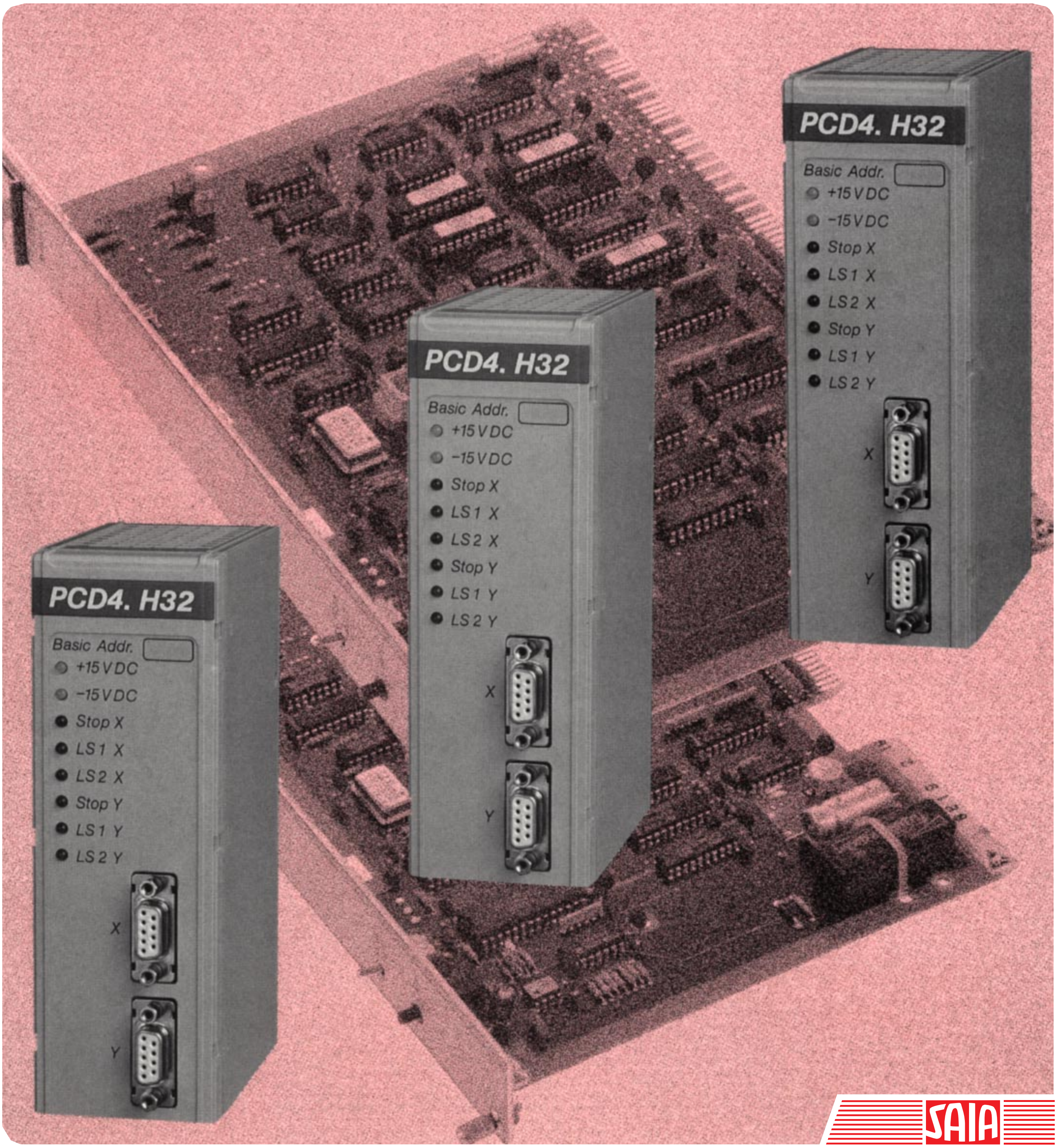
SAIA-Burgess Electronics

SWITCHES • MOTORS • CONTROLLERS

SAIA®PCD

Process Control Devices

Manual
Motion control modules for servo drives PCD4.H3..



Edition 26/729 E1

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

PCD4.H3xx

Edition 26/729 E1 - 10.1990

Subject to technical changes

Updates

Manual : PCD4.H3xx - Motion control modules for servo drives - Edition E1

Date	Chapter	Page	Description

Contents

Notes :

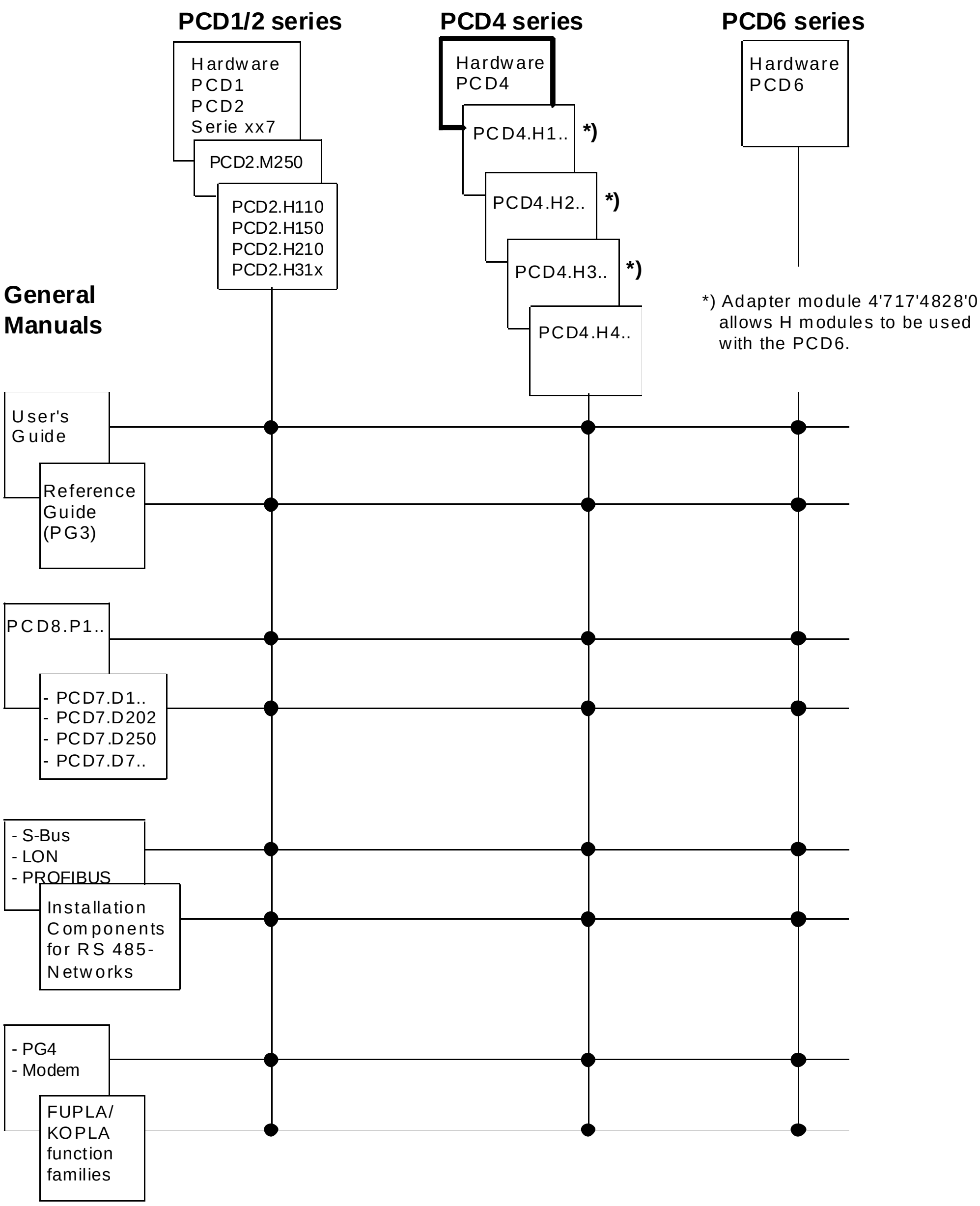
Please note :

A number of detailed manuals are available to aid installation and operation of the SAIA® PCD. These are for use by technically qualified staff, who may also have successfully completed one of our "workshops".

