rightarrow$ straight P controller).
2. Travel slowly with the axis (approach destination position or use the 'GoForw'/'GoBackw' functions)
3. Increase the kp factor gradually until the control circuit starts to pulse. Then reduce that value by approx. 30% and load it as the kp factor. This sets the proportional factor and should not be changed further for the time being.

**Setting the integral factor (ki):**

The ki factor is gradually increased until the desired settling time is achieved for the position error. This sets the integral factor and should not be changed further for the time being.

If excessive overshoot is noticed when approaching the destination position, this may be due to the following:

- The chosen rate of acceleration or braking is too high. The motor cannot follow the setpoint. $\rightarrow$ Reduce acceleration.
- The same behaviour can also result when the chosen velocity is too high.
- The chosen ki factor is too high.
- Practice has shown that, if the integral factor is increased and the differential factor is left at zero, the tendency of the controller to pulse rises. $\rightarrow$ The differential factor should be increased simultaneously with the integral factor to reduce overshoot.

Observing the integration sum (instruction 'RdIntSum') is a good way of tracking such behaviour. During motion, if the integration sum adds up to a high value, overshoot can be expected.

| | |
|---|---|
| Action: | - Reduce the above parameters. |
| | - Limit the integration sum (instruction 'LdIntLim'). |

**Setting the differential factor (kd):**

The kd factor is gradually increased until the desired overshoot width or settling time is achieved for the position error. This also sets the differential factor.

Set-up rule for sample time (instruction 'LdSampInt') in the D portion:
For operation at low velocities, opt for a rather large sample time.
Experience shows that sample time values generally lie between 2 ms and 9 ms. This corresponds to values 5 and 25 in the instruction 'LdSampInt'.

**Optimization of settings:**

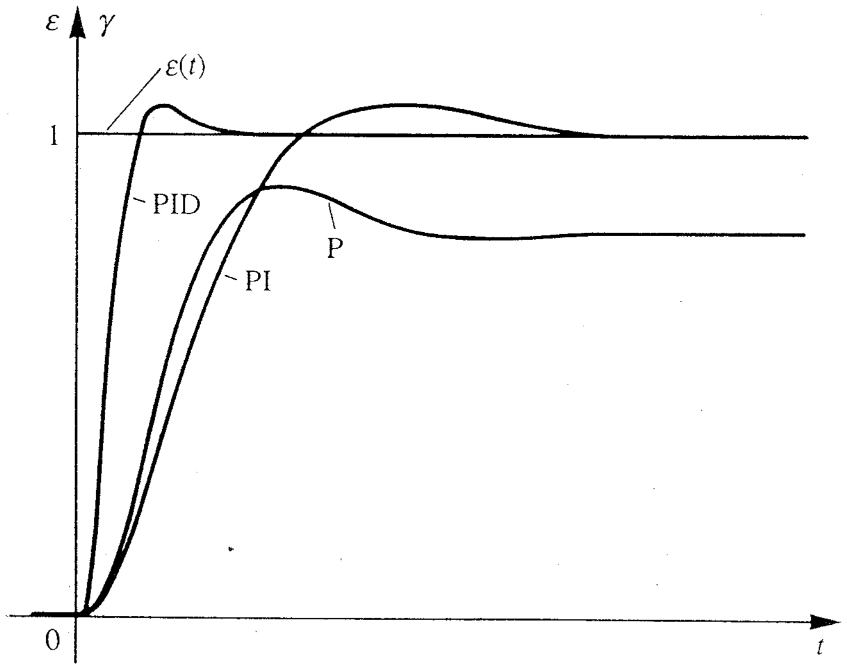
If regulation does not proceed satisfactorily after values have been set as above, a further attempt must be made to vary individual parameters, so that a satisfactory result is achieved.

### 9.4.6 Effect of individual factors on controller behaviour

Increasing the **kp factor** heightens the tendency to overshoot but equally reduces the permanent control deviation and increases controller hardness.

Raising the **ki factor** heightens the tendency to overshoot, but simultaneously ensures faster settling of the position error.

A correctly adjusted **kd factor** stabilizes the system and ensures less overshoot and a shorter settling time. If the kd factor is too high, it produces pulsing in the system.



### 9.4.7 Simple commissioning program

A convenient, graphical commissioning tool in FUPLA is described in chapter 10.

The following is an extremely simple user program for forward and backward motion with actual position display, using a single axis. The parameters suggested above have been applied:

(Explanatory notes on this user program can be referred to in section 6.1.1: "Entry-level example in IL with wait loops").

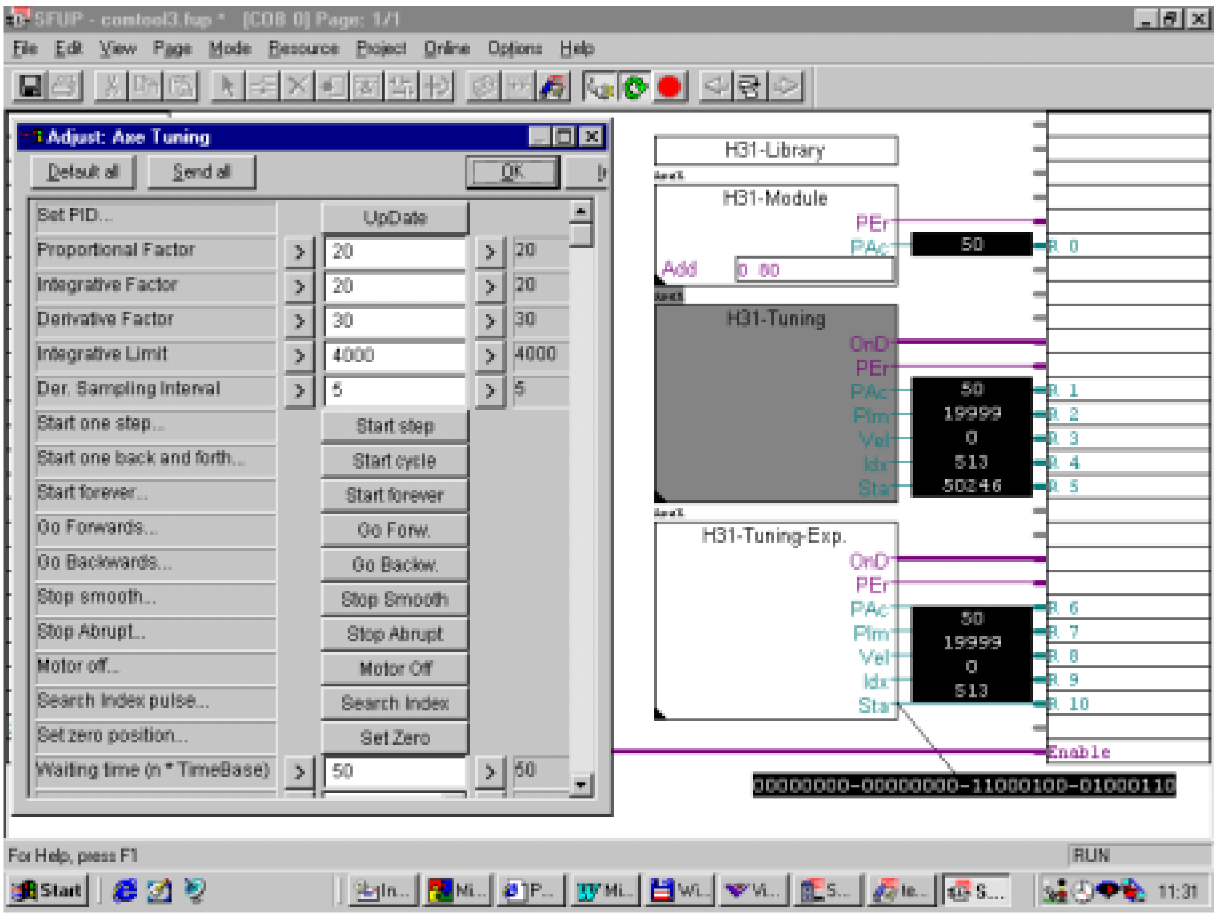
```
$include D2H310_B.equ
$group H310

            xob    16

            LD   R 1000
                   2.0       ; Mechanical factor
            LD   R 1001
                   50000     ; Initial absolute speed
            LD   R 1002
                   10000     ; Initial absolute acceleration

            CFB    Init      ; Initialization FB
                 K 1         ;  Module number
                   10        ;  Proportional factor (regulator)
                   0         ;  Integrative factor (regulator)
                   0         ;  Derivative factor (regulator)
                   30000     ;  Integrative limit value
                   15        ;  Derivative term sampling interval
                   2000      ;  Position tolerance
                   1         ;  Behaviour in case of position error
                 R 1000      ;  Mechanical factor register
                 R 1001      ;  Initial velocity register
                 R 1002      ;  Initial acceleration register

            exob
;           ----------------------------------------------------------
;
            cob    0
                   0

start:      sth  i 0         ; 'Start' OK?

            CFB    Exec      ; Executable FB
                 K 1         ;  Module number
                   RdActPos  ;  Command: Read Actual Position
                 R 90        ;  Result register
            DSP  R 90        ; Display register

            jr   l start     ; if 'Start' not OK, wait

            LD   R 100       ; Target Position
                   10000     ;  Absolute value

            CFB    Exec      ; Executable FB
                 K 1         ;  Module number
                   LdDestAbs ;  Command: Load Absolute Destination
                 R 100       ;  Absolute Destination register
```

```
pos1:       CFB    Exec      ; Executable FB
                 K 1         ;  Module number
                   StartMot  ;  Command: Start motion
                   rNotUsed  ;  Dummy register

            CFB    Exec      ; Executable FB
                 K 1         ;  Module number
                   RdActPos  ;  Command: Read Actual Position
                 R 90        ;  Actual Position register
            DSP  R 90        ; Display register

            CFB    Exec      ; Executable FB
                 K 1         ;  Module number
                   RdStatRg  ;  Command: Read Status Register
                 R 0         ;  Value of Status Register

            STH    fOnDest_1 ; Position reached?
            jr   l pos1      ; if no, wait

            ld   t 0
                   50
pause1:     sth  t 0
            jr   h pause1

            LD   R 100        ; Target Position
                   0          ;  Absolute value

            CFB    Exec       ; Executable FB
                 K 1          ;  Module number
                   LdDestAbs  ;  Command: Load Absolute Destination
                 R 100        ;  Absolute Destination register

            CFB    Exec       ; Executable FB
                 K 1          ;  Module number
                   StartMot   ;  Command: Start motion
                   rNotUsed   ;  Dummy register

pos2:       CFB    Exec       ; Executable FB
                 K 1          ;  Module number
                   RdActPos   ;  Command: Read Actual Position
                 R 90         ;  Result register
            DSP  R 90         ; Display register

            CFB    Exec       ; Executable FB
                 K 1          ;  Module number
                   RdStatRg   ;  Command: Read Status Register
                 R 0          ;  Value of Status Register

            STH    fOnDest_1  ; Position reached?
            jr   l pos2       ; if no, wait

            ld   t 0
                   50
pause2:     sth  t 0
            jr   h pause2

            ecob

$endgroup
```

# 10. Commissioning Tool

The PCD8.H31 diskette includes a programming tool for use during commissioning entitled 'comtool-fup'. This tool offers a moderate level of user comfort and is based on the PG4’s FUPLA program from version V2.0.70.

The program consists of an organization FBox, an FBox for the initialization of a PCD2.H31x, an FBox with a simple path program enabling the online modification of all PID parameters (for the purpose of optimizing control behaviour) and all path parameters, plus a final FBox that allows the creation and online modification of a path program with any 4 sequences. The optimum PID parameters determined in this way are transferred to the final user program (FB 'Init' and possibly an additional FB 'Exec' to adjust individual PID parameters).



| | | | |
|---|---|---|---|
| Set PID... | UpDate | | |
| Proportional Factor | 20 | | 20 |
| Integrative Factor | 20 | | 20 |
| Derivative Factor | 30 | | 30 |
| Integrative Limit | 4000 | | 4000 |
| Der. Sampling Interval | 5 | | 5 |
| Start one step... | Start step | | |
| Start one back and forth... | Start cycle | | |
| Start forever... | Start forever | | |
| Go Forwards... | Go Forw. | | |
| Go Backwards... | Go Backw. | | |
| Stop smooth... | Stop Smooth | | |
| Stop Abrupt... | Stop Abrupt | | |
| Motor off... | Motor Off | | |
| Search Index pulse... | Search Index | | |
| Set zero position... | Set Zero | | |
| Waiting time (n * TimeBase) | 50 | | 50 |

H31-Module: PAc 50 → R 0; Add 0 80

H31-Tuning: PAc 50 → R 1; Pim 19999 → R 2; Vel 0 → R 3; Idx 513 → R 4; Sta 50246 → R 5

H31-Tuning-Exp.: PAc 50 → R 6; Pim 19999 → R 7; Vel 0 → R 8; Idx 513 → R 9; Sta → R 10

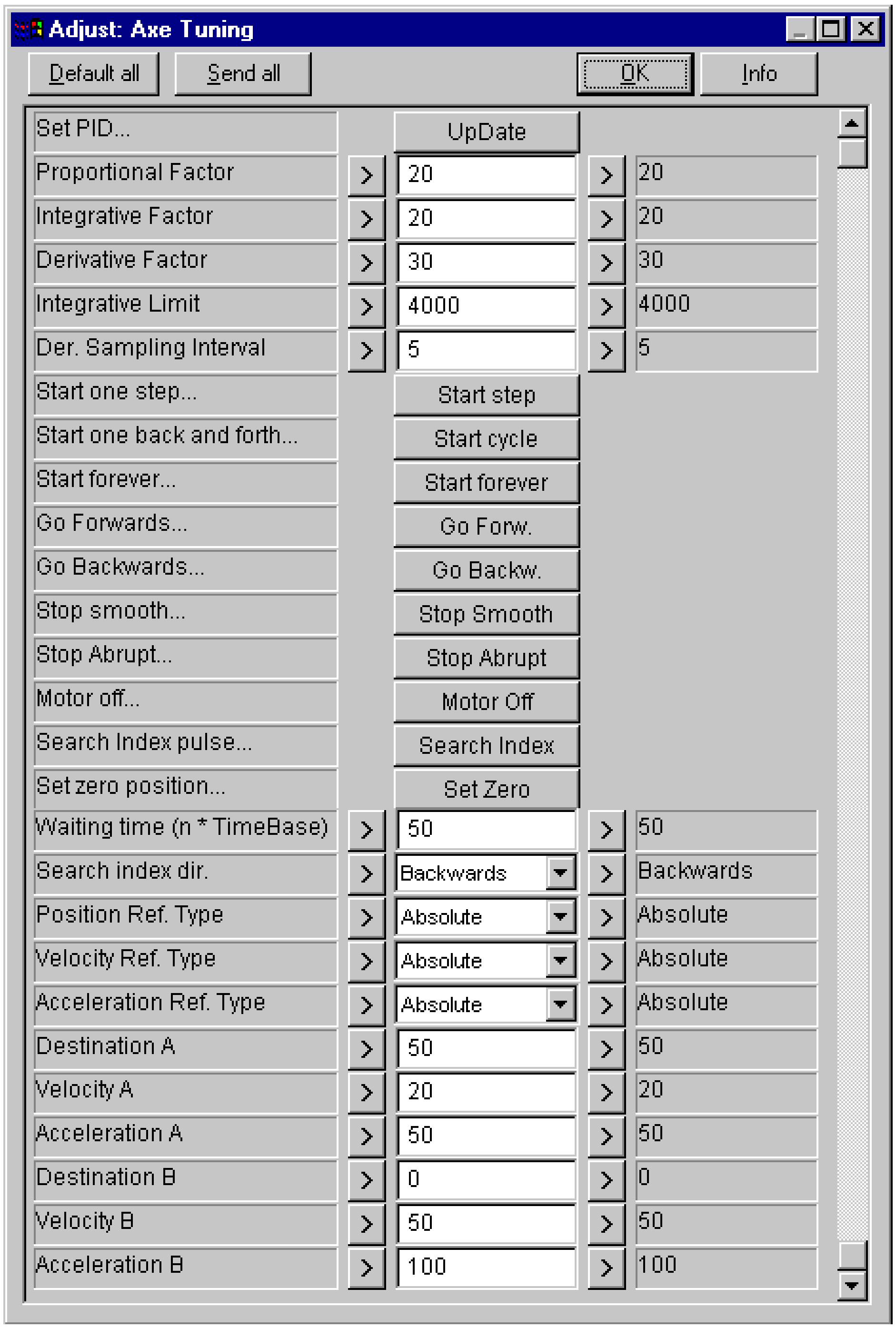
00000000-00000000-11000100-01000110

This is how the tool appears on the screen. The adjust window for the FBox with the simple path program is open.

PID parameters are set individually, transferred separately and then activated together with 'Update'.

The following keys are used to activate a single path (Start Step), back-and-forth path (Start Cycle) or continuous back-and-forth path (Start forever). The 'Waiting time' parameter can be used to define a pause after each sequence in the path program. It is also possible to travel forwards or backwards as desired (Go forward or Go backward) and to stop travel smoothly (Stop smooth) or abruptly (Stop abrupt). With 'Set Zero' any position can be declared as the new zero position. It is also possible to switch off the motor control (Motor off) or find the next index position.

The adjust window of the previously described FBox for simple back-and-forth travel by the carriage is shown below in full size.