To obtain the best performance from your SAIA® PCD, closely follow the guidelines for assembly, wiring, programming and commissioning given in these manuals. In this way, you will also become one of the many enthusiastic SAIA® PCD users.

If you have any technical suggestions or recommendations for improvements to the manuals, please let us know. A form is provided on the last page of this manual for your comments.

Summary



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, United Laboratories (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".

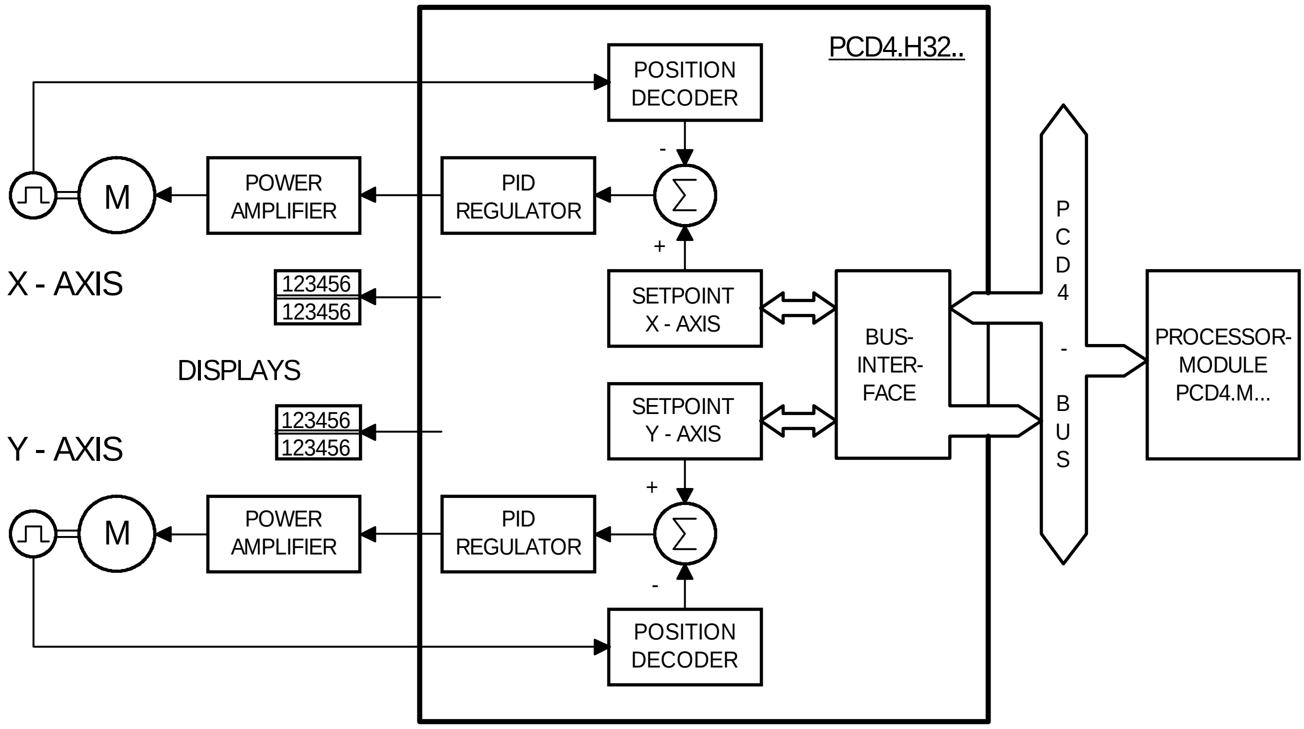
The plant engineer must in turn also contribute his share to the reliable operation of an installation. He is therefore responsible for ensuring that controller use conforms to the technical data and that no excessive stresses are placed on it, e.g. with regard to temperature ranges, overvoltages and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

Block diagram :



Function and use:

The PCD4.H3.. motion control module can position one or two independent axes using variable speed motors (servo motors). These servo motors are AC or DC motors with power drivers and incremental shaft encoders for registering position and velocity.
The module connects to the PCD4 via the PCD4 bus. The module uses 16 addresses. In theory this means that up to 16 motion control modules (32 axes) can be connected to a single PCD4 system.
For each axis a single-chip processor controls movement according to the parameters loaded (speed, acceleration and destination position). Each axis is controlled independently, i.e. interpolation to trace curved courses is not possible. However, the linkage of several axes (point-point) in quasi-synchronous operation can be programmed.

Typical applications:

- Handling robots
- Pick-and-place and automatic assembly machines
- Automatic palleting machines
- Packing machines
- Sheet metal forming machines

Programming:

A software library is provided which contains Function Blocks for controlling the module. This is supplied in readable PCD source code form. The motion control module is therefore easily programmed, without the need for complicated instructions.

Essential characteristcs:

- Position and speed are PID controlled.

- Velocity, destination position and PID parameters can be changed during movement.

- Analogue ±10V or Pulse Width Modulated (PWM) output to drive the motor power stage.

- Digital inputs for reference and limit switch at 24V (source operation).

- Encoder signal inputs for 24V (source or sink operation) or 5V RS422 (differential lines).

- Digital outputs for connection to PCA2.D14 display modules with 2 x 6 digits per axis.

Summary of modules :

Type	Axes	Controller output	Encoder signals
PCD4.H310 [1)]	X	± 10V	24V
PCD4.H320 [1)]	X , Y	± 10V	24V
PCD4.H311 [1)]	X	± 10V	5V (RS422)
PCD4.H321 [1)]	X , Y	± 10V	5V (RS422)
PCD4.H316 [2)]	X	PWM	24V
PCD4.H326 [2)]	X , Y	PWM	24V
PCD4.H317 [2)]	X	PWM	5V (RS422)
PCD4.H327 [2)]	X , Y	PWM	5V (RS422)

1) Standard range

2) Supplied on demand

Notes :

2. Technical data

Displacement control	Incremental : 2 quadrature and index signals (and their inverse signals)
24V inputs	
Signal level	Low = 0..4V High = 19..32V
Input current at 24V	10mA
Operating mode	Source or sink
5V inputs	5V differential RS422 inputs
Isolated	No
Max. frequency	100kHz
Digital inputs	Per axis: 2 limit switches and 1 reference signal at 24VDC
Signal level	Low = 0..4V High = 19..32V
Input current at 24V	10mA
Operating mode	Source
Input filter	100kHz
Digital outputs	To drive PCA2.D14 display module PCA2.D14 (2*6 digits per axis)
2 outputs	Data and clock (shared by both axes)
1 output per axis	Enable (inverse logic: active low)
Output current Ia	1..100mA (not short-circuit protected) Load resistance min. 240 Ohm at 24V DC

Control unit output	To drive the servo-amplifier
Analogue output	±10V (12-bit resolution plus sign bit) Load resistance min. 3kOhm Short-circuit protected
PWM output	**P**uls-**W**idth-**M**odulated (8-bit resolution)
	Signals : SIGN and MAGNITUDE i.e. adjusted for direct control of a bridge driver
	Open collector outputs : Imax. = 500mA Umax. = 50V
Operating mode	Position or speed control

Position determinants

Position	Units : µm Range : [-2^{30} ... + (2^{30} - 1)] / k * 10^{-3} k = f{encoder resolution and spindle ascent}
Velocity	Units : µm/s Range : [-2^{30} ... + (2^{30} - 1)] / k * 22348 * 10^{-6} k = f{encoder resolution and spindle ascent}
Acceleration	Units : µm/s^2 Range : [-2^{30} ... + (2^{30} - 1)] / k * 76206 * 10^{-9} k = f{encoder resolution and spindle ascent}
PID controller	Scanning time : 341µs Proportional, integral and derivative factors are programmable. (Scanning time for derivative part is programmable separately).
Programming	With Function Blocks supplied as PCD source code

Supply

External (user)	+ 24V DC (19V .. 32V) smoothed ripple 10%
External 24V supply current:	
for H310, 320, 316 u. 326	I_{max} = 150mA/axis + encoder supply
for H311, 321, 317 u. 327	I_{max} = 300mA/axis

(Max. current for 5V encodersupply 300mA/axis)

Internal from PCD4 bus	+5V / 120mA per axis ±15V / 5mA per axis (for H310, 311, 320 and 321 only)

Operating conditions

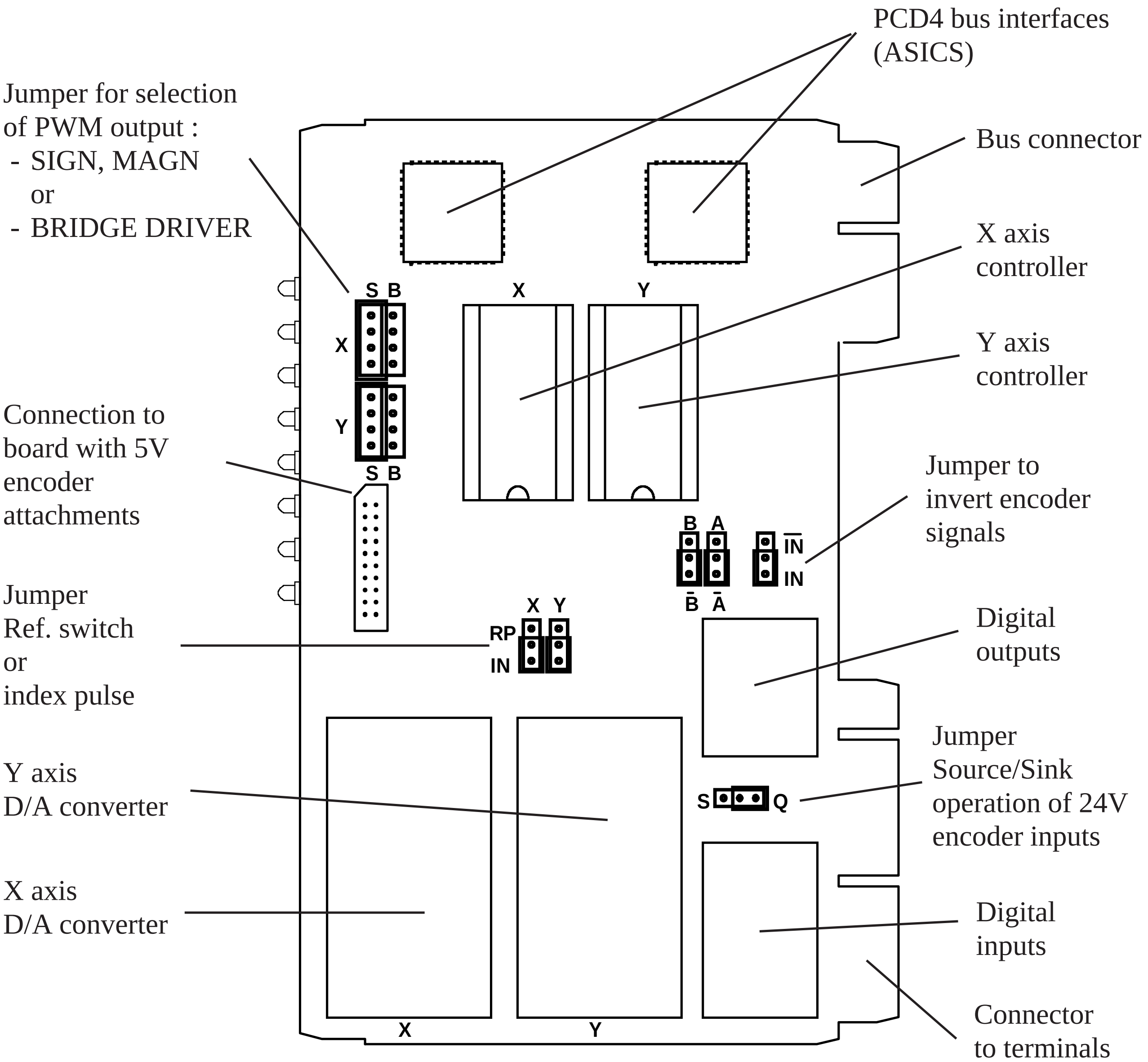
Ambient temperature	0°C .. +50°C without ventilation
Interference resistance	1kV , capacitive coupling, according to IEC 801-4
Mechanical resistance	According to IEC 65A
Storage conditions	Temperature: -20°C .. +85°C Humidity: 0 .. 95%

Notes :

3. Presentation

3.1 Printed circuit board

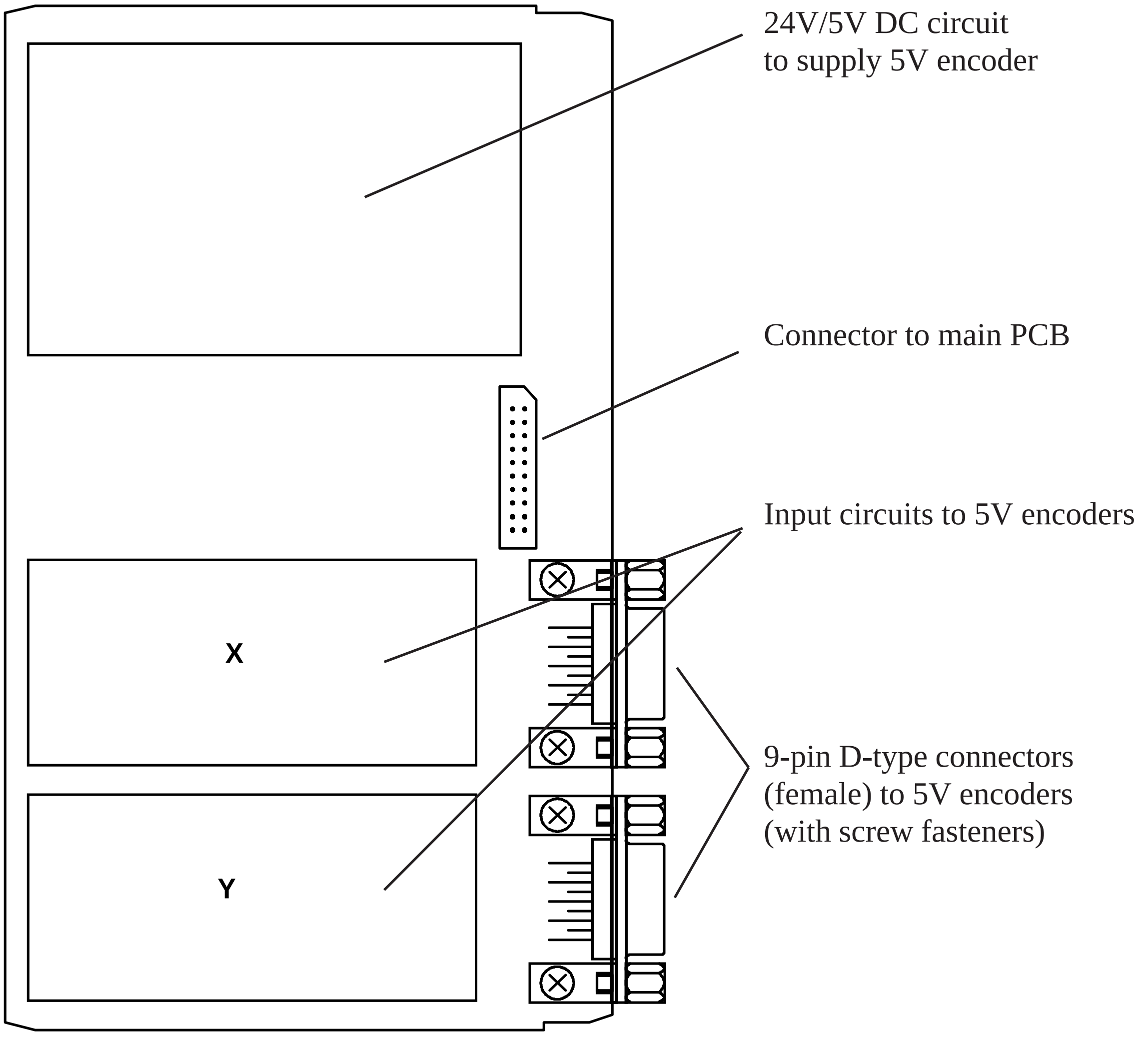
3.1.1 Main PCB (2 axes)