Adjust: Axe Tuning

Default all | Send all | OK | Info

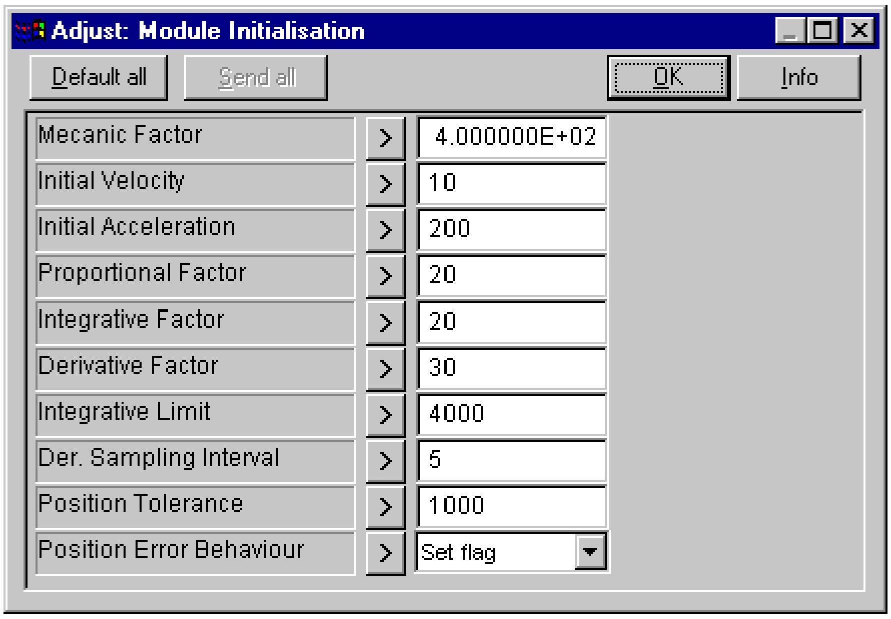
| | | | | | |
|---|---|---|---|---|---|
| Set PID... | | UpDate | | | |
| Proportional Factor | > | 20 | > | 20 | |
| Integrative Factor | > | 20 | > | 20 | |
| Derivative Factor | > | 30 | > | 30 | |
| Integrative Limit | > | 4000 | > | 4000 | |
| Der. Sampling Interval | > | 5 | > | 5 | |
| Start one step... | | Start step | | | |
| Start one back and forth... | | Start cycle | | | |
| Start forever... | | Start forever | | | |
| Go Forwards... | | Go Forw. | | | |
| Go Backwards... | | Go Backw. | | | |
| Stop smooth... | | Stop Smooth | | | |
| Stop Abrupt... | | Stop Abrupt | | | |
| Motor off... | | Motor Off | | | |
| Search Index pulse... | | Search Index | | | |
| Set zero position... | | Set Zero | | | |
| Waiting time (n * TimeBase) | > | 50 | > | 50 | |
| Search index dir. | > | Backwards | > | Backwards | |
| Position Ref. Type | > | Absolute | > | Absolute | |
| Velocity Ref. Type | > | Absolute | > | Absolute | |
| Acceleration Ref. Type | > | Absolute | > | Absolute | |
| Destination A | > | 50 | > | 50 | |
| Velocity A | > | 20 | > | 20 | |
| Acceleration A | > | 50 | > | 50 | |
| Destination B | > | 0 | > | 0 | |
| Velocity B | > | 50 | > | 50 | |
| Acceleration B | > | 100 | > | 100 | |

As before, PID parameters are set individually, transferred separately and then activated together with 'Update'.

Parameters in the lower section, i.e. those for the path program, are transferred immediately after confirmation.

Adjust window of the initialization FBox.

Parameters are only transferred after a download. The parameters of this adjust window, therefore, should be set **before** compiling and **before** downloading to the PCD.

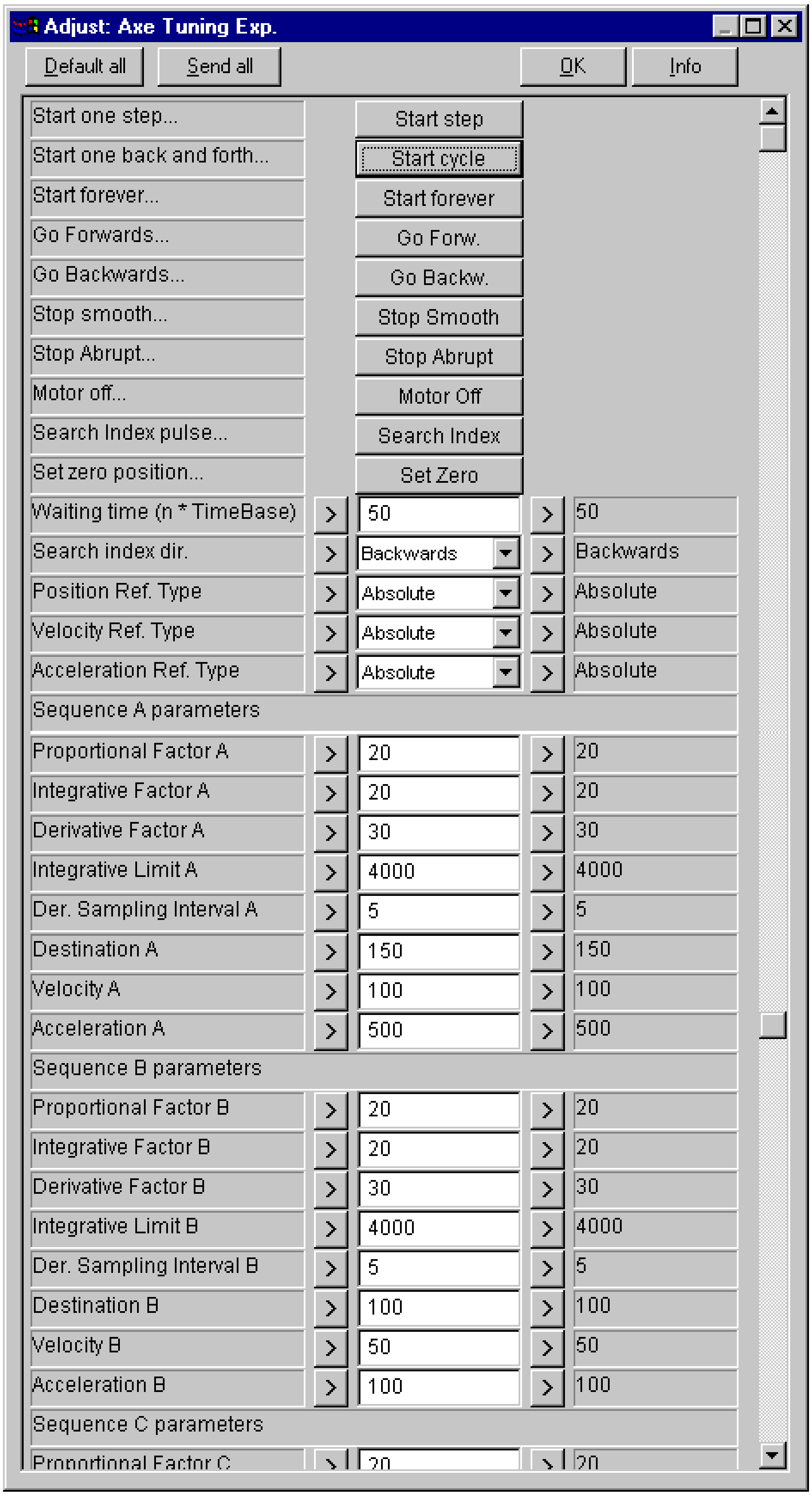


| | |
|---|---|
| Mecanic Factor | 4.000000E+02 |
| Initial Velocity | 10 |
| Initial Acceleration | 200 |
| Proportional Factor | 20 |
| Integrative Factor | 20 |
| Derivative Factor | 30 |
| Integrative Limit | 4000 |
| Der. Sampling Interval | 5 |
| Position Tolerance | 1000 |
| Position Error Behaviour | Set flag |

The last of the 4 FBoxes can be used to create a path program with any 4 desired sequences. In each sequence all parameters can be set individually and modified online.

Operation is the same as for the simple 'back-and-forth' program. With the 'Start Step' button, individual sequences are set off one after the other. 'Start Cycle' is used to start a single complete sequence and 'Start for ever' keeps the cycle of sequences moving until 'Stop ...' or 'Motor Off' interrupts it. The 'Waiting time' parameter can be used to define a pause after each sequence. It is also possible to move back or forth as desired (Go forward or Go backward) and to bring the path to a halt smoothly (Stop smooth) or abruptly (Stop abrupt). 'Set Zero' enables any position to be declared as the new zero position. It is also possible to switch the motor control off (Motor off) or to seek the next index position (Backwards or Forwards). Positions, velocity and acceleration can be set with 'Absolute' or 'Relative' parameters.

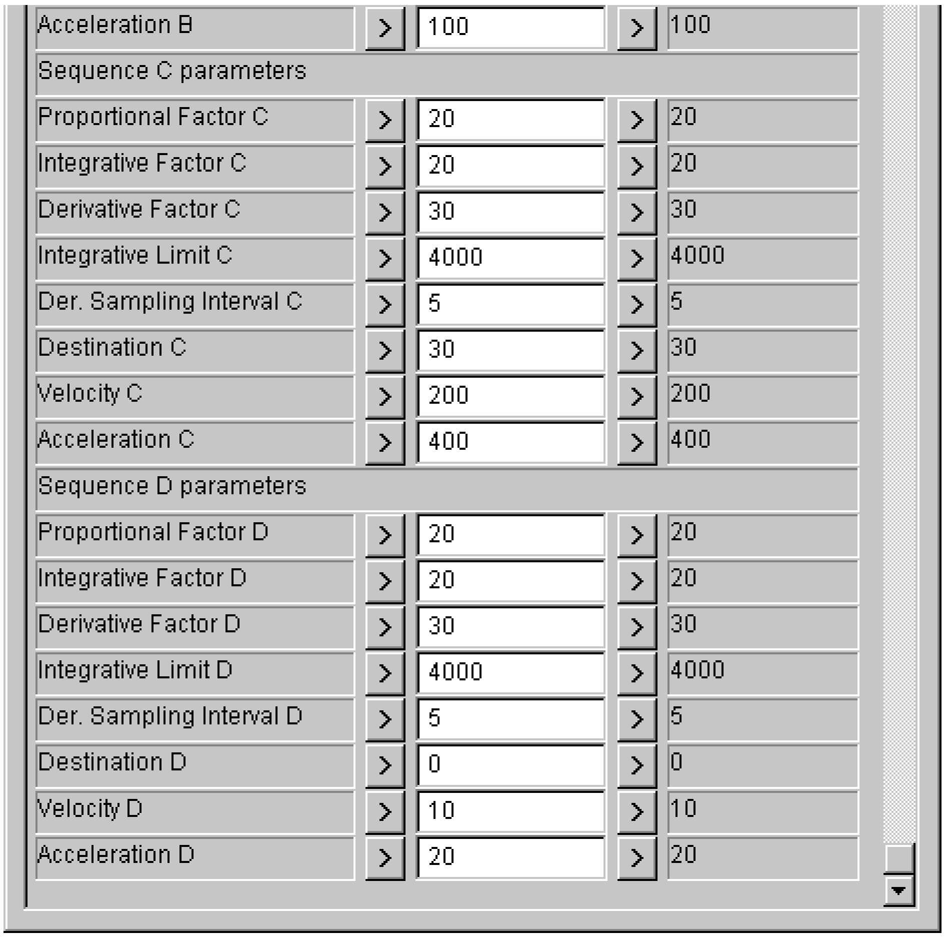
The adjust window for the FBox with a 4-sequence cycle appears as follows: (The values entered serve only as an example, but work well with the V-PCX 24 demonstration model).


Adjust: Axe Tuning Exp.

Default all | Send all | OK | Info

|  |  |  |
|---|---|---|
| Start one step... | Start step | |
| Start one back and forth... | Start cycle | |
| Start forever... | Start forever | |
| Go Forwards... | Go Forw. | |
| Go Backwards... | Go Backw. | |
| Stop smooth... | Stop Smooth | |
| Stop Abrupt... | Stop Abrupt | |
| Motor off... | Motor Off | |
| Search Index pulse... | Search Index | |
| Set zero position... | Set Zero | |
| Waiting time (n * TimeBase) | 50 | 50 |
| Search index dir. | Backwards | Backwards |
| Position Ref. Type | Absolute | Absolute |
| Velocity Ref. Type | Absolute | Absolute |
| Acceleration Ref. Type | Absolute | Absolute |
| Sequence A parameters | | |
| Proportional Factor A | 20 | 20 |
| Integrative Factor A | 20 | 20 |
| Derivative Factor A | 30 | 30 |
| Integrative Limit A | 4000 | 4000 |
| Der. Sampling Interval A | 5 | 5 |
| Destination A | 150 | 150 |
| Velocity A | 100 | 100 |
| Acceleration A | 500 | 500 |
| Sequence B parameters | | |
| Proportional Factor B | 20 | 20 |
| Integrative Factor B | 20 | 20 |
| Derivative Factor B | 30 | 30 |
| Integrative Limit B | 4000 | 4000 |
| Der. Sampling Interval B | 5 | 5 |
| Destination B | 100 | 100 |
| Velocity B | 50 | 50 |
| Acceleration B | 100 | 100 |
| Sequence C parameters | | |
| Proportional Factor C | 20 | 20 |

(Adjust window, continued)

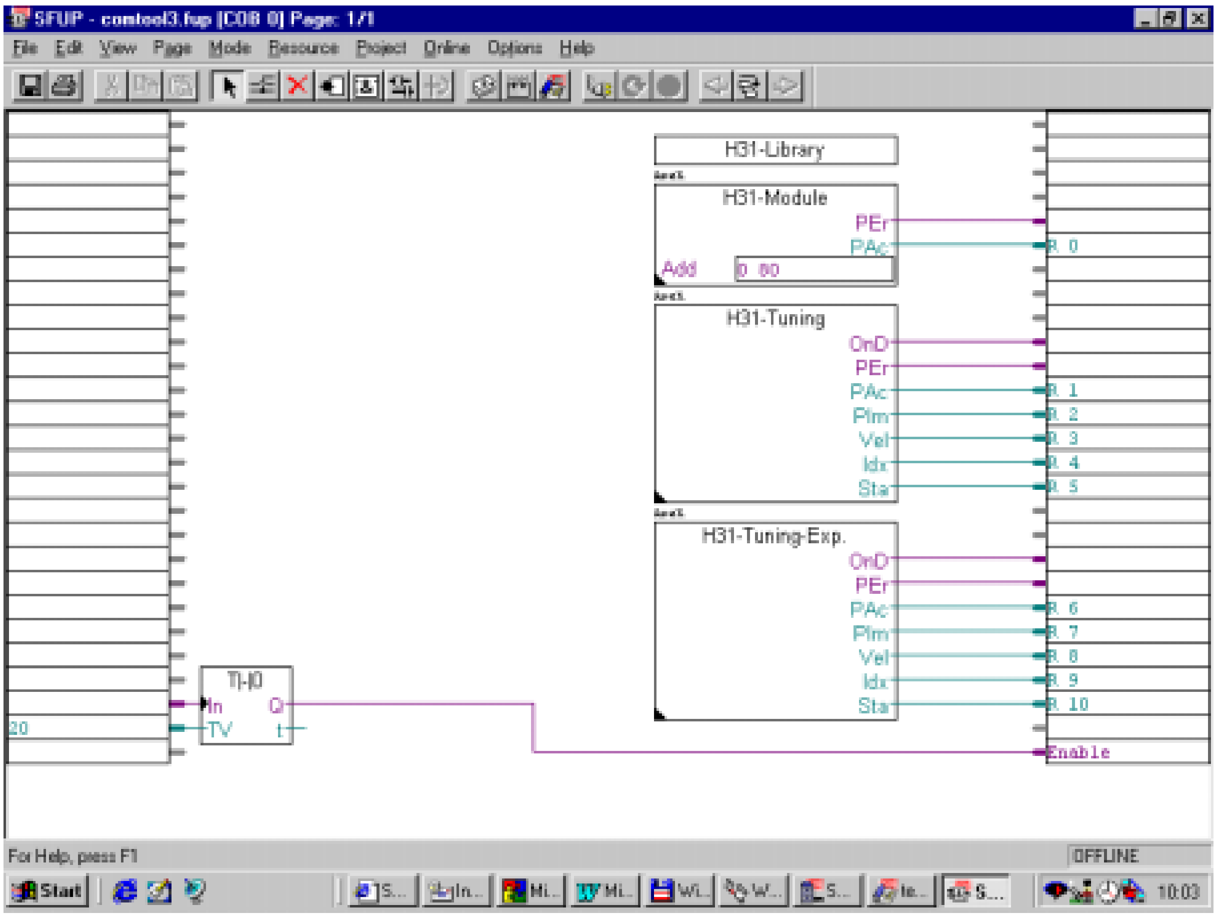


| | | | | |
|---|---|---|---|---|
| Acceleration B | > | 100 | > | 100 |
| Sequence C parameters | | | | |
| Proportional Factor C | > | 20 | > | 20 |
| Integrative Factor C | > | 20 | > | 20 |
| Derivative Factor C | > | 30 | > | 30 |
| Integrative Limit C | > | 4000 | > | 4000 |
| Der. Sampling Interval C | > | 5 | > | 5 |
| Destination C | > | 30 | > | 30 |
| Velocity C | > | 200 | > | 200 |
| Acceleration C | > | 400 | > | 400 |
| Sequence D parameters | | | | |
| Proportional Factor D | > | 20 | > | 20 |
| Integrative Factor D | > | 20 | > | 20 |
| Derivative Factor D | > | 30 | > | 30 |
| Integrative Limit D | > | 4000 | > | 4000 |
| Der. Sampling Interval D | > | 5 | > | 5 |
| Destination D | > | 0 | > | 0 |
| Velocity D | > | 10 | > | 10 |
| Acceleration D | > | 20 | > | 20 |

For the best possible commissioning, a storage oscilloscope to display the ±10V control voltage should also be available, as this is the only way of viewing controller behaviour under the various loads and optimizing its settings. Unfortunately, the present commissioning tool does not include this display.

## Description of the commissioning tool's individual FBoxes

The commissioning tool appears as follows:



The delay time on the lower margin is not present in the original tool. It is advisable not to release the motor control until the CPU's self diagnostic is completed. This prevents the carriage from making any uncontrolled jump when the controller is switched on.

**FBox 'H31-Library'**
Organization FBox. Must be located at the beginning of a tool page. This FBox has no inputs or outputs and no adjust window either.

**FBox 'H31-Module'**
Initialization of PCD2.H31x module.
The module base address should be written in the 'Add' window, e.g. O 80.

FBox outputs:

| | | |
|---|---|---|
| 'PEr' | binary | Position error |
| 'PAc' | integer | Actual position |

Adjust window: see page 10-4

**FBox 'H31-Tuning'**
Simple back-and-forth path program to seek and optimize the best possible parameters.

FBox outputs:

| | | |
|---|---|---|
| 'OnD' | binary | On destination |
| 'PEr' | binary | Position error |
| 'PAc' | integer | Actual position (in units of length) |
| 'PIm' | integer | Actual position (in impulses) *) |
| 'Vel' | integer | Actual velocity |
| 'IdX' | integer | Index register |
| 'Sta' | integer | Status register |

Adjust window: see page 10-3

**FBox 'H31-Tuning-Exp.'**
FBox with path program comprising any 4 sequences, whose parameters can be set individually.

FBox outputs:

| | | |
|---|---|---|
| 'OnD' | binary | On destination |
| 'PEr' | binary | Position error |
| 'PAc' | integer | Actual position (in units of length) |
| 'PIm' | integer | Actual position (in impulses) *) |
| 'Vel' | integer | Actual velocity |
| 'IdX' | integer | Index register |
| 'Sta' | integer | Status register |

adjust window: see page 10-6/7

*) The 'PIm' output shows the result of multiplying the 'PAc' output value by the 'Mechanical Factor'.