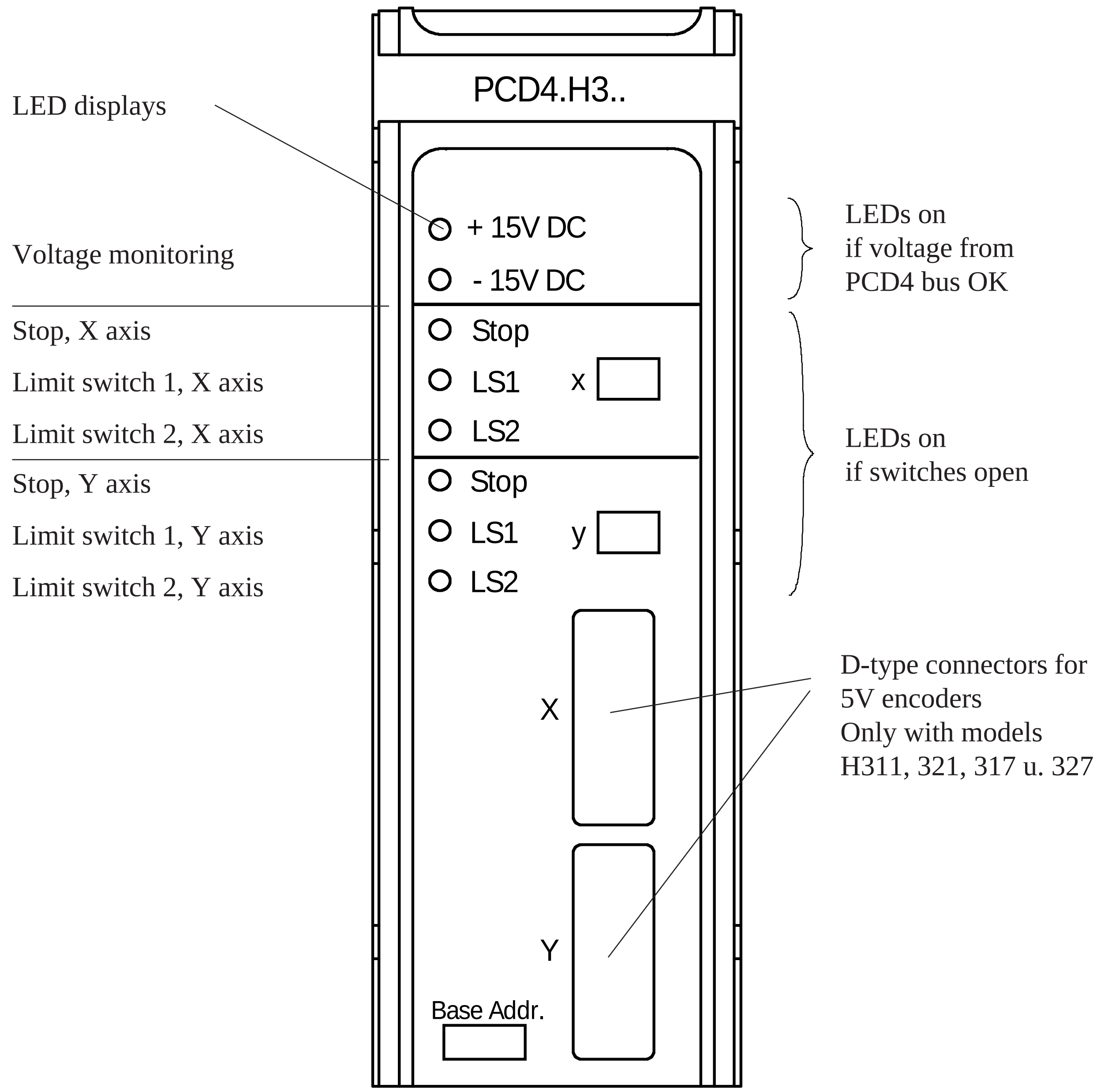
- D/A converter modules are only fitted to models H310, 320, 311 and 321

- Jumpers for the selection of the PWM output are only fitted to models H316, 317, 326 u. 327

3.1.2 Additional PCB for 5V encoder (2 axes)



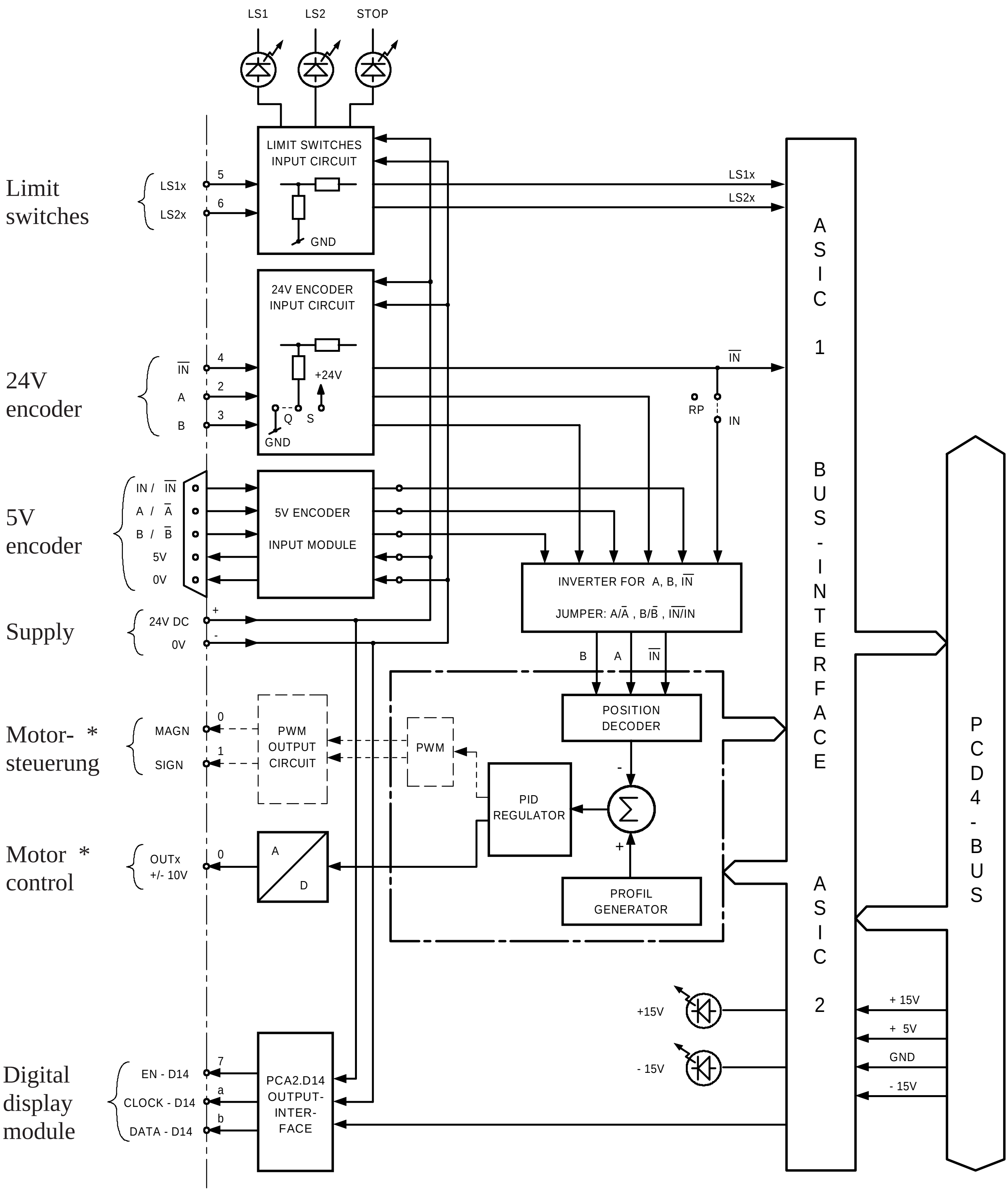
3.2 Front panel



Notes :

4. Logic diagram

Only the X axis is shown

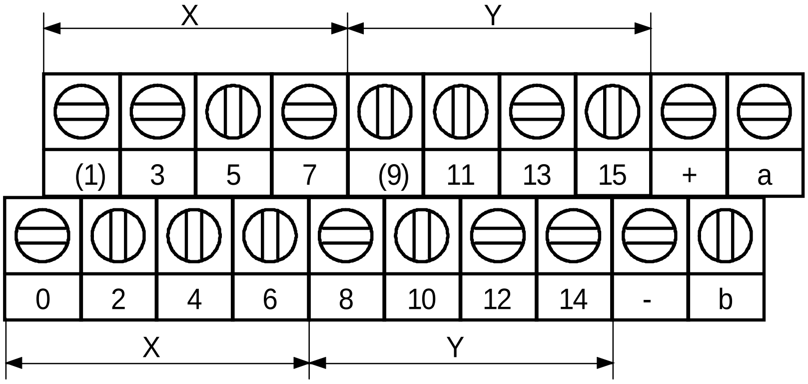


*) Motor control output either PWM or analogue ±10V (see summary of model types).

Notes :

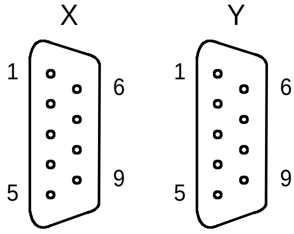
5. Connectors and addressing

5.1 Connectors



Terminal connectors :

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



D-type connectors :
(female)

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

Key :

Out-x, y	Controller output ±10V and PWM-MAGN respectively
PWM-SIGN	Output
Phase-A	Encoder input for Phase A (24V)
Phase-B	Encoder input for Phase B (24V)
/IN (Ref.)	Encoder input for index signal and reference switch
LS1	Limit switch input (24V)
LS2	" " " "
EN-D14	Output ENABLE for PCA2.D14 display module
CLK-D14	Output CLOCK " " " "
DATA-D14	Output DATA " " " "
+24V	Power supply for +24V DC
+5V	5V output for supply of RS422 encoder
GND	Negative connection for 24VDC and 5VDC supplies respectively
PGND	Protective ground connection for the 5V encoder. Must not be connected, as the cable shield connects with the positive ground via the metal casing of the D-type connector and the screw terminal. See also section 6.4.

5.2 Addressing

The module takes up 16 addresses on the PCD4 bus. Since the module has two ASIC bus interfaces, each of the 16 addresses has two uses.

Meaning of the 16 addresses :

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

Addresses indicated are offsets from the base address.
Absolute address = module base address + offset address

Only the addresses marked (*) are of interest to the user.
All other addresses are used by the supplied Function Blocks.

The Enable signals for the PCA2.D14 display are active low, inverters for these outputs are located on the motion control module.

Notes :

6. Operation

6.1 Operating modes

There are two basic operating modes:

- POSITION CONTROL

- VELOCITY CONTROL

Position control

Positioning requires the following command sequence:

1. Input of position and parameters for velocity profile
2. Start positioning
3. Await “destination position reached” signal

After the parameters have been input (PID factors, velocity, acceleration etc.) position operation involves a controlled approach to the destination position, whereby velocity, PID factors and destination position may be changed during movement.

Velocity control

Command sequence:

1. Input of parameters for velocity profile
2. Start movement
3. Cease movement with input of a stop command

In velocity control operation, velocity is increased by the defined rate of acceleration until the target velocity is reached. Operation is then controlled at this velocity until a stop command is received. The target velocity can be changed during movement.

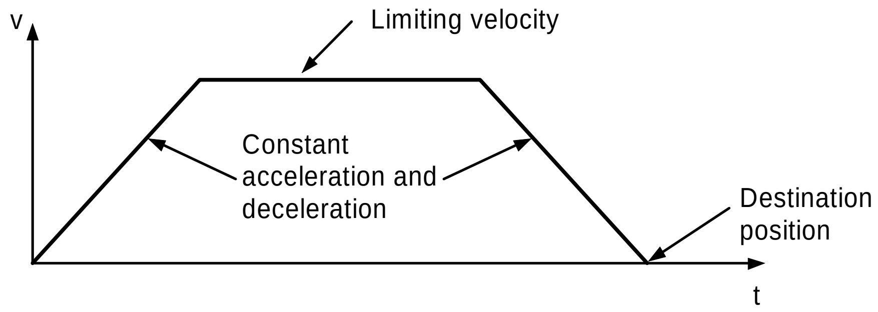
Functional units

The logic diagram shows that the motion control module consists of the following major functional units:

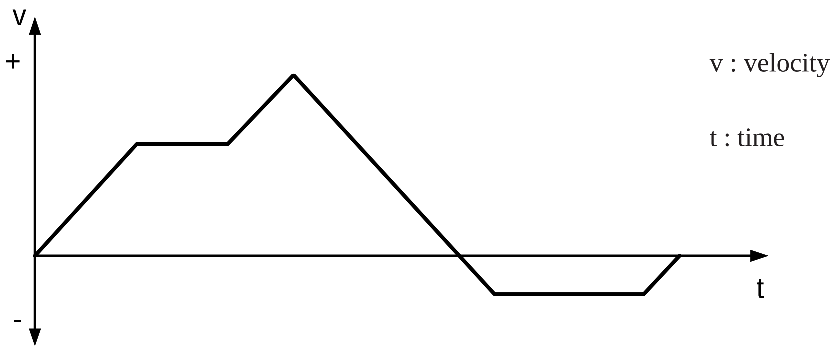
- Velocity profile generator
- PID controller
- Position decoder and input circuit
- Bus interface (ASIC) to PCD bus
- D/A converter for analogue control output or generator for pulse width modulation (PWM)

6.2 Velocity profile generator

According to the indicated acceleration and velocity, the profile generator calculates the setpoint velocity as a function of the time in positioning and velocity mode. When operating in positioning mode, the difference between the setpoint and actual positions is sent to the PID controller. Very precise positioning of the motor is thus achieved.



Standard velocity profile



Velocity profile with setpoint velocity and position altered during movement.

Velocity and destination position can be changed at any desired point during the movement, and the controller will accelerate or decelerate accordingly at the defined rate of acceleration.

In velocity control operation, the controller accelerates to the user-defined velocity and continues at a constant velocity until a stop command is received.

Functional principle of velocity control:

The setpoint position is continuously augmented (according to the desired velocity). The difference between setpoint and actual position (which is worked out by the encoder) is in turn passed on to the PID controller. The latter compensates for fluctuations in velocity, caused by any effects of interference, by immediately increasing or reducing the controller output.