Notes

# 11. Security aspects

If a drive unit is triggered with a PCD2.H31x module, particular attention should be paid to the following points:

**Drive power-up phase**

When the main supply for the PCD has been switched on, it takes max. 2 seconds before all input/output modules are initialized, the CPU is in RUN and the user program can therefore be processed.

During this power-up phase (max. 2 seconds), the analogue controller output of the H3 module have any value between -10V and +l0V. For this reason it is absolutely necessary that the power section of the drive should either be switched on or released by the user program through a digital output. This guarantees that the drive does not proceed out of control during this power-up phase.

**Monitoring the position error**
(See flag 'fPosErr_x')

Exceeding the permitted position error signals serious problems and should therefore always be monitored.

The following causes can result in exceeding the position error:

1. Connection fault in the H3 installation.
   Examples: - loose connection
   - directional mismatch in rotation of motor and incremental shaft encoder.
2. Badly adjusted PID parameters
3. The size of drive (amplifier and/or motor) is not strong enough.
4. Wrong choice of motion parameters. The servo-amplifier or motor cannot follow the acceleration or velocity ordered by the H3 module.
5. Blocked rotor in servomotor due to mechanical problems.
6. Hardware error in H3 module (e.g. faulty analogue controller output).
7. Hardware error in servo-amplifier.

If points 1 to 4 account for the position error being exceeded, this is usually discovered during commissioning and can be remedied immediately.

If points 5 to 7 account for the position error being exceeded, this can also occur after commissioning. To avoid damaging the machine, the following measures are necessary:

Permanently monitor the 'fPosErr_x' flag (which signals that the position error has been exceeded) from the user program and, in case of error, disconnect the drive via a digital output.

There are different ways of disconnecting the drive. With some drives it is enough to withdraw the controller release at the power section. With others again it is necessary to reduce drive speed as quickly as possible by braking (shorting out the setpoint input at the power section) and to disconnect  the main supply with a time delay (of a few milliseconds). However this disconnection can or must take place depends on the machine concerned and must be decided on a case by case basis.

**Limit Switches**

These can be used in different ways:

- in connection with the 'Home' function to find the zero position when the installation is powered up
- as an alarm message in the user program to signal that the position has been exceed (if appropriately located and programmed)

**Security limit switches**

The security limit switches (emergency-off limit switches) should always disconnect the drive directly. (Cut main supply to drive.)

**Watchdog**

Always activate the PCD’s watchdog and use the watchdog contact to disconnect the drive directly.

**X0Bs for PCD hardware errors**

Program XOBs 0 to 5 and, if necessary, disconnect the drive directly via a digital output.

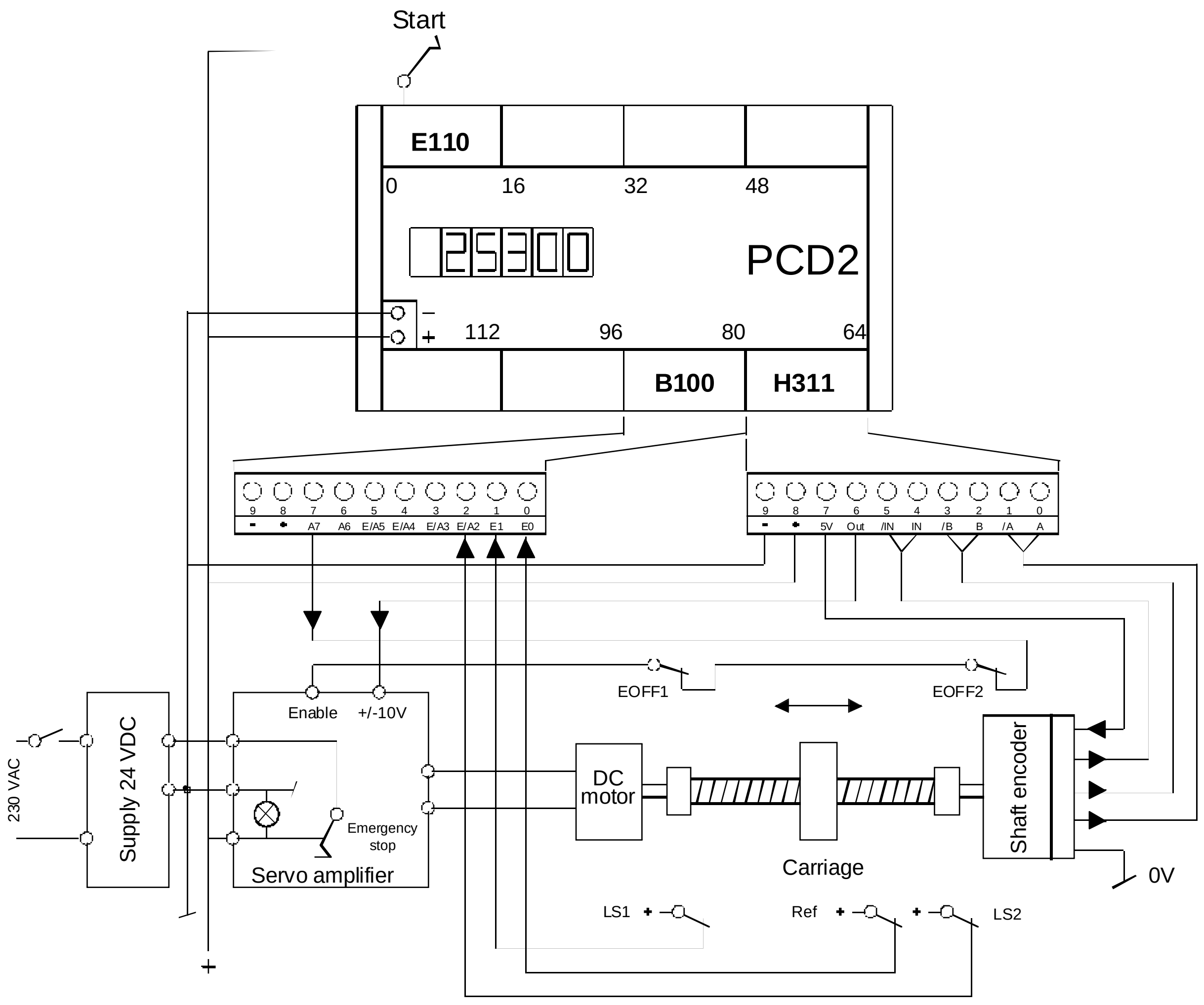
# 12. Application examples

## 12.1 Example: 1 axis with homing

This example concerns the simple backward and forward motion of a carriage as already described in chapter 6: "Quick start". The present example additionally includes the reference switch and both limit switches at either end of the range of travel for automatic homing.

The hardware is based on workshop model V-PCX 24 and consists of the slide with DC motor, reference switch, limit switches and final limit switches, the servo-amplifier with emergency switch, the supply and a PCD2 with 6-digit display.

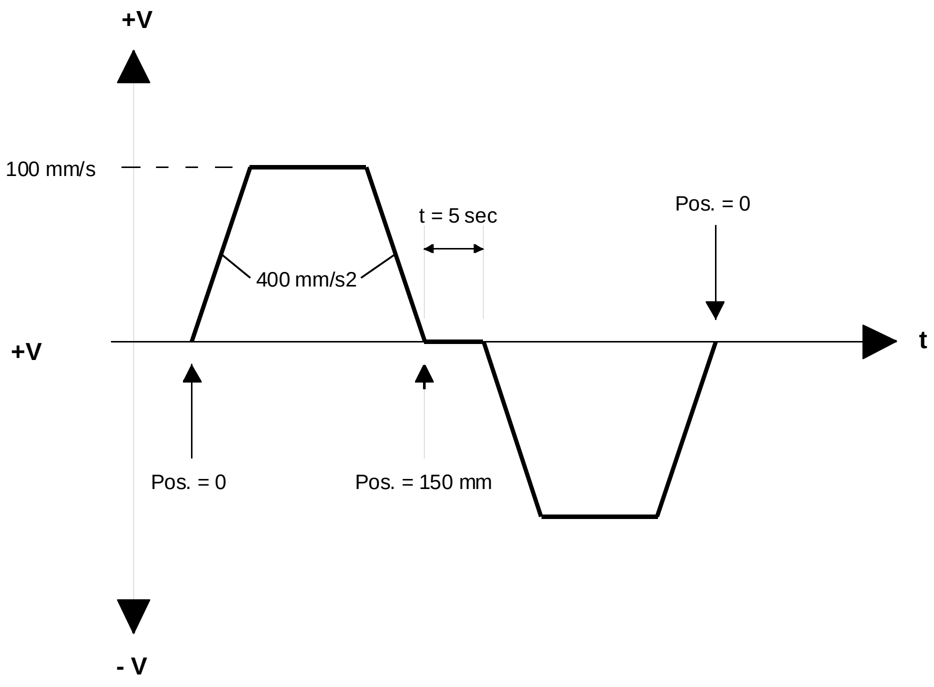
When fully assembled and wired up, the model has the following presentation:



Data for carriage:

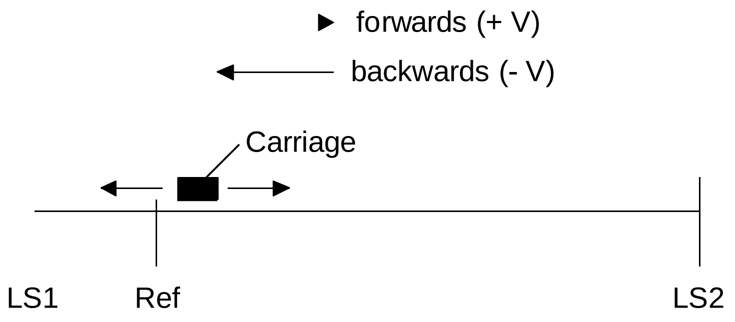
| | |
|---|---|
| Encoder: | 500 impulses/revolution |
| Spindle: | gradient 2 mm |
| Maximum velocity: | 100 mm/s |
| Max. permissible acceleration: | 50 mm/s$^2$ |

After homing, directly following system power-up, the following motion sequence should be executed:



**Homing**

Before presenting the overall user program, homing should be looked at more closely.



The arrangement of switches on the model is roughly in line with the above drawing. During normal program execution the carriage is in the position illustrated.

An assumption is made that, in normal cases, the carriage is located between the positions 'Ref' and 'LS2'. For homing, the carriage should travel in the direction of the reference switch, i.e. searching in a 'backwards' direction. On reaching the reference switch it should travel freely in a forwards direction until it arrives at the next index signal of the incremental shaft encoder. This means that the reference position has been reached and the travel program can start.

For this purpose, the following parameters should be set for FB 'Home':

```
LD    R 1010       ; PCD register with velocity for free
        500        ;  travel of the reference switch (slow)

LD    R 1011       ; PCD register with velocity for seeking
        5000       ;  the reference switch (fast)

CFB     Home       ; Call FB 'Home'
      K 1          ;  Par 1: Module number
        0          ;  Par 2: Search backwards
        1          ;  Par 3: Free travel forwards
      R 1010       ;  Par 4: Velocity for free travel
      R 1011       ;  Par 5: Velocity for search
        30         ;  Par 6. Timeout 30s
      I 64         ;  Par 7: Reference input address
```

<u>Behaviour under various starting positions:</u>

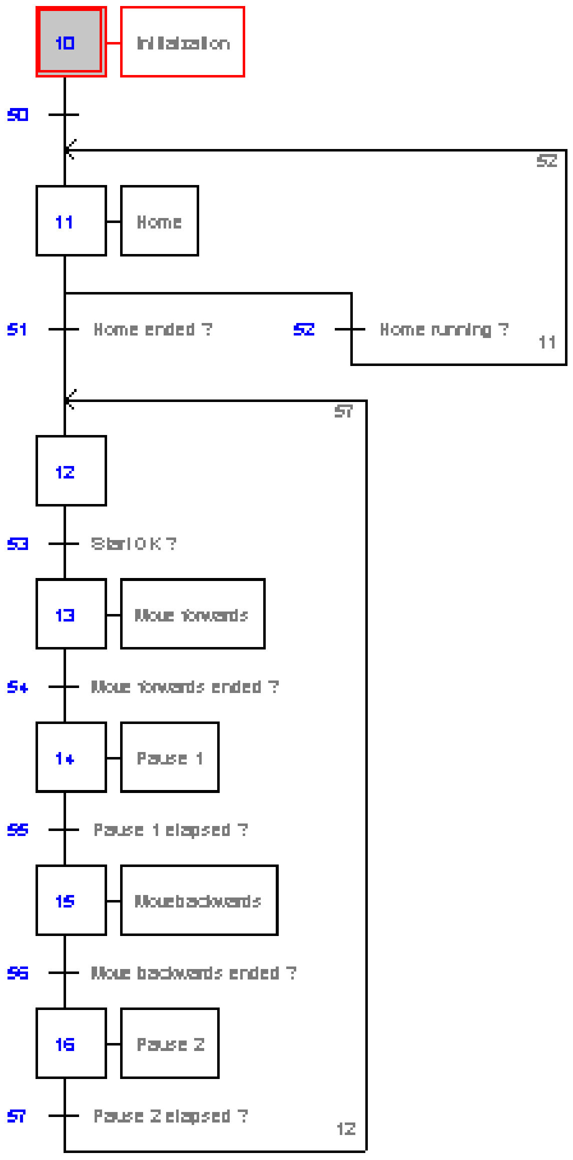
a) Starting position 'normal' (carriage between 'Ref' position and 'LS2'): Behaviour as described above. If the reference point is not found within a defined timeout, homing is halted. If the reference switch is activated on power-up (carriage located on reference switch), free travel only takes place and travel to the next index signal.

b) Starting position with carriage between 'LS1' and 'Ref': The carriage first travels backwards to 'LS1', then forwards to reference switch 'Ref'. The reference switch is travelled over at the velocity of free travel. After this free travel, it again travels on until it reaches the next index signal of the incremental shaft encoder. This means that the home position has been reached and the travel program can start.

c) Starting position located to the right of 'LS2': Homing proceeds as with a). 'LS2' is travelled over and does not affect homing.

d) Starting position located to the left of 'LS1'. The carriage travels backwards as defined until the final limit switch. The home position cannot be found. If the final limit switch is not reached within the defined timeout, homing is halted.

**User program in GRAFTEC** (int-home.sfc)



Initialization occurs in IST. Homing is then executed (ST 1). ST 1 will run through continuously until homing is complete. The end of homing is signalled by the 'fEndHome_x' flag. Before calling the 'Home' FB, limit switches 'LS1' and 'LS2', which are wired to digital PCD inputs, must be read into flags 'fLS1_x' and 'fLS2_x'.

After completion of homing the program goes to TR 53, where the start condition of the actual travel program is awaited.

Consult section 6.1.2 for a description of the travel program.

**Program code for "int-home.sfc"**

(To obtain this representation, rename the file "int-home.sfc" as "int-home.src").

```
  SB     0
;--------------------------------
  IST    10        ;Initialization
         O 50

$include D2H310_B.equ
$group H310

  LD   R 1000
         1.00      ; Mechanical factor
  LD   R 1001
         100000    ; Initial absolute speed
  LD   R 1002
         400000    ; Initial absolute acceleration

  CFB    Init      ; Initialization FB
       K 1         ;  Module number
         250       ;  Proportional factor (regulator)
         0         ;  Integrative factor (regulator)
         60        ;  Derivative factor (regulator)
         4000      ;  Integrative limit value
         5         ;  Derivative term sampling interval
         500       ;  Position tolerance
         0         ;  Behaviour in case of position error
         R 1000    ;  Mechanical factor register
         R 1001    ;  Initial velocity register
         R 1002    ;  Initial acceleration register

  ld   t 0         ; Delay for Enable
         20        ;  2 sec
  EST    ;10
```