If the motor does not reach the setpoint velocity (e.g. because of a blocked rotor), the difference between setpoint and actual position is very large. This produces a position error message, which can trigger an alarm or automatically stop the motor. The maximum permissible position error is an adjustable value.

6.3 PID controller

The PID controller enables the motor to approach the destination position exactly and to maintain this position, as the controller is active until a stop command is received.
The controller uses the following algorithm:

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

Where :	U(n)	-->	Control output for motor
	e(n)	-->	Position error at n'th scanning
	n	-->	Scanning for integral part
	n'	-->	Scanning for derivative part
	kp	-->	Proportional factor
	ki	-->	Integral factor
	kd	-->	Derivative factor

User-definable parameters:

- Control factors kp, ki, kd
- Derivative scan time
- Integration limit (IL) for integral part

It is possible to change **factors kp, ki and kd** during a movement.
Scan time for the **proportional and integral parts** amounts to 341µs. This means that control output is refreshed at an interval of 341µs.
Scan time for the derivative part can be defined in steps of 341µs (max. 256*341µs). Longer scan times should be selected for low velocity operation.

The integration limit IL limits the amount resulting from the expression:

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

6.4 Position decoder and input circuit

Registering position and velocity

The precise position and speed of the motor are registered by an incremental turn encoder. The following encoder signals can be connected:

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

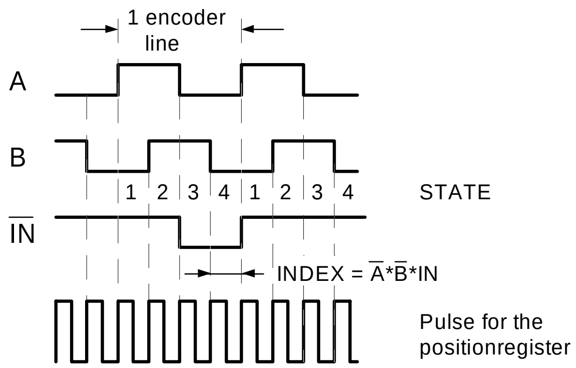
} A,B,IN and $\overline{A}$,$\overline{B}$,$\overline{IN}$ (terminal connectors)
24V signals in source or sink operation

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

} A,$\overline{A}$; B,$\overline{B}$; IN,$\overline{IN}$; (front D-type connector)
5V RS 422 inputs (differential line)

Inputs A, B, $\overline{IN}$:

Diagram to show state of signals A,B,$\overline{IN}$ on position decoder :



Direction: ← Neg. | Pos. →

STATE	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 :

At each change of state (0->1 and 1->0) of the signals A and B, the internal position register is raised or lowered by 1. In this way the four-way resolution of encoder partition is obtained. The input for the target position must accordingly also be multiplied by four.
For the position decoder, the signals must have exactly the same sequence as shown in the above figure. If the encoder supplies other signals, a jumper must be used to invert them (see following pages).

Input $\overline{\text{IN}}$:

In the case of modules for 24V encoders (types H310, 320, 316 and 326), input $\overline{\text{IN}}$ can be used as an input for the index pulse (zero signal from encoder) or reference point.

- Use as index pulse input:
  (IN/RP jumper in IN position)

  Each time all three encoder signals are at zero, and before the function block “SetIP” (set index position) has been called, the absolute motor position is written to the index position register.

- Use as reference point input:
  (IN/RP jumper in RP position)

  A reference switch can be connected, for example, to define position zero. But to do this, the IN/RP jumper must be in the RP position, making the input no longer active for the position decoder.

  When using modules H310, 320, 316 or 326 with input $\overline{\text{IN}}$ as an index signal, the reference switch must be connected to an input on a digital input module (e.g. E 100).

Modules for 5V encoder signal connection

These are equipped with an additional printed board assembly, where the 5V supply is produced and the encoder connections attached to a D-type plug. When modules H311, 321, 317 or 327 are used, a reference switch (on the terminals) as well as the index signal (D-type plug) can be connected.

As can be seen from the following input diagram, shielded cable must be used to connect with the 5V encoder:

- maximum cable length : 20m
- minimum conductor cross-section : $0.25 mm^2$

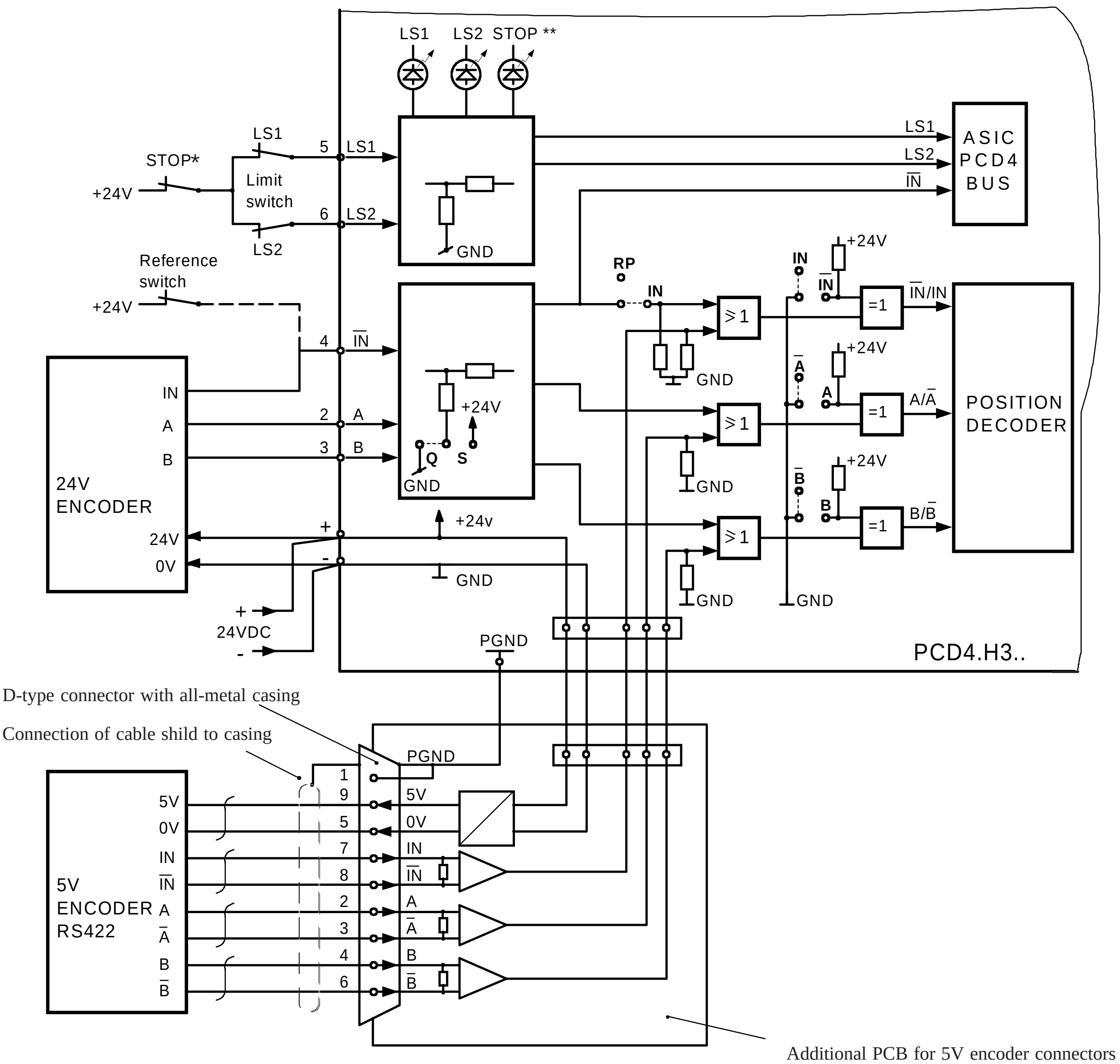
In order to ensure the specified resistance to interference, a D-type connector with an all-metal casing must be used, providing direct connection to the PCD4’s protective ground (PGND).

Limit switches and reference switch

The limit switches (inputs LS1 and LS2) and the reference switch (input IN) require a 24V DC source. The signals from these switches are readable via the PCD4’s bus. This means that they must be monitored by the user program, so that any necessary procedures can be executed.
The limit switches (LS1/2) and the optinal connected stop switch should not be used as safety precaution. **Additional safety and emergency switches** should be provided, which should act directly on the main motor drive circuits.

Input diagram and connectors:

(only one axis is shown)



*) For safety requirements, see previous section "Limit switches and reference switch"

**) LED status table (LS1, LS2 and STOP) :

INPUTS		LED		
LS1	LS2	LS1	LS2	STOP
24V	24V	OFF	OFF	OFF
OPEN	24V	ON	OFF	OFF
24V	OPEN	OFF	ON	OFF
OPEN	OPEN	OFF	OFF	ON

Jumper selection (see also printed circuit board, section 3.1) **:**

Jumper	Function	Factory setting	
Q/S 1)	Source/sink operation inputs A,B,$\overline{IN}$	Source operation (Q)	24V encoders only
IN/RP 2)	Input for index pulse or reference point	Index pulse (IN)	
A/$\overline{A}$ 3)	Inverse phase signal A	$\overline{A}$	all 24V and 5V encoders
B/$\overline{B}$ 3)	Inverse phase signal B	$\overline{B}$	
IN/$\overline{IN}$ 3)	Inverse index pulse $\overline{IN}$	IN	

1) Switching between source/sink operation for 24V encoder signals takes place with one jumper only for both axes.

2) For 24V encoders, it is possible to connect via the terminals either 4 (X axis) or 12 (Y axis):

- encoder index signals
  or
- reference switches

For 5V encoders the index signals are supplied via the D-type connectors, i.e. terminals 4 and 12 are left free for the reference switch (jumper in position "RP"). The "IN,RP" jumper can be selected separately for each axis on the printed circuit board.

3) Encoder signal inversion (24V and 5V) also requires 1 jumper each for A, B and IN, shared in the same way for both axes.

Opening the module housing to change jumper

To change jumper position, the printed circuit board must be removed from the module housing. This is done by pressing in the snap fastenings on either side of the front panel. Next, unscrew the screw at the top left of the module which holds the PCB, so that the printed circuit board can be removed from the housing.

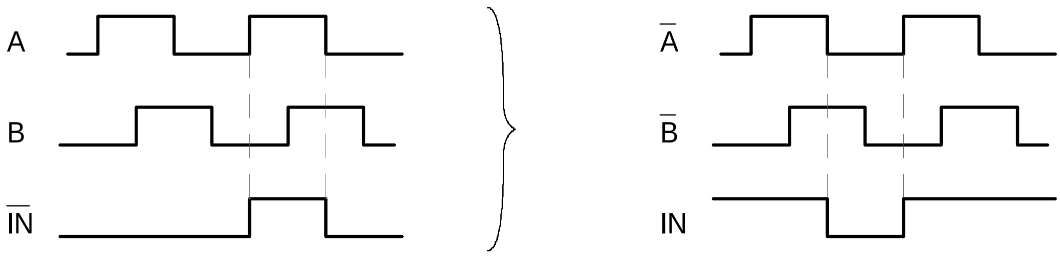
When the jumper has been properly inserted, close the casing and fasten the PCB screw again.

Note: The main printed circuit board, like the analogue modules, contains components which are sensitive to electrostatic charges.

Applied example of jumper selection

An encoder supplies the following 24V signals in source operation:

Signals from encoder | Signals at position decoder



To produce the desired sequence of signals at the position decoder input, all three signals must be inverted.

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

} corresponds to factory settings

Important :

If **24V encoders** are being used with **reference switches** (jumper "IN,RP" in position RP), the $\overline{IN}$,IN jumper must be set at **IN**.

Reason: The index signal $\overline{IN}$ at the position decoder input should not be left permanently at zero, since at high speed this can lead to an error function in the controller.

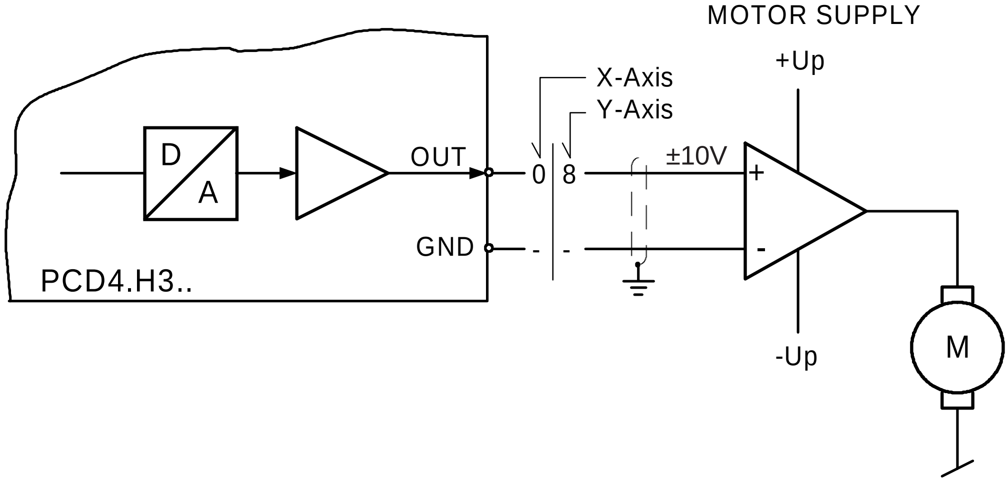
6.5 D/A converter (analogue control output)

Modules:	PCD4.H310 " .H320 " .H311 " .H321	have an analogue output for motor control

A 12-bit D/A converter is fitted for each axis.

Analogue output connection:

(only one axis is shown)



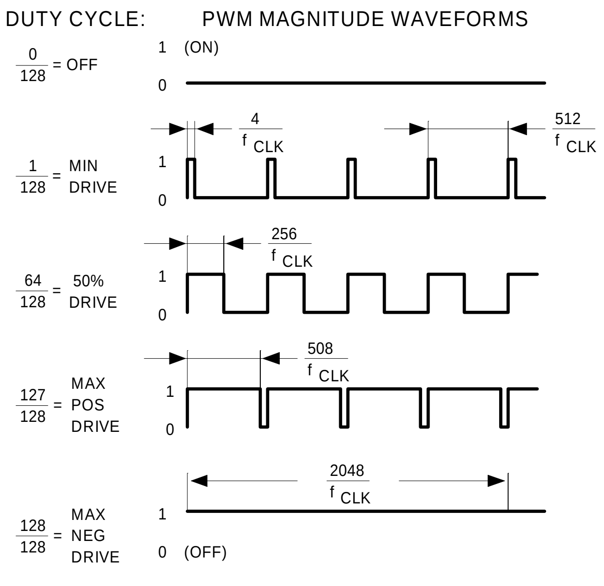
6.6 PWM generator

Modules: PCD4 .H316, " .H326, " .H317, " .H327 — have a PWM output with SIGN and MAGNITUDE signal, or with output logic for running a bridge driver directly.

In some power amplifiers, the above-mentioned logic is integral. Therefore it is possible to output the SIGN and MAGNITUDE signals either directly or via a logic circuit using a jumper.