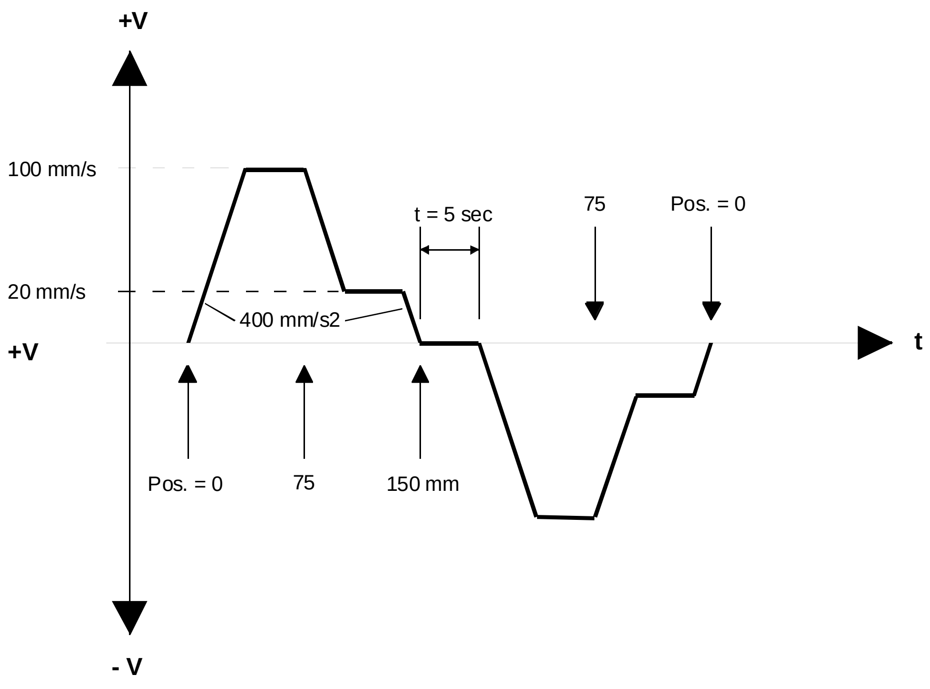
```
;-------------------------------
  ST      11        ;Home
          I 50
          I 52      ;Home running ?
          O 51      ;Home ended ?
          O 52      ;Home running ?

  set   o 71        ; Enable
  res     fEndHome_1

  stl   i 65
  out     fLS1_1

  stl   i 66
  out     fLS2_1

  ld    r 1010
          1000
  ld    r 1011
          10000

  cfb     home
        k 1         ; Module number
          0         ; search direction
          1         ; free run direction
        r 1010      ; min. speed
        r 1011      ; max. speed
          50        ; timeout
        i 64        ; input reference switch
  EST     ;11

;-------------------------------
  ST      12
          I 51      ;Home ended ?
          I 57      ;Pause 2 elapsed ?
          O 53      ;Start OK ?
  EST     ;12

;-------------------------------
  ST      13        ;Move forwards
          I 53      ;Start OK ?
          O 54      ;Move forwards ended ?

  LD    R 100       ; Target Position
          60000     ;  Absolute value

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          LdDestAbs ;  Command: Load Absolute Destination
        R 100       ;  Absolute Destination register

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          StartMot  ;  Command: Start motion
          rNotUsed  ;  Dummy register
  EST     ;13

;-------------------------------
  ST      14        ;Pause 1
          I 54      ;Move forwards ended ?
          O 55      ;Pause 1 elapsed ?

  ld    t 0
          50
  EST     ;14
```

```
;-------------------------------
ST     15          ;Move backwards
       I 55        ;Pause 1 elapsed ?
       O 56        ;Move backwards ended ?

  LD   R 100       ; Target Position
         0         ;  Absolute value

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         LdDestAbs ;  Command: Load Absolute Destination
       R 100       ;  Absolute Destination register

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         StartMot  ;  Command: Start motion
         rNotUsed  ;  Dummy register
  EST    ;15

;-------------------------------
ST     16          ;Pause 2
       I 56        ;Move backwards ended ?
       O 57        ;Pause 2 elapsed ?

  ld   t 0
         50
  EST    ;16

;-------------------------------
TR     50
       I 10        ;Initialization
       O 11        ;Home

  stl  t 0
  ETR    ;50

;-------------------------------
  TR     51        ;Home ended ?
         I 11      ;Home
         O 12

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         RdActPos  ;  Command: Read Actual Position
       R 90        ;  Actual Position register

  DSP  R 90        ; Display register

  sth    fEndHome_1
  ETR    ;51

;-------------------------------
  TR     52        ;Home running ?
         I 11      ;Home
         O 11      ;Home

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         RdActPos  ;  Command: Read Actual Position
       R 90        ;  Actual Position register

  DSP    R 90      ; Display register

  stl    fEndHome_1
  ETR    ;52
```

```
;------------------------------
  TR     53        ;Start OK ?
         I 12
         O 13      ;Move forwards

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP  R 90        ; Display register
  sth  i 0
  ETR    ;53

;------------------------------
  TR     54        ;Move forwards ended ?
         I 13      ;Move forwards
         O 14      ;Pause 1

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP  R 90        ; Display register

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdStatRg  ;  Command: Read Status Register
      R 0          ;  Value of Status Register

  STH    fOnDest_1 ; Position reached?
  ETR    ;54

;------------------------------
  TR     55        ;Pause 1 elapsed ?
         I 14      ;Pause 1
         O 15      ;Move backwards
  stl  t 0
  ETR    ;55

;------------------------------
  TR     56        ;Move backwards ended ?
         I 15      ;Move backwards
         O 16      ;Pause 2

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP  R 90        ; Display register

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdStatRg  ;  Command: Read Status Register
      R 0          ;  Value of Status Register

  STH    fOnDest_1 ; Position reached?
  ETR    ;56

;------------------------------
  TR     57        ;Pause 2 elapsed ?
         I 16      ;Pause 2
         O 12
  stl  t 0
  $endgroup
  ETR    ;57
  ESB    ;0
```

## 12.2 Example: 1 axis with two velocities

This example again concerns the forward and backward motion of a carriage, with the travel velocity being switched from fast to slow during both forward and backward travel. This example is intended above all to show the mechanism of parameter modification during travel. To avoid overloading the program, the 'Home' function has been dispensed with.

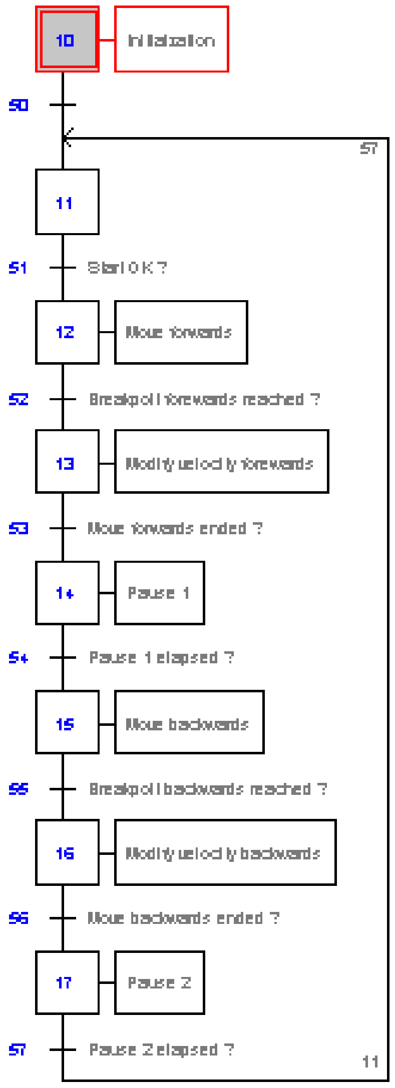


The examples given in sections 6.1.2 ("Entry-level example") and 11.1 ("1 axis with homing") both featured fixed parameters for velocity, acceleration and PID values. All these parameters were defined once in FB 'Init' and left unchanged throughout the course of the program. The only values that are modified are the relative or absolute positions for any carriage travel.

In the present example, velocity is modified during travel in accordance with the diagram. The technique applied here uses a breakpoint. ST 12 and ST 15 not only contain definitions of the next destination position but also of a breakpoint. The position of the breakpoint is the switch-over point for velocity.

In TR 52 and TR 55 arrival at the breakpoint is established by querying the 'fBrkPt_x' flag and then, in ST 13 and ST 16, the new velocity is loaded and activated in each case with a 'Start' FB. The modification of other parameters during travel follows the same pattern.

**User program in GRAFTEC** (2-speed.sfc)



Initialization occurs in the IST.

ST 12 and ST 15 contain entries not only of the next destination position but also the breakpoint for change of speed and the next velocity, all of which is activated with 'Start'.

In TR 52 and TR 55 waiting occurs until the relevant breakpoint is reached for the purpose of user program control. The correct change-over of velocity is performed by the module itself, not the user program.

In ST 13 and ST 16 the new (slow) velocity is defined and activated with 'Start'.

In TR 53 and TR 56 waiting occurs until the relevant destination position is reached for the purpose of user program control. The correct halting of travel is performed by the module itself, not the user program.

In most TRs the actual position is read and output to the display. In practice these parts of the program can be dispensed with or only provided where they seem appropriate.

**Program code for "2-speed.sfc"**

(To obtain this representation, rename the file "int-home.sfc" as "2-speed.src").

```
  SB     0
;------------------------------
  IST    10        ;Initialization
         O 50

$include D2H310_B.equ
$group H310

  LD   R 1000
         1.00      ; Mechanical factor
  LD   R 1001
         100000    ; Initial absolute speed
  LD   R 1002
         400000    ; Initial absolute acceleration

  CFB    Init      ; Initialization FB
       K 1         ;  Module number
         250       ;  Proportional factor (regulator)
         0         ;  Integrative factor (regulator)
         60        ;  Derivative factor (regulator)
         4000      ;  Integrative limit value
         5         ;  Derivative term sampling interval
         500       ;  Position tolerance
         0         ;  Behaviour in case of position error
       R 1000      ;  Mechanical factor register
       R 1001      ;  Initial velocity register
       R 1002      ;  Initial acceleration register

  ld   t 0         ; Delay for Enable
         20        ;  2 sec

  EST    ;10

;------------------------------
  ST     11
         I 50
         I 57      ;Pause 2 elapsed ?
         O 51      ;Start OK ?

  set    o 71      ; Enable

  EST    ;11
```

```
;------------------------------
  ST     12        ;Move forwards
         I 51      ;Start OK ?
         O 52      ;Breakpoint forwards reached ?

  LD   R 100       ; Target Position
         60000     ;  Absolute value

  LD   R 101       ; Breakpoint Position
         30000     ;  Absolute value

  LD   R 102       ; Speed
         100000    ;  Absolute value

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         LdDestAbs ;  Command: Load Absolute Destination
       R 100       ;  Absolute Destination register

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         LdBrkPtAbs ; Command: Load Absolute Breakpoint
       R 101       ;  Absolute Destination register

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         LdVelAbs  ;  Command: Load absolute Velocity
       R 102       ;  Speed

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         StartMot  ;  Command: Start motion
         rNotUsed  ;  Dummy register

  EST    ;12

;------------------------------
  ST     13        ;Modify velocity forwards
         I 52      ;Breakpoint forwards reached ?
         O 53      ;Move forwards ended ?

  LD   R 102       ; Speed
         20000     ;  Absolute value

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         LdVelAbs  ;  Command: Load absolute Velocity
       R 102       ;  Speed

  CFB    Exec      ; Executable FB
       K 1         ;  Module number
         StartMot  ;  Command: Start motion
         rNotUsed  ;  Dummy register

  EST    ;13

;------------------------------
  ST     14        ;Pause 1
         I 53      ;Move forwards ended ?
         O 54      ;Pause 1 elapsed ?

  ld   t 0
         50

  EST    ;14
```

```
;-------------------------------
  ST      15        ;Move backwards
          I 54      ;Pause 1 elapsed ?
          O 55      ;Breakpoint backwards reached ?

  LD    R 100       ; Target Position
          0         ;  Absolute value

  LD    R 101       ; Target Position
          30000     ;  Absolute Breakpoint

  LD    R 102       ; Speed
          100000    ;  Absolute value

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          LdDestAbs ;  Command: Load Absolute Destination
        R 100       ;  Absolute Destination register

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          LdBrkPtAbs ;  Command: Load Absolute Breakpoint
        R 101       ;  Absolute Destination register

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          LdVelAbs  ;  Command: Load absolute Velocity
        R 102       ;  Speed

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          StartMot  ;  Command: Start motion
          rNotUsed  ;  Dummy register

  EST     ;15

;-------------------------------
  ST      16        ;Modify velocity backwards
          I 55      ;Breakpoint backwards reached ?
          O 56      ;Move backwards ended ?

  LD    R 102       ; Speed
          20000     ;  Absolute value

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          LdVelAbs  ;  Command: Load absolute Velocity
        R 102       ;  Speed

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          StartMot  ;  Command: Start motion
          rNotUsed  ;  Dummy register

  EST     ;16

;-------------------------------
  ST      17        ;Pause 2
          I 56      ;Move backwards ended ?
          O 57      ;Pause 2 elapsed ?
  ld    t 0
          50

  EST     ;17
```

```
;--------------------------------
  TR     50
         I 10      ;Initialization
         O 11
  ETR    ;50

;--------------------------------
  TR     51        ;Start OK ?
         I 11
         O 12      ;Move forwards

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP R 90         ; Display register

  sth  i 0

  ETR    ;51

;--------------------------------
  TR     52        ;Breakpoint forwards reached ?
         I 12      ;Move forwards
         O 13      ;Modify velocity forwards

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP R 90         ; Display register

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdStatRg  ;  Command: Read Status Register
      R 0          ;  Value of Status Register

  STH    fBrkPt_1  ; Breakpoint position reached?

  ETR    ;52

;--------------------------------
  TR     53        ;Move forwards ended ?
         I 13      ;Modify velocity forwards
         O 14      ;Pause 1

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdActPos  ;  Command: Read Actual Position
      R 90         ;  Actual Position register
  DSP R 90         ; Display register

  CFB    Exec      ; Executable FB
      K 1          ;  Module number
         RdStatRg  ;  Command: Read Status Register
      R 0          ;  Value of Status Register

  STH    fOnDest_1 ; Position reached?

  ETR    ;53
```

```
;-------------------------------
  TR      54        ;Pause 1 elapsed ?
          I 14      ;Pause 1
          O 15      ;Move backwards

  stl  t 0

  ETR    ;54

;-------------------------------
  TR      55        ;Breakpoint backwards reached ?
          I 15      ;Move backwards
          O 16      ;Modify velocity backwards

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          RdActPos  ;  Command: Read Actual Position
        R 90        ;  Actual Position register
  DSP   R 90        ; Display register

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          RdStatRg  ;  Command: Read Status Register
        R 0         ;  Value of Status Register

  STH     fBrkPt_1  ; Breakpoint position reached?

  ETR    ;55

;-------------------------------
  TR      56        ;Move backwards ended ?
          I 16      ;Modify velocity backwards
          O 17      ;Pause 2

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          RdActPos  ;  Command: Read Actual Position
        R 90        ;  Actual Position register
  DSP   R 90        ; Display register

  CFB     Exec      ; Executable FB
        K 1         ;  Module number
          RdStatRg  ;  Command: Read Status Register
        R 0         ;  Value of Status Register

  STH     fOnDest_1 ; Position reached?

  ETR    ;56

;-------------------------------
  TR      57        ;Pause 2 elapsed ?
          I 17      ;Pause 2
          O 11

  stl     t 0

$endgroup

  ETR    ;57

  ESB    ;0
```

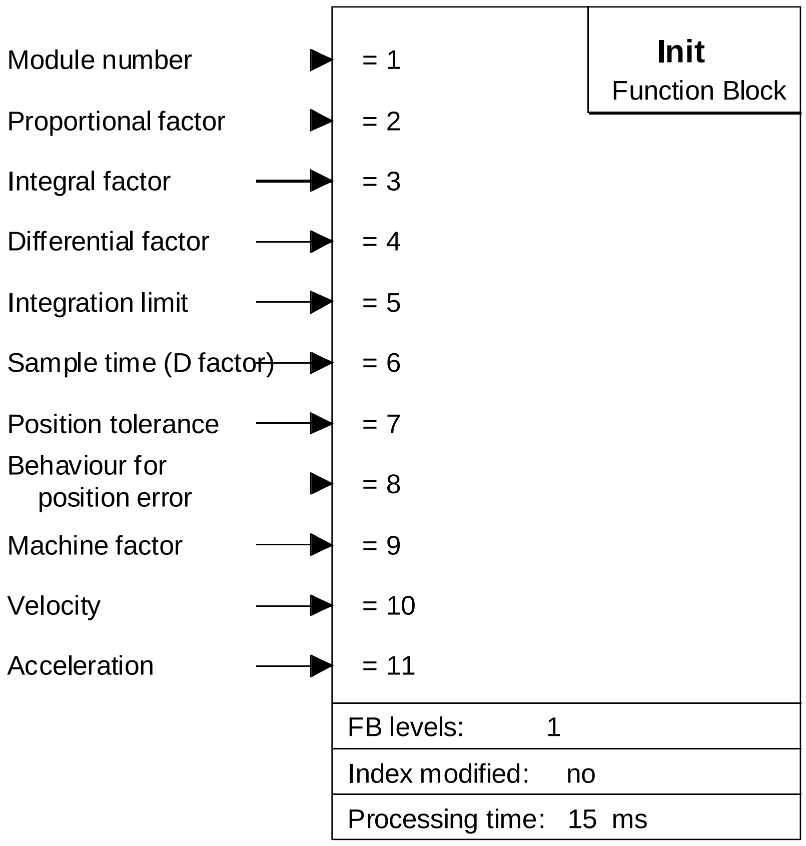
Notes

# Appendix A: Summary of all software elements for programming in IL

## Function block 'Init'

### Init

**FB:** Initialization of an H31x module



**Function description:**

This FB defines the settings of a PCD2.H31x module and reads the module base address from the D2H310_B.MBA file.

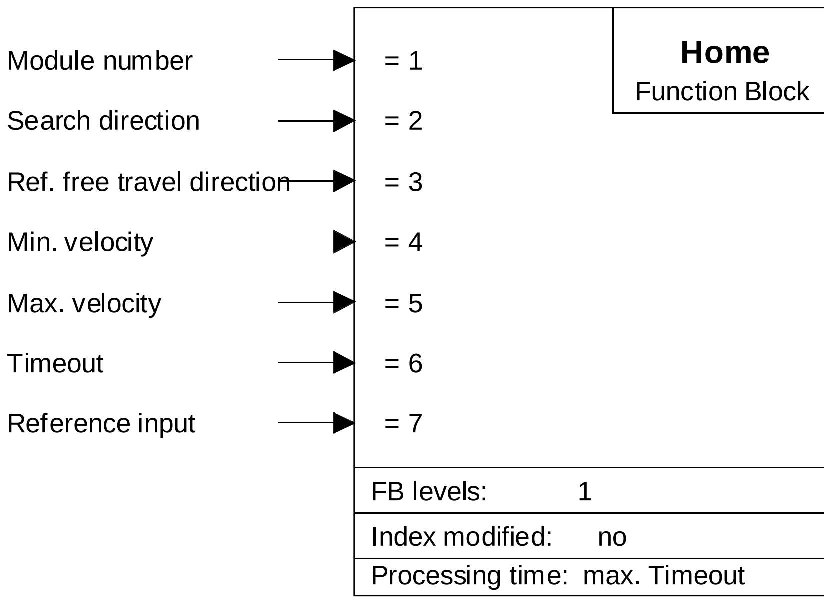
Parameter '1' should be specified as a constant 'K', parameters '9' to '11' as PCD register addresses (absolute or symbolic) and all other parameters should be specified as integer values.

Parameter '9' is the machine factor. This contains the encoder parameters and the mechanical parameters of the system. (Must be entered in floating-point format, e.g. for '4' as 4.0 or 4E1).

| Para-meter | Meaning | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number. | K | K n | K 1 – K 16 | |
| = 2 | Proportional factor | | Integer | 0 – 32767 | P factor of PID controller. |
| = 3 | Integral factor | | Integer | 0 – 32767 | I factor of PID controller. |
| = 4 | Differential factor | | Integer | 0 – 32767 | D factor of PID controller. |
| = 5 | Integration limit | | Integer | 0 – 32767 | Limit on influence of I portion in system. |
| = 6 | Sample time of D factor | | Integer | 0 – 255 | Sample interval for D factor |
| = 7 | Position tolerance | | Integer | 0 – 32767 | Maximum position error, compared with calculated value. |
| = 8 | Behaviour for position error | | Integer | 0 - 1 | Behaviour for exceeding position tolerance<br>0 = Error flag = H<br>1 = Stop motion |
| = 9 | Machine factor | R | Floating | 0..9.223371*$10^{18}$ | Factor containing parameters for the encoder and the system. |
| = 10 | Velocity | R | Integer | -- | Initial velocity, defined by machine factor. |
| = 11 | Acceleration | R | Integer | -- | Initial acceleration, defined by machine factor. |

# Function block 'Home'

## Home **FB:** Initialization of home position



**Function description:**

This FB defines the homing settings for the PCD2.H31x module.

Parameter 1 should be entered as a constant 'K', parameter 7 as a 'I' input, and all other parameters should be entered as integers.

The PCD2.H310 module has a 'Ref' input. The address of this input is the module base address + 11. For an H310 module at base address 64, the 'Ref' input is located at address I 75.

The PCD2.H311 module has no 'Ref' input. Any PCD input can be chosen for this.

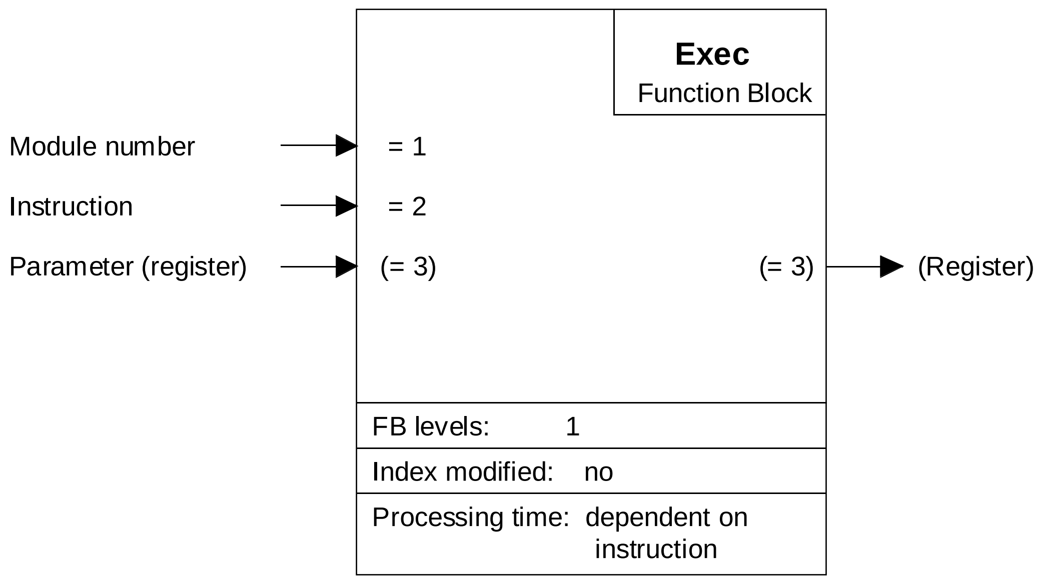
**The 'fEndHome' flag should be reset before any homing,** otherwise homing cannot start. This flag is automatically set high when homing is complete.

| **Para-meter** | **Meaning** | **Type** | **Format** | **Value** | **Comment** |
|---|---|---|---|---|---|
| = 1 | Module number | K | K n | K 1 – K 16 | |
| = 2 | Search direction | | Integer | 0 - 1 | Defines the direction in which to find the reference switch:<br>0 = backwards<br>1 = forwards |
| = 3 | Free travel direction from reference switch | | Integer | 0 - 1 | Defines the free travel direction, away from the reference switch:<br>0 = backwards<br>1 = forwards |
| = 4 | Minimum velocity | R | Integer | -- | Velocity for leaving the reference switch. |
| = 5 | Maximum velocity | R | Integer | -- | Velocity for seeking the reference switch. |
| = 6 | Timeout | | Integer | 0 – 65535 [s] | Time until homing will be halted. |
| = 7 | Reference input | I | Integer | -- | Input to which the refer-ence switch is connected. |

## Function block 'Exec'

### Exec

**FB:** Execution of an instruction for the H31x module



**Function description:**

This FB is used to send execution instructions to the PCD2.H31x module.

The module number (parameter 1) must be a constant (k 1…k 16). The base address is defined in the 'D2H310_B.MBA' file. The FBs support max. 16 PCD2.H31x modules per PCD system.

Individual instructions (parameter 2) are dealt with on the following pages.

The parameters of an instruction (e.g. the acceleration value in the instruction LdAccAbs) are transferred in a register (parameter 3). If an instruction requires no parameters (e.g. Start) any register can be transferred, or 'rNotUsed'.

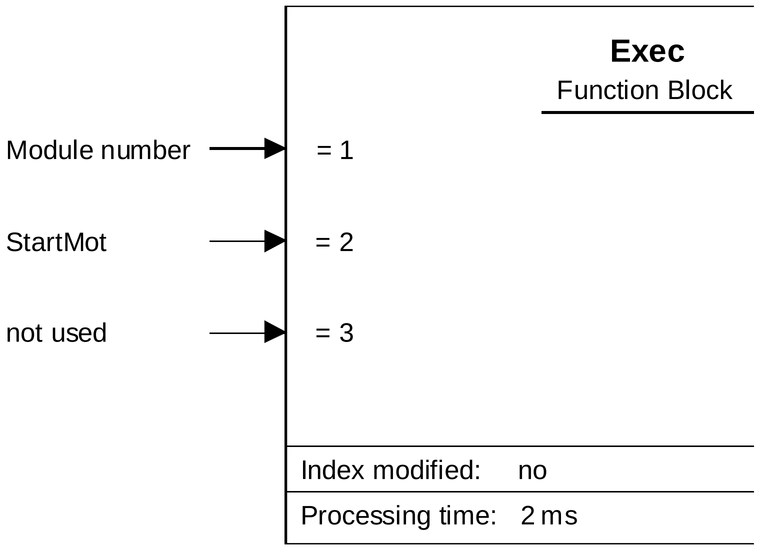
## Individual instructions for the PCD2.H310 (FB parameters)

| No. | Symbol | Instruction | Page |
|---|---|---|---|
| 01 | StartMot | Start Motion | A-7 |
| 02 | StopUrg | Stop motion in Urgency | A-8 |
| 03 | Stop | Stop motion | A-9 |
| 04 | MotOff | Motor regulation Off | A-10 |
| 05 | RdActPos | Read Actual Position | A-11 |
| 06 | RdActVel | Read Actual Velocity | A-12 |
| 07 | RdIntSum | Read Integration Sum | A-13 |
| 08 | RdIndexRg | Read Index Register | A-14 |
| 09 | RdStatRg | Read Status Register | A-15 |
| 10 | RdTargPos | Read Target Position | A-16 |
| 11 | RdTargVel | Read Target Velocity | A-17 |
| 12 | GoForw | Go Forwards | A-18 |
| 13 | GoBackw | Go Backwards | A-19 |
| 14 | SgStpFor | Single Step Forwards | A-20 |
| 15 | SgStpBak | Single Step Backwards | A-21 |
| 16 | LdDestAbs | Load Destination Absolute | A-22 |
| 17 | LdDestRel | Load Destination Relative | A-23 |
| 18 | LdVelAbs | Load Velocity Absolute | A-24 |
| 19 | LdVelRel | Load Velocity Relative | A-25 |
| 20 | LdAccAbs | Load Acceleration Absolute | A-26 |
| 21 | LdAccRel | Load Acceleration Relative | A-27 |
| 22 | LdPropG | Load Proportional Gain | A-28 |
| 23 | LdIntG | Load Integrative Gain | A-29 |
| 24 | LdDerG | Load Derivative Gain | A-30 |
| 25 | LdSampInt | Load derivative Sampling Interval | A-31 |
| 26 | LdIntLim | Load Integrative Limit | A-32 |
| 27 | ActRegFact | Activate Regulation Factors | A-33 |
| 28 | LdBrkPtAbs | Load Breakpoint Absolute | A-34 |
| 29 | LdBrkPtRel | Load Breakpoint Relative | A-35 |
| 30 | ResStatRg | Reset Status Register | A-36 |
| 31 | SetIdxPos | Set Index Position | A-37 |
| 32 | SetZero | Set Zero position | A-38 |
| 33 | MotConf | Motion Configuration . | A-39 |
| 34 | SetPosTol | Set Position Tolerance | A-40 |

The number in the first column (0 - 34) is the absolute value of parameter no. 2 in FB 'Exec'. When the user program is being traced in the debugger, this number can help to interpret the function of FB 'Exec'.

## StartMot **Instruction:** Start Motion

[01]



**Function description:**

This instruction is used to start a motion with the defined velocity and acceleration to the predefined destination position. It is also used to select 'position control mode'.

This instruction switches back 2 bits in the status byte:

bit 6 : Breakpoint reached
bit 5 : Excessive position error

The instruction 'ResStatRg' enables all or some of the status bits to be reset. The status byte is represented in the first 8 bits of the signal register. These 8 bits are reset simultaneously with the status byte.

The 'StartMot' instruction also resets bits 10 to 15 of the signal register, which can be read with the instruction 'RdStatRg'.

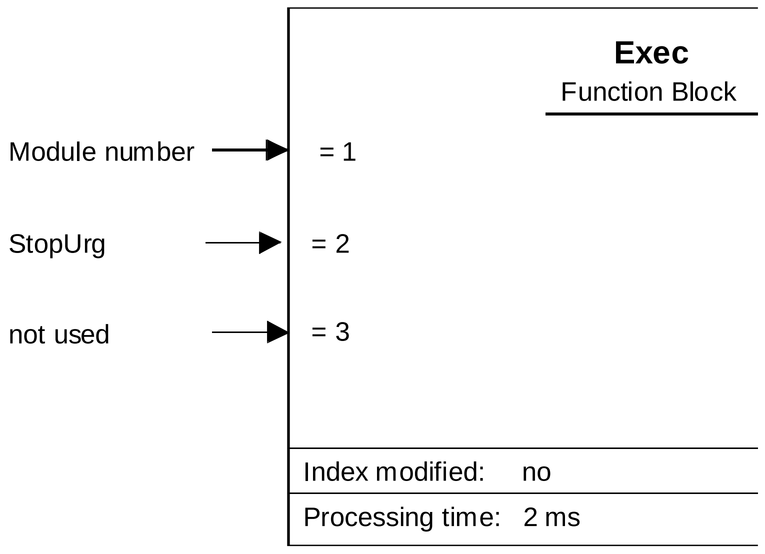
The flage 'fOnDest_x', 'fBrkPt_x' and 'fPosErr_x' are only refreshed after a 'StartMot' instruction forllowed by a 'RdStatRg' instruction.

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: StartMot | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## StopUrg

**Instruction:** Stop motion in Urgency

[02]



**Function description:**

This instruction is used to stop motion abruptly (without braking ramp).

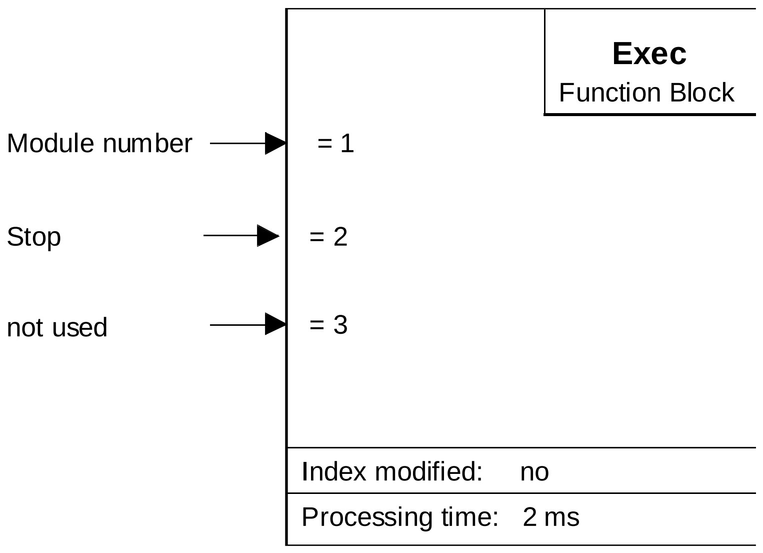
The instruction can be used both in position control and speed control modes.

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: StopUrg | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## Stop

**Instruction:** Stop motion

[03]



**Function description:**

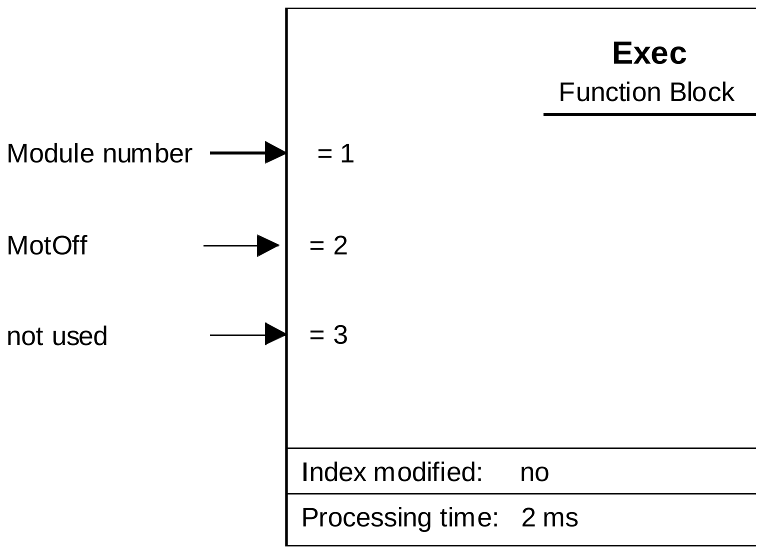
This instruction can be used to stop motion, using the normal braking ramp.

The instruction can be used both in position control and speed control modes.

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: Stop | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## MotOff **Instruction:** Motor regulation is put Off

[04]



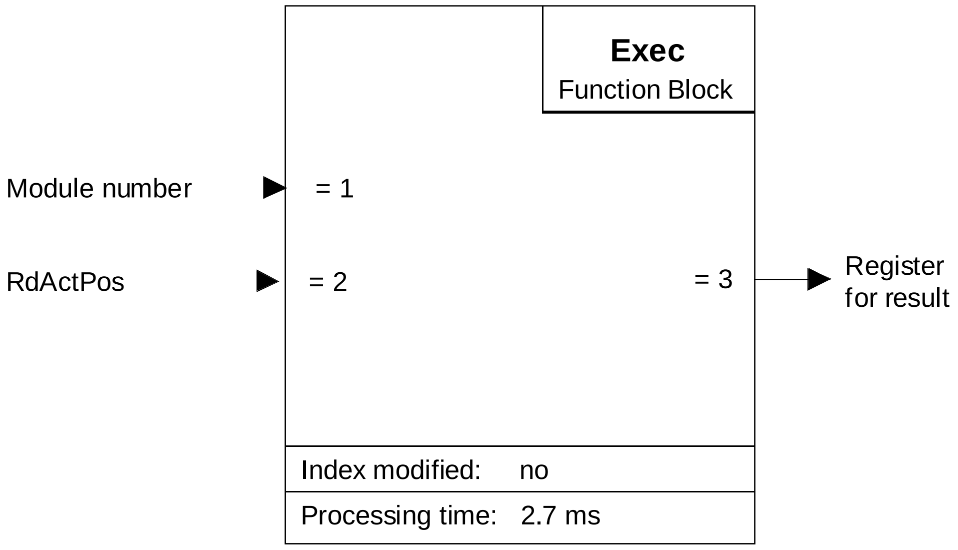
**Function description:**

This instruction is used to switch off the PID function that influences the motor. The effect is the same as if the motor supply were removed.

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: MotOff | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## RdActPos **Instruction:** Read Actual Position

[05]



**Function description:**

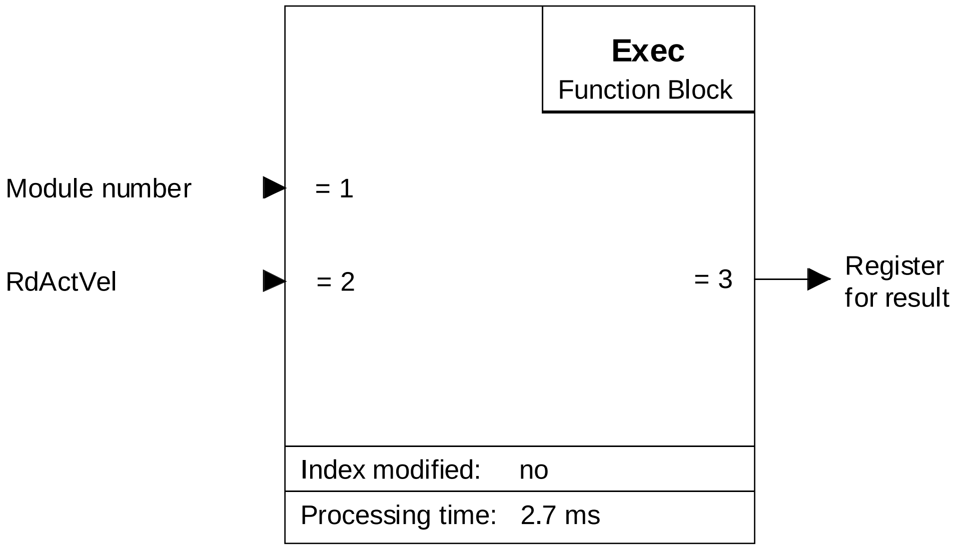
This instruction is used to read the actual position of the system. It should be called whenever position has to be read.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdActPos | | | | |
| = 3 | PCD register for the actual Position | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |

## RdActVel **Instruction:** Read Actual Velocity

[06]



**Function description:**

This instruction can be use to read the actual velocity of motion. For each new reading of the velocity this instruction shuld be executed.

The processor format for velocity is 14 bits integer and 16 bits fractional number. With the present instruction the processor only outputs the integer part. This means that, at low velocities, values obtained are not comparable with the result of the 'RdTargVel' instruction, which contains the fractional part.

If the machine factor is less than 1, low velocities are difficult to read, as the fractional part of the velocity is not available.

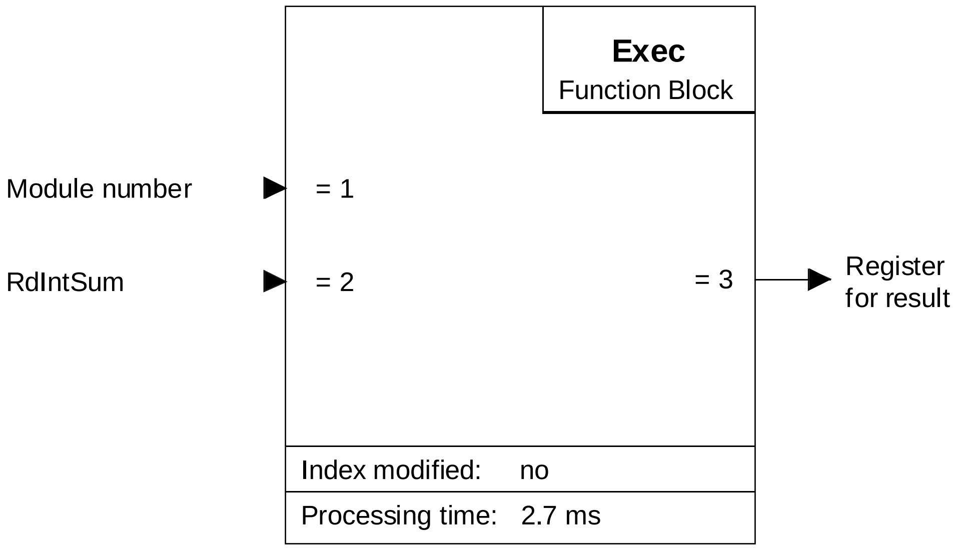
Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdActVel | | | | |
| = 3 | PCD register for actual velocity | R | integer | 30 bit *) | dependent on machine factor |

*) Resolution 14 bit

## RdIntSum **Instruction:** Read Integration Sum

[07]



**Function description:**

This instruction can be used to read the integration sum of the PID controller's I portion.

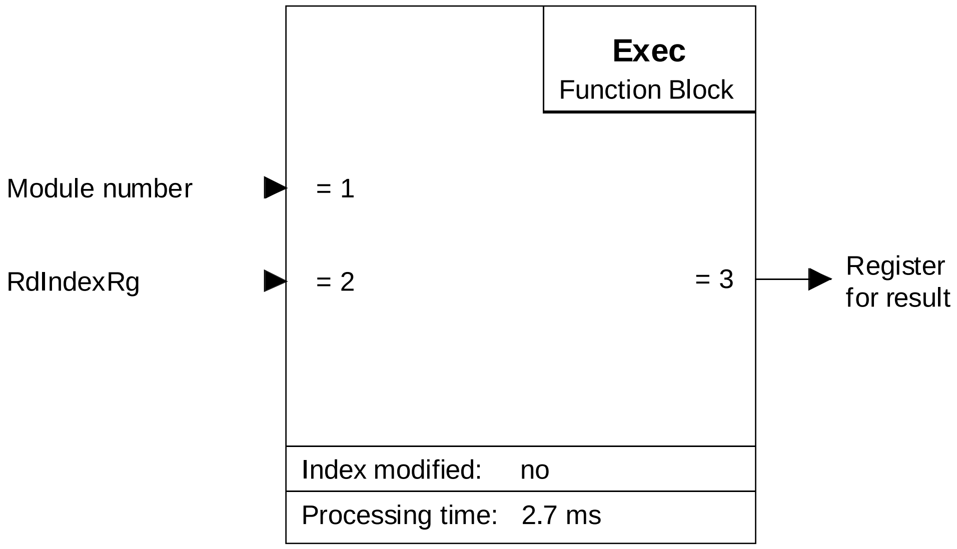
The instruction should be called whenever the integration sum has to be read.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdIntSum | | | | |
| = 3 | PCD register for the integration sum | R | integer | 15 bit | 0 .. 32767 |

## RdIndexRg **Instruction:** Read Index Register

[08]



**Function description:**

This instruction is used to read the index register (encoder revolution counter) of the H310 module. (Not to be confused with the index register of the PCD's CPU). If a 'SetIdxPos' (Set Index Position) instruction has been executed, absolute position will be recorded at the next encoder index impulse.

Reading the index register can be part of the error-checking program routine. The new index position minus the old one, divided by the encoder resolution (encoder division by 4), must always produce an integer value.

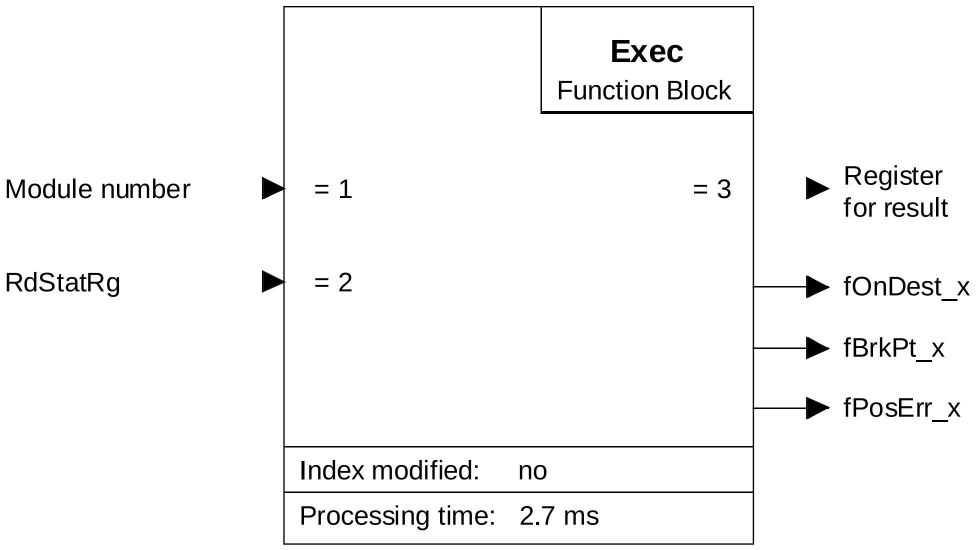
The instruction should be called whenever the index register has to be read.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdIndexRg | | | | |
| = 3 | PCD register for value of index register | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |

## RdStatRg **Instruction:** Read Status Register

[09]



**Function description:**

This instruction can be used to read the status register. It must be executed **before** the flags 'fOnDest_x', 'fPosErr_x' and 'fBrkPt_x' can be evaluated.

Reading the status register allows access to the following parameters:

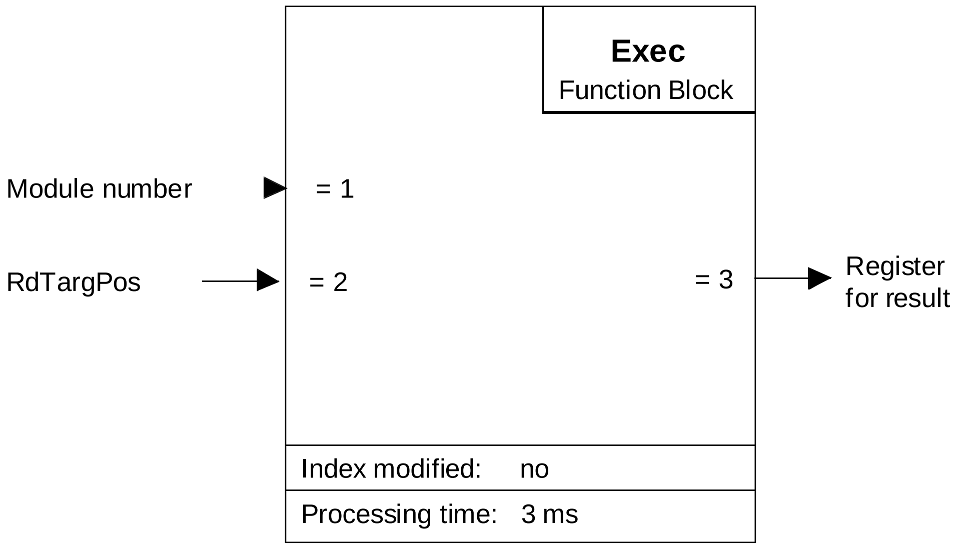
bit 15: Host Interrupt
bit 14: Acceleration loaded
bit 13: Regulator parameters updated
bit 12: Forward direction
bit 11: Velocity mode
bit 10: On Target
bit 9: Turn off upon excessive position error
bit 8: Internal use only
bit 7: Motor off
bit 6: Breakpoint reached
bit 5: Excessive position error
bit 4: Wraparound occurred
bit 3: Index pulse acquired
bit 2: Trajectory complete
bit 1: Internal use only
bit 0: Acquire next index pulse

The instruction should be called whenever the status register has to be read.

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdStatRg | | | | |
| = 3 | PCD register for value of status register | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |
| fOnDest_x | Flag on destination | F | | | |
| fBrkPt_x | Flag breakpoint reached | F | | | |
| fPosErr_x | Flag excessive position error | F | | | |

## RdTargPos **Instruction:** Read Target Position

[10]

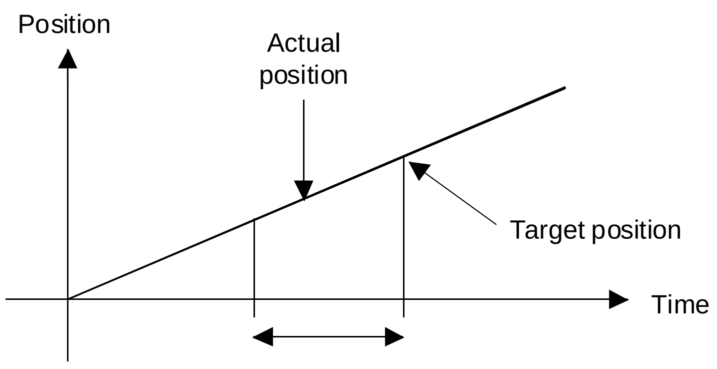


**Function description:**

This instruction can be used to read the current target position of the actual movement.

Description of input and output elements concerned:

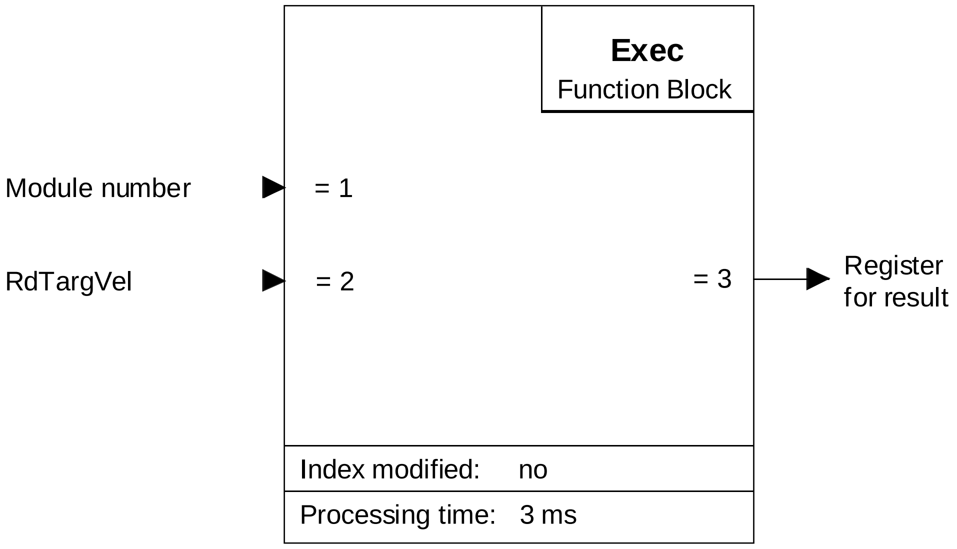
| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K |  | 1 - 16 |  |
| = 2 | Instruction: RdTargPos |  |  |  |  |
| = 3 | PCD register for target position | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |



This refers to the calculated target position in units of the sample time (341 µs).

## RdTargVel **Instruction:** Read Target Velocity

[11]



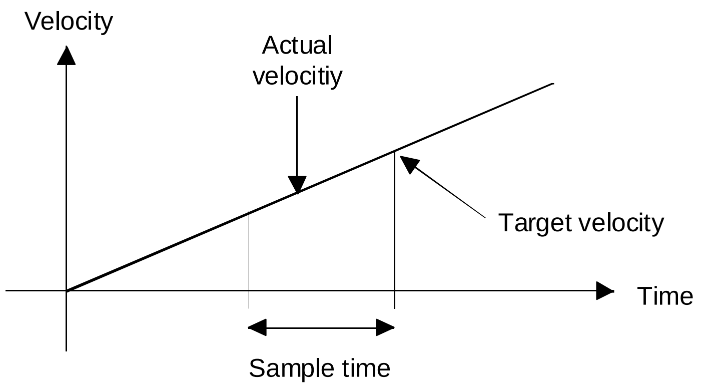
**Function description:**

This instruction can be used to read the target velocity of the system. The reading corresponds to the value that must be reached in the next sample cycle.

The instruction should be called whenever the target velocity has to be read.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: RdTargVel | | | | |
| = 3 | PCD register for the target velocity | R | integer | 30 bit | dependent on machine factor |

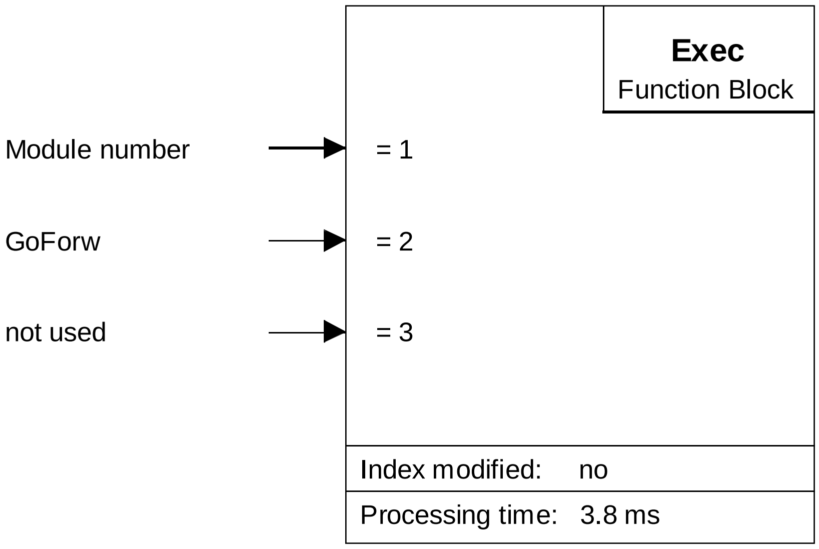


This refers to the calculated target velocity in units of the sample time (341 µs).

## GoForw

**Instruction:** Go Forwards

[12]



**Function description:**

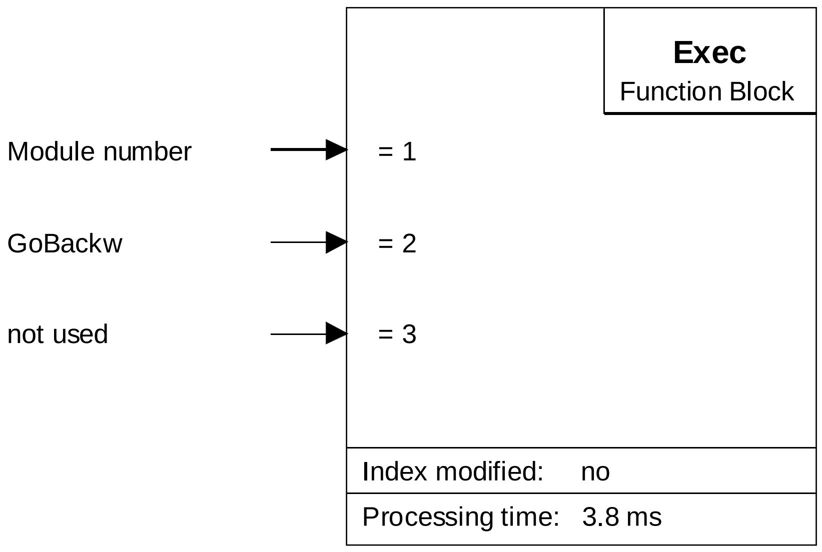
This instruction is used to start forward motion at the predefined velocity and acceleration, but without destination position. With this instruction 'speed control mode' is also selected.

Description of input and output elements concerned:

| **Par.** | **Designation/function** | **Type** | **Format** | **Value** | **Comment** |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: GoForw | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## GoBackw **Instruction:** Go Backwards

[13]



**Function description:**

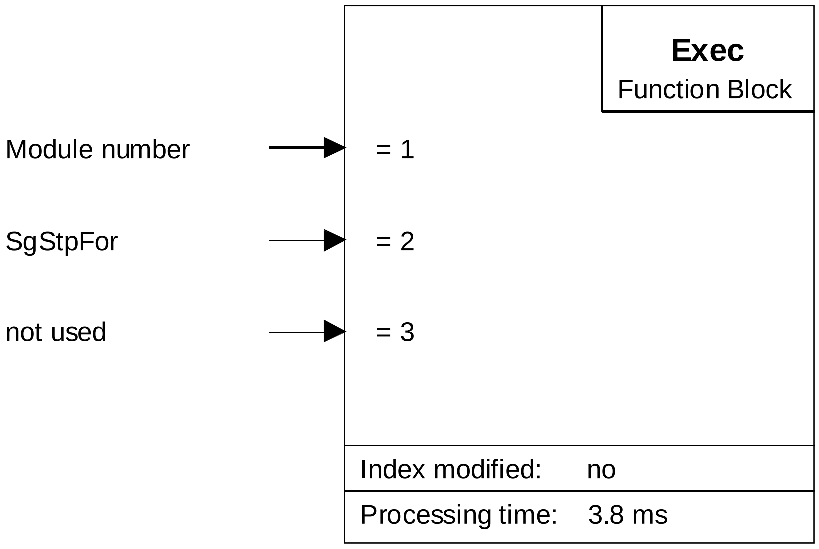
This instruction is used to start backward motion at the predefined velocity and acceleration, but without destination position. With this instruction 'speed control mode' is also selected.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: GoBackw | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## SgStpFor **Instruction:** Single Step Forwards

[14]



**Function description:**

This instruction can be used to switch forward a single step of the encoder resolution at the predefined velocity and acceleration

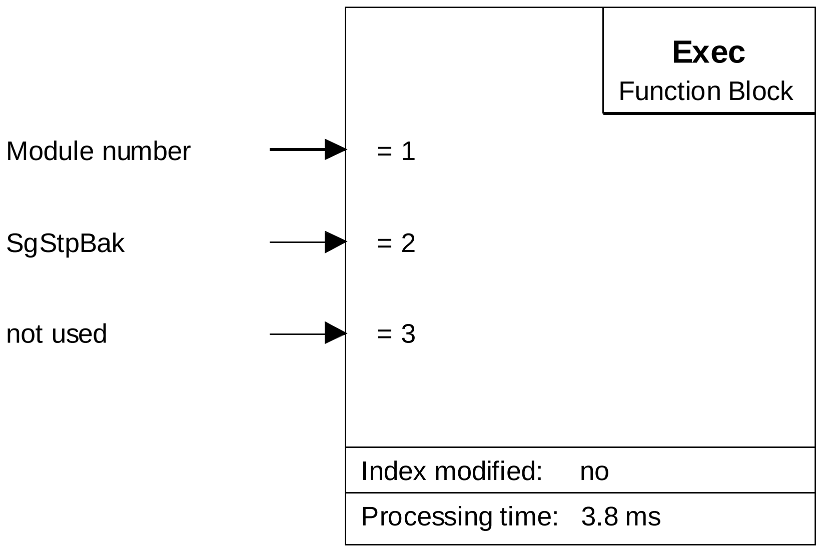
Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: SgStpFor | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## SgStpBak

**Instruction:** Single Step Backwards

[15]



**Function description:**

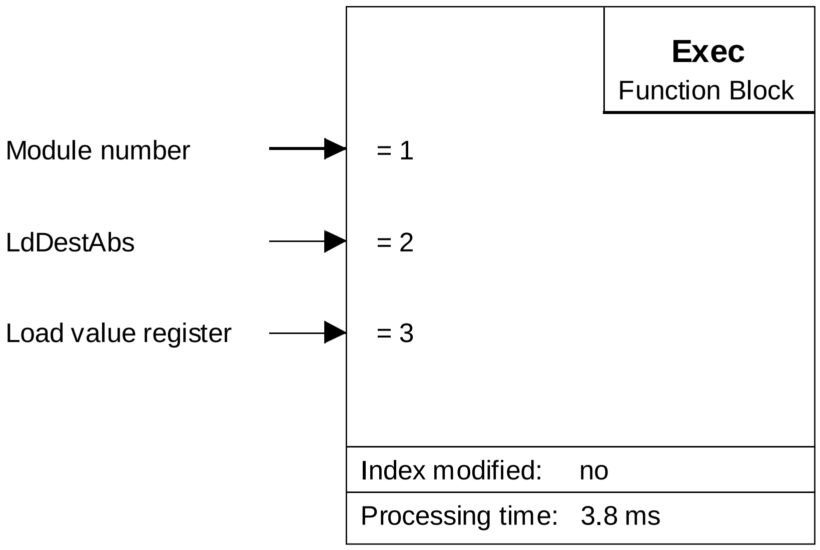
This instruction can be used to switch backward a single step of the en-coder resolution at the predefined velocity and acceleration.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: SgStpBak | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## LdDestAbs **Instruction:** Load Destination Absolute

[16]



**Function description:**

This instruction is used to load the absolute target position. A 'StartMot' instruction must follow to activate the function.

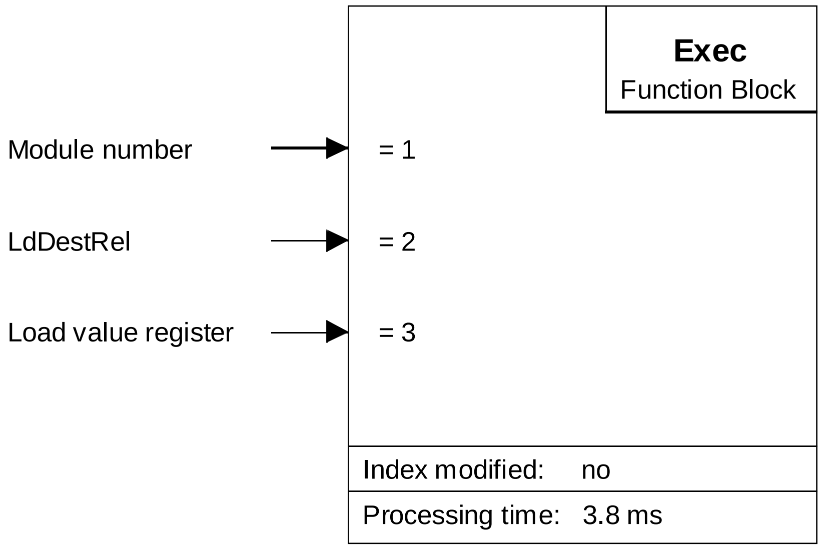
The absolute target position always refers to the zero position.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdDestAbs | | | | |
| = 3 | PCD register with the load value | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |
| fOnDest_x | Flag on destination | F | | | |

## LdDestRel **Instruction:** Load Destination Relative

[17]



**Function description:**

This instruction is used to load the relative target position. A 'StartMot' instruction must follow to activate the function.

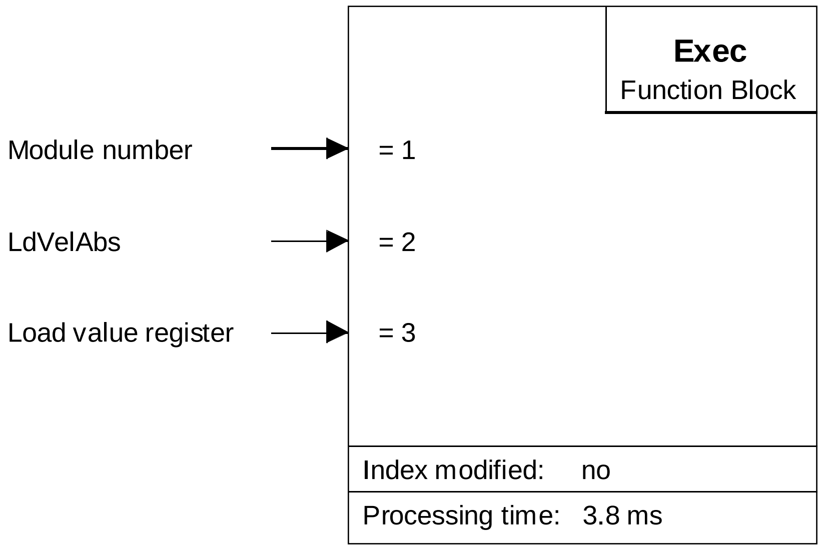
The relative target position always refers to the actual position.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdDestRel | | | | |
| = 3 | PCD register with the load value | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |
| fOnDest_x | Flag on destination | F | | | |

## LdVelAbs **Instruction:** Load Velocity Absolute

[18]



**Function description:**

This instruction is used to load or modify the absolute velocity. A 'Start-Mot' instruction must follow to activate the function.

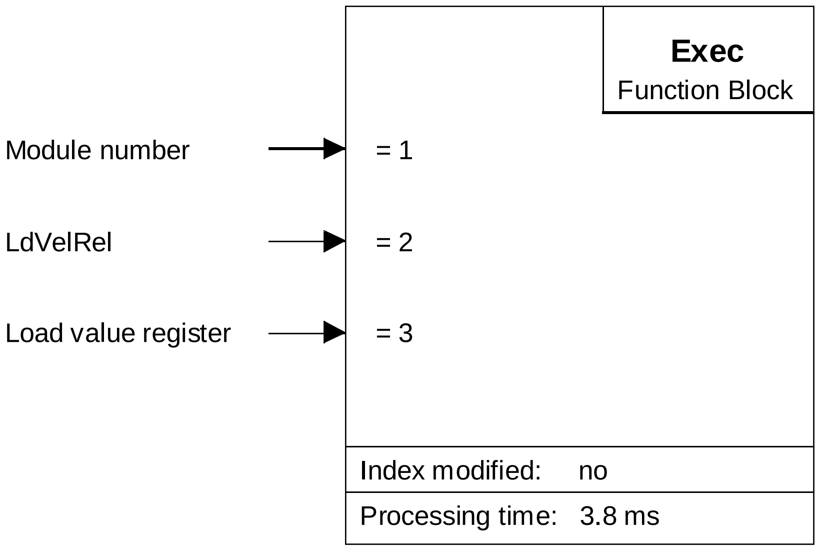
The absolute velocity always refers to the zero velocity.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdVelAbs | | | | |
| = 3 | PCD register with the load value | R | integer | 30 bit | dependent on machine factor |

## LdVelRel **Instruction:** Load Velocity Relative

[19]



**Function description:**

This instruction is used to load or modify the relative velocity. A 'Start-Mot' instruction must follow to activate the function.

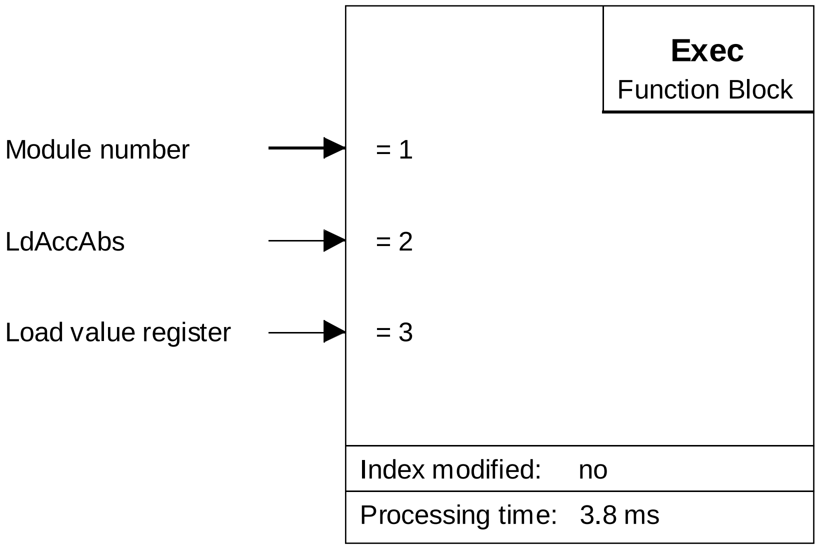
The relative velocity always refers to the actual velocity.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdVelRel | | | | |
| = 3 | PCD register with the load value | R | integer | 30 bit | dependent on machine factor |

## LdAccAbs **Instruction:** Load Acceleration Absolute

[20]



**Function description:**

This instruction is used to load or modify the absolute acceleration. A 'StartMot' instruction must follow to activate the function.

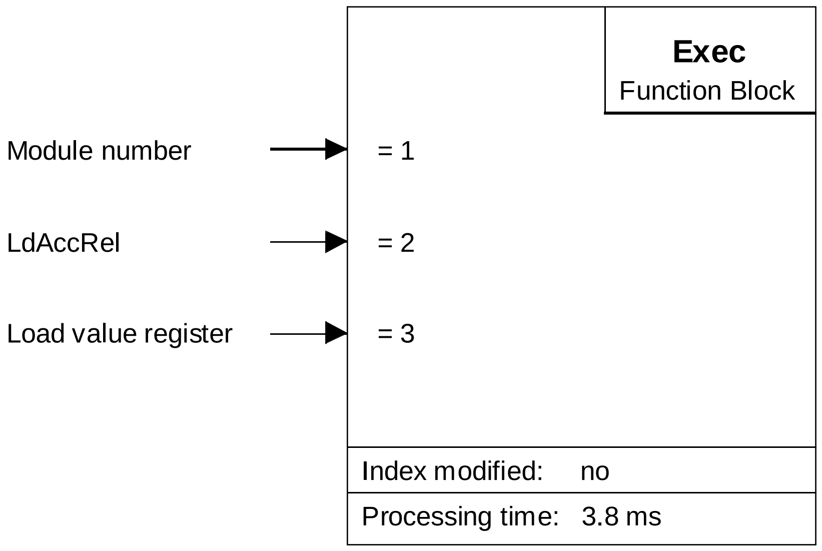
The absolute acceleration always refers to the zero velocity or acceleration.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdAccAbs | | | | |
| = 3 | PCD register with the load value | R | integer | 30 bit | dependent on machine factor |

## LdAccRel **Instruction:** Load Acceleration Relative

[21]



**Function description:**

This instruction is used to load or modify the relative acceleration. A 'StartMot' instruction must follow to activate the function.

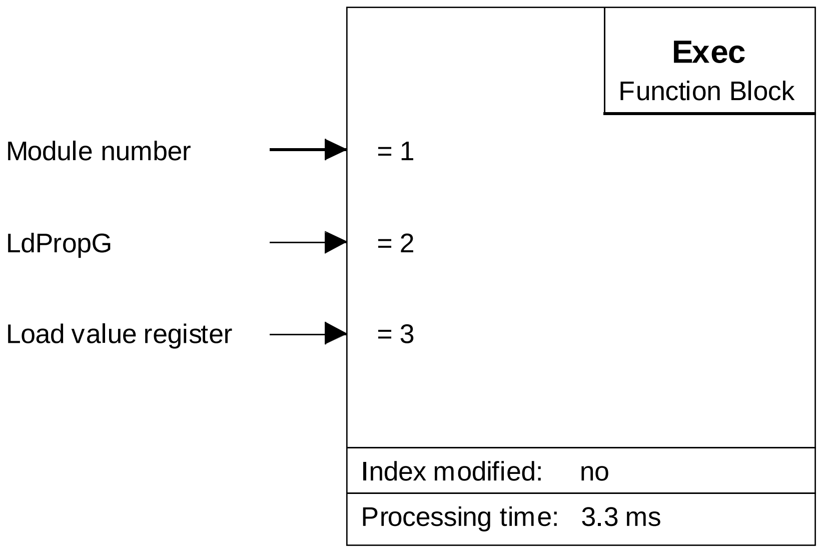
The relative acceleration always refers to the actual acceleration.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdAccRel | | | | |
| = 3 | PCD register with the load value | R | integer | 30 bit | dependent on machine factor |

## LdPropG **Instruction:** Load Proportional Gain

[22]



**Function description:**

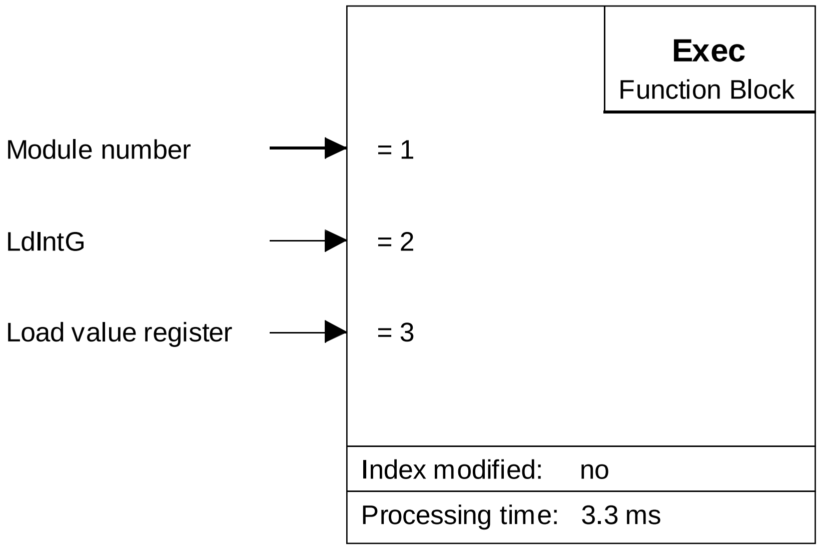
This instruction can be used to load or modify the proportional factor. An 'ActRegFact' instruction must follow to activate the function.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdPropG | | | | |
| = 3 | PCD register with the load value | R | integer | 15 bit | 0 .. 32767 |

## LdIntG **Instruction:** Load Integrative Gain

[23]



**Function description:**

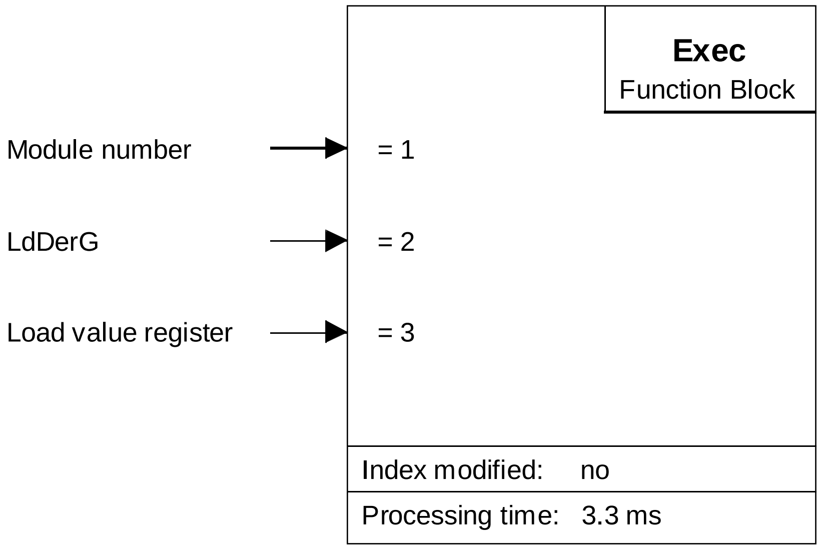
This instruction can be used to load or modify the integral factor. An 'ActRegFact' instruction must follow to activate the function.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdIntG | | | | |
| = 3 | PCD register with the load value | R | integer | 15 bit | 0 .. 32767 |

## LdDerG **Instruction:** Load Derivative Gain

[24]



**Function description:**

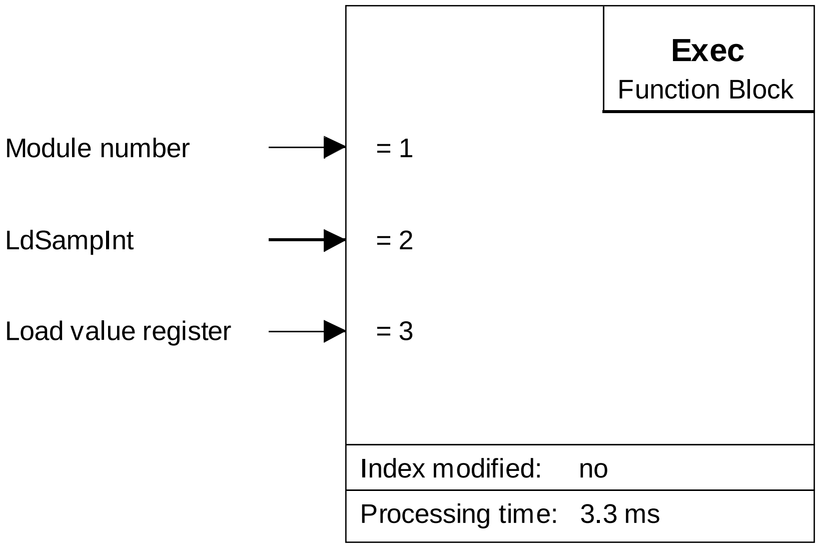
This instruction can be used to load or modify the differential factor. An 'ActRegFact' instruction must follow to activate the function.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdDerG | | | | |
| = 3 | PCD register with the load value | R | integer | 15 bit | 0 .. 32767 |

## LdSampInt **Instruction:** Load Sampling Interval

[25]



**Function description:**

This instruction can be used to load or modify the PID controller’s sample time. An 'ActRegFact' instruction must follow to activate the function.

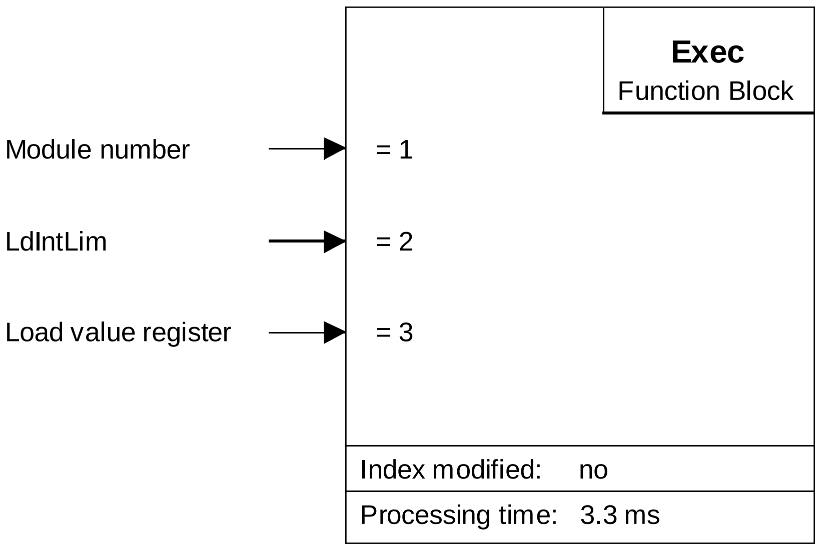
The original sample time (341 µs), which is used as factors 'P' and 'I', is normally too small for the D factor. The differential factor works with the difference of error signal between two sampling signals. For this reason, 341 µs is too small to have a real difference. Experience shows that the sample time generally lies between 2 and 9 ms, corresponding to values of 5 to 25.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdSampInt | | | | |
| = 3 | PCD register with the load value | R | integer | 8 bit | 0 .. 255 |

## LdIntLim **Instruction:** Load Integrative Limit

[26]



**Function description:**

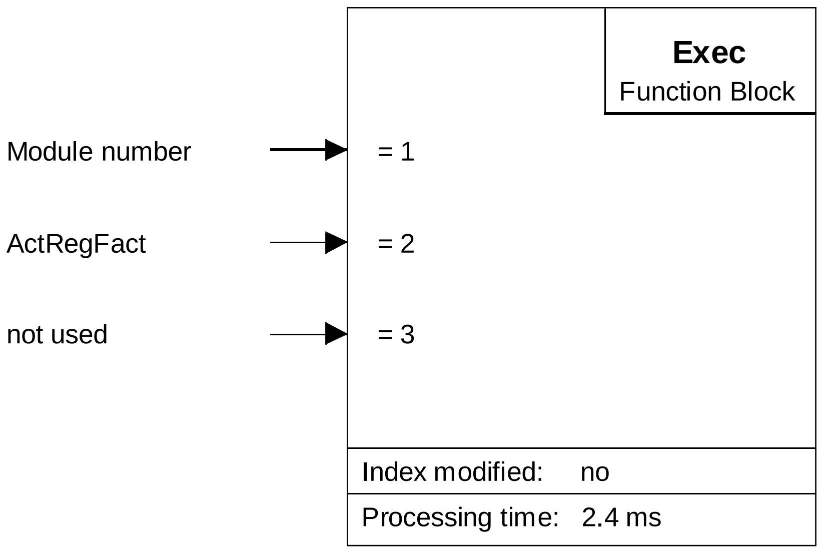
This instruction can be used to load or modify the integration limit. An 'ActRegFact' instruction must follow to activate the function.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdIntLim | | | | |
| = 3 | PCD register with the load value | R | integer | 15 bit | 0 .. 32767 |

## ActRegFact **Instruction:** Activate Regulation Factors

[27]



**Function description:**

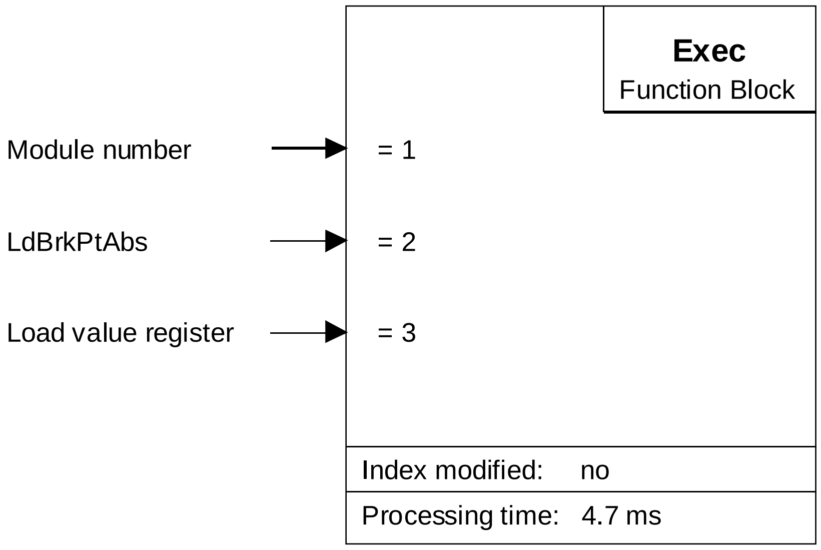
This instruction can be used to activate simultaneously all factors of PID control. It is also possible to activate one parameter alone.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: ActRegFact | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## LdBrkPtAbs **Instruction:** Load Break Point Absolute

[28]



**Function description:**

This instruction can be used to load an absolute breakpoint (signalling an intermediate position). The absolute position always refers to the zero position.

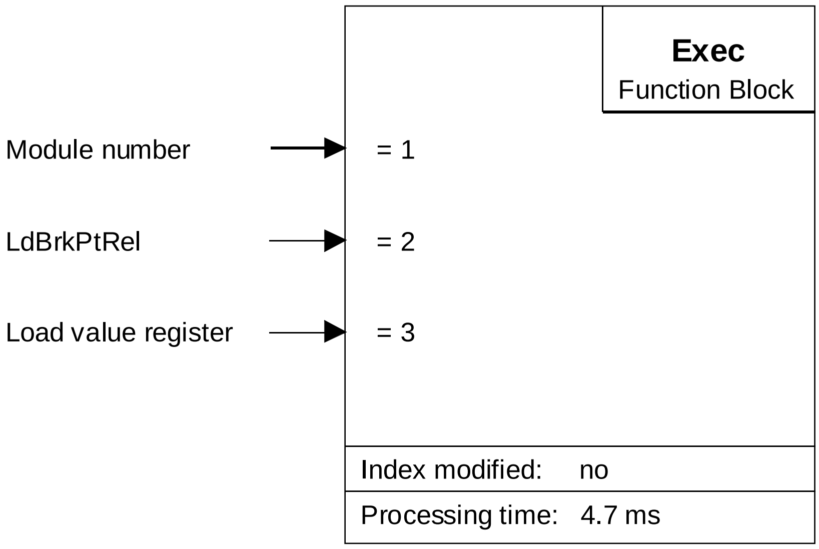
If the breakpoint is reached, the 'fBrkPt_x' flag is set. However, this flag only becomes active after FB 'RdStatRg' has been processed.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdBrkPtAbs | | | | |
| = 3 | PCD register with the load value | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |
| fBrkPt_x | Flag 'Breakpoint' | F | | | This flag is reset with instructions 'StartMot' or 'ResStatRg' . |

## LdBrkPtRel **Instruction:** Load Break Point Relative

[29]



**Function description:**

This instruction can be used to load a relative breakpoint (signalling an intermediate position). The relative position always refers to the actual position.

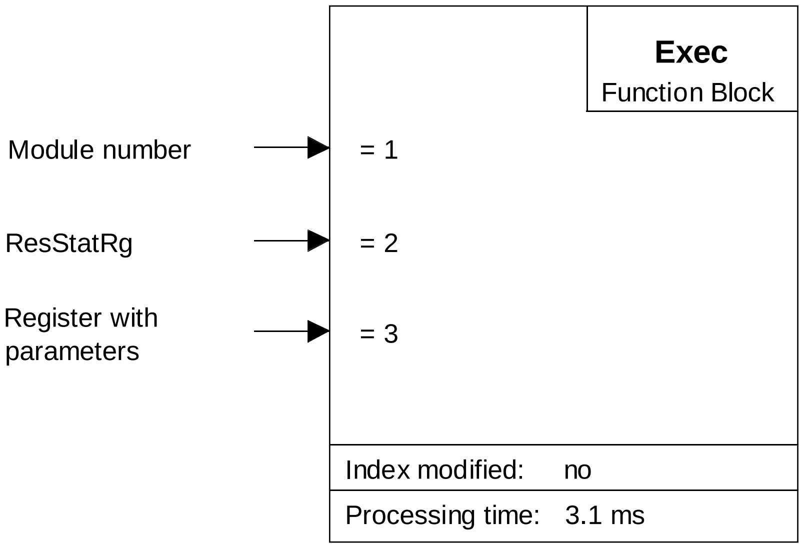
If the breakpoint is reached, the 'fBrkPt_x' flag is set. However, this flag only becomes active after FB 'RdStatRg' has been processed.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: LdBrkPtRel | | | | |
| = 3 | PCD register with the load value | R | integer | 31 bit | $-2^{30}$ .. $+(2^{30} - 1)$ |
| fBrkPt_x | Flag 'Breakpoint' | F | | | This flag is reset with instructions 'StartMot' or 'ResStatRg'. |

## ResStatRg **Instruction:** Reset Status Register

[30]



**Function description:**

This instruction can be used to reset all or any parameters contained in the status register.

**Status register:**

| | |
|---|---|
| bit 7: Motor off | 128 |
| bit 6: Breakpoint reached | 64 |
| bit 5: Excessive position error | 32 |
| bit 4: Wraparound occurred | 16 |
| bit 3: Index pulse observed | 8 |
| bit 2: Trajectory complete | 4 |
| bit 1: Internal use only | 2 |
| bit 0: Internal use only | 1 |
| | ---- |
| | 255 |

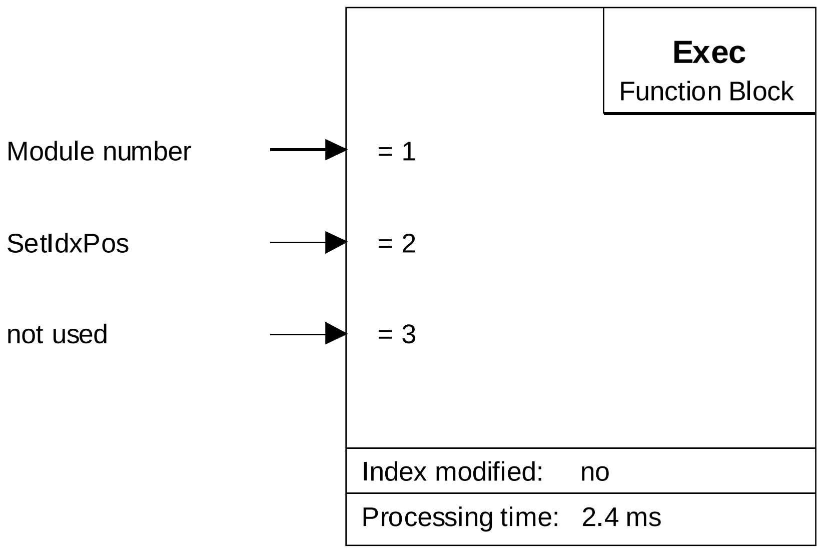
To reset the whole status register, 0 must be loaded in the register for parameter 3.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: ResStatRg | | | | |
| = 3 | PCD register with reset information | R | integer | 8 bit | 0 .. 255 |
| fBrkPt_x | Flag breakpoint | F | | | |
| fOnDest_x | Flag on destination | F | | | |
| fPosErr_x | Flag position error | F | | | |

## SetIdxPos **Instruction:** Set Index Position

[31]



**Function description:**

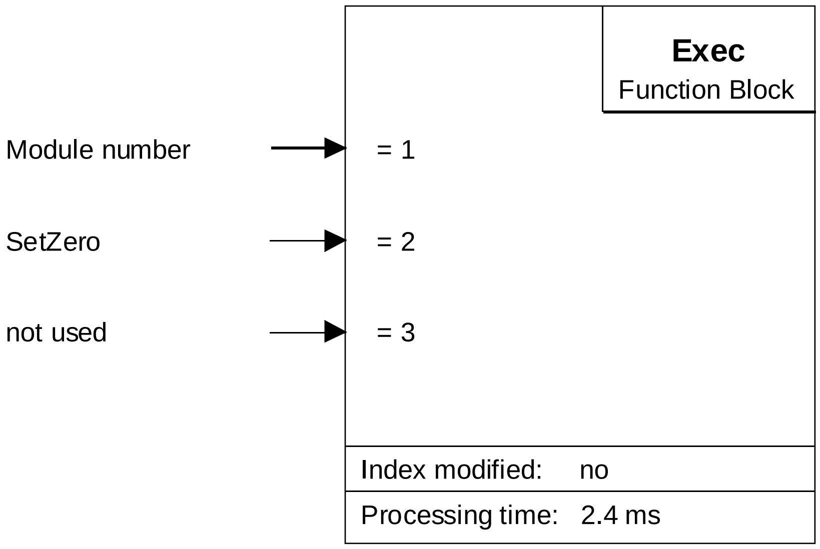
This instruction is used to load the absolute index position on arrival of the next index impulse from the encoder into the module’s index register. The position is adopted when both encoder inputs 'A' and 'B' and index input 'IN' are low. The 'Home' procedure uses this function to simplify definition of the 'Home' function.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: SetIdxPos | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## SetZero **Instruction:** Set Zero position

[32]



**Function description:**

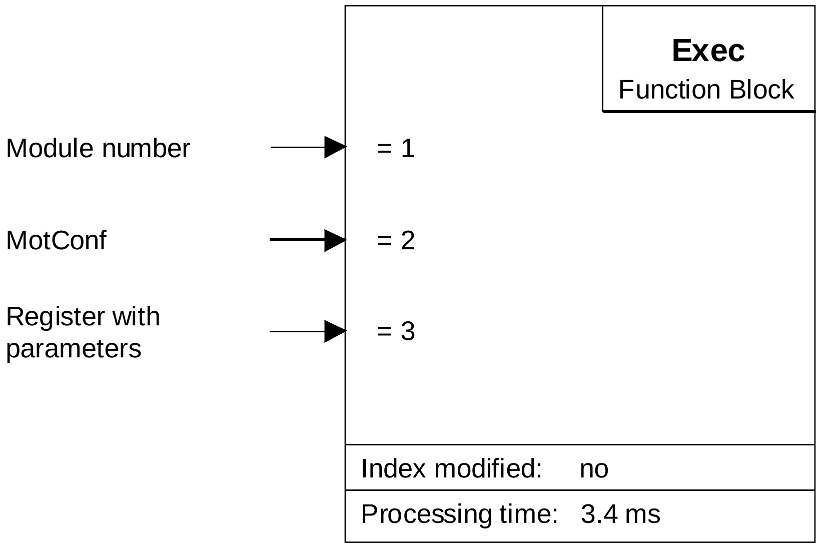
This instruction can be used to set any position to zero. That position then becomes the new 'Home' position.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: SetZero | | | | |
| = 3 | Empty PCD register or 'rNotUsed' | R | | | |

## MotConf **Instruction:** Motion Configuration (speed or position control mode)

[33]



**Function description:**

This instruction is used to configure the motion:

| | |
|---|---|
| bit 0: | L = speed control mode<br>H = position control mode |
| bit 1: | only effective in position control mode:<br>L = backwards<br>H = forwards |

<u>The standard option is "position control mode".</u>

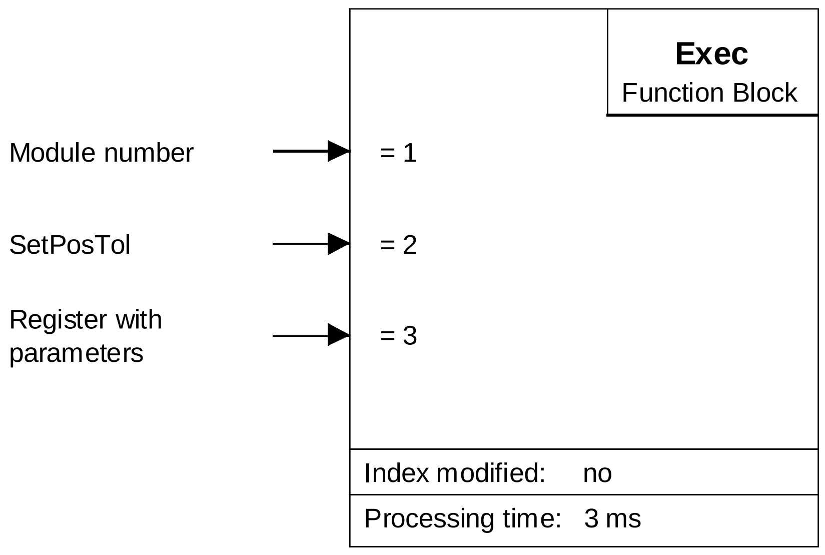
Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: Mot Conf | | | | |
| = 3 | PCD register for configuration | R | integer | 2 bit | 0 .. 3 |

## SetPosTol

**Instruction:** Set Position Tolerance

[34]



**Function description:**

This instruction is used to define the maximum permissible position error between the actual and the target position, calculated by the processor for every sampling interval. Behaviour in cases of excessive error is defined in FB 'Init' with parameter 8. It is specified there whether the motor should be stopped or whether only the 'fPosErr_x' flag should be set.

Before the 'fPosErr_x' flag can be evaluated, FB 'RdStatRg' must be processed, as it deals with the procedure for the position error flag. This flag will be reset at the next 'StartMot' instruction.

Description of input and output elements concerned:

| Par. | Designation/function | Type | Format | Value | Comment |
|---|---|---|---|---|---|
| = 1 | Module number | K | | 1 - 16 | |
| = 2 | Instruction: SetPosTol | | | | |
| = 3 | PCD register for position tolerance | R | integer | 15 bit | 0 .. 32767 |
| fPosErr_x | Position error flag | | | | |

# Appendix B: Summary of all FBoxes for programming in FUPLA

In preparation

Notes

| | |
|---|---|
| From : | To send back to : |
| Company :<br>Department :<br>Name :<br>Address : | SAIA-Burgess Electronics Ltd.<br>Bahnhofstrasse 18<br>CH-3280 Murten (Switzerland)<br>http://www.saia-burgess.com |
| Tel. : | BA : Electronic Controllers |
| Date : | Manual PCD2.H31x |

If you have any suggestions concerning the SAIA® PCD, or have found any errors in this manual, brief details would be appreciated.

**Your suggestions :**