PWM signal on PWM generator output:
(without SIGN signal)

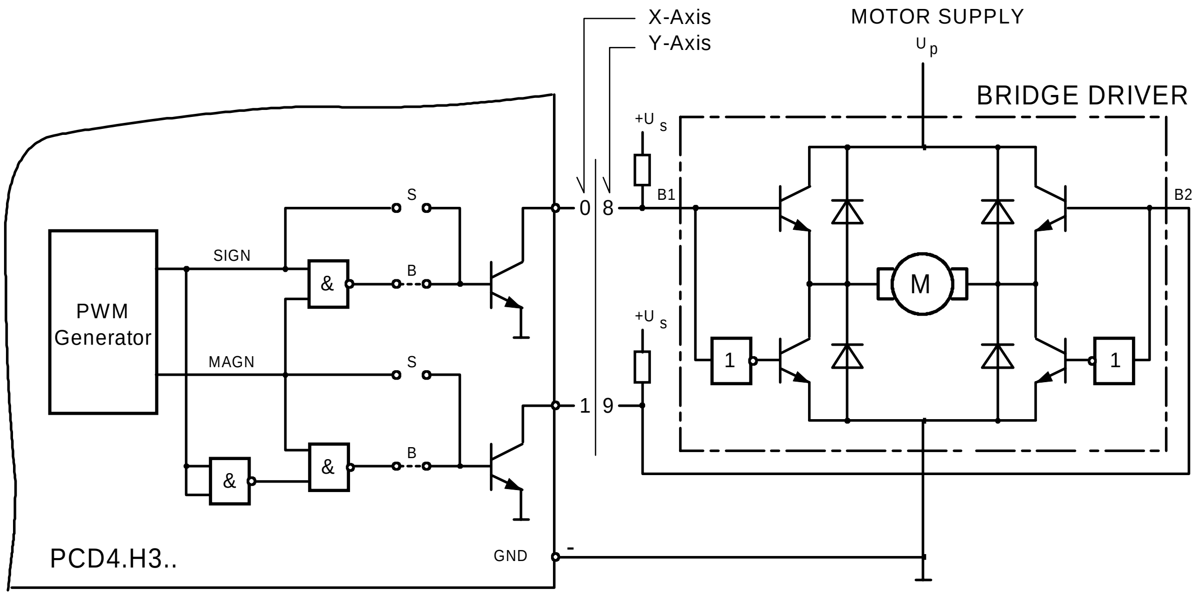
The signal has 8-bit resolution.
Range: -128 .... +127



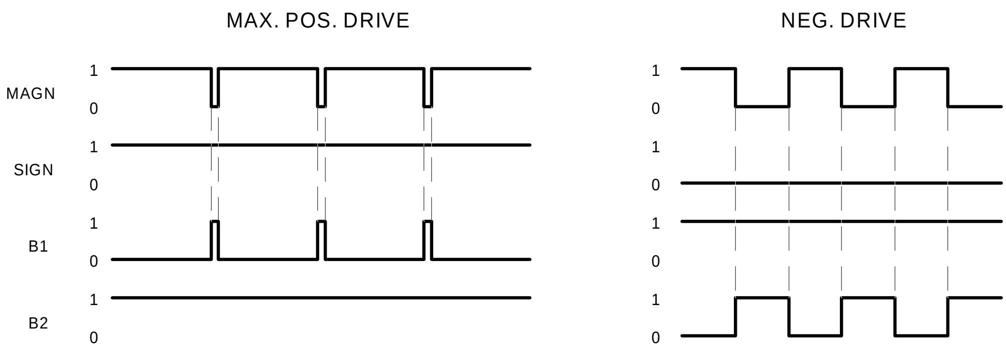
f_{CLK} = 6MHz

Logic diagram of PWM output:

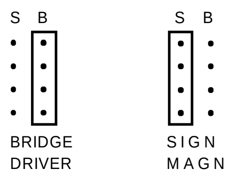
(only one axis is shown)



The following diagram shows the connection between the SIGN/ MAGN signal and the output for running a bridge driver directly.



Jumper selection:



Separate selection of X and Y axis outputs.

Factory setting: SIGN, MAGN (Pos. S)

Notes :

7. Writing programs for the H3 module

7.1 Software installation

7.1.1 The PCD9.H3E1 software package

(Package PCD9.H3E1 for 5.25" disks, PCD9.H3E6 for 3.5" disks)
The PCD9.H3 software package contains function blocks, written in PCD instruction list code, which can be called from a user program to control the H3 module.

The package consists of the following two files:

H3DEF.SRC	This file contains all the symbol declarations for the package. The H3 installation is configured here.
H3FB.SRC	This file contains the function blocks for controlling the module.

The package has the following requirements:

- number of program lines	≤	1250
- FB call nesting levels		6

7.1.2 File assembly and linkage

There are two ways of assembling and linking H3 files. Either external (global) symbol definitions can be used, or local symbols defined in the H3DEF.SRC can be used by including this file in the user program with the$INCLUDE assembler directive. The H3 program can support either method, symbols which control conditional assembly can be defined to select the method.
The default is for assembly without using external symbol definitions.

Using local symbol definitions

The symbol definition file H3DEF.SRC must be included at the start of the user program with the $INCLUDE directive, and the files must be edited so that these EQUates are defined as follows:

- H3DEF.SRC : PUBLSYM EQU 0
- H3FB.SRC : EXTNSYM EQU 0

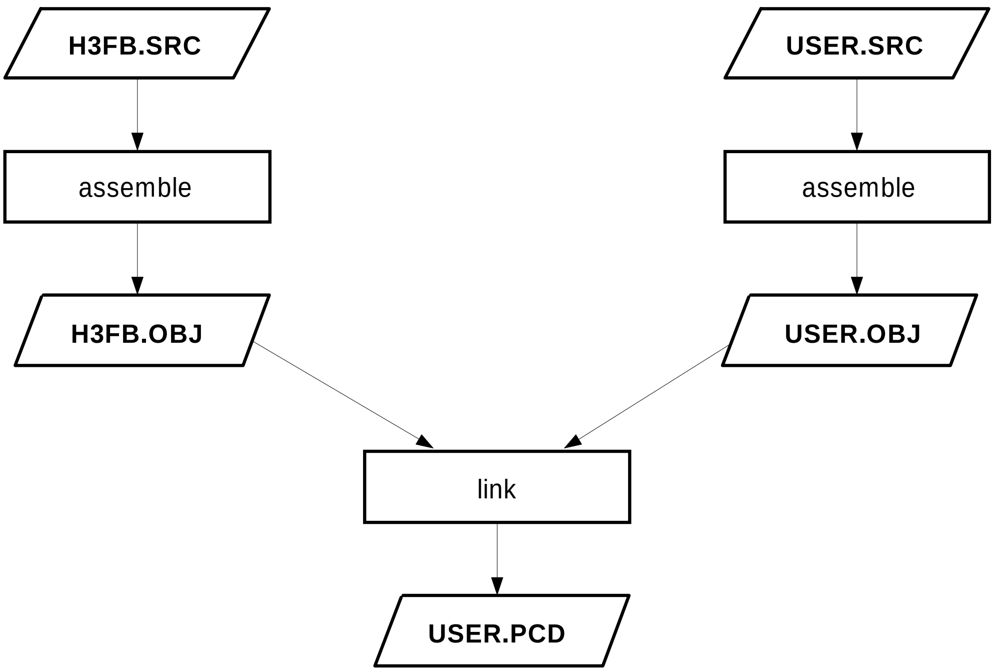
Include the H3 file in the user program:

User program file (e.g. USER.SRC)

```
<top of file> (title, program heading, etc.)

$INCLUDE        H3DEF.SRC
$INCLUDE        <other include files>
      .
      "
      "
<user program>
      "
      "
<end of file>
```

The H3 symbol definition must be included before any other include files (which may use symbols from H3DEF.SRC).

The diagram below shows how the files are assembled and linked.



Using external symbols

All files are assembled individually and then linked. The files must first be edited so that these EQUates are defined as follows:

- H3DEF.SRC : PUBLSYM EQU 1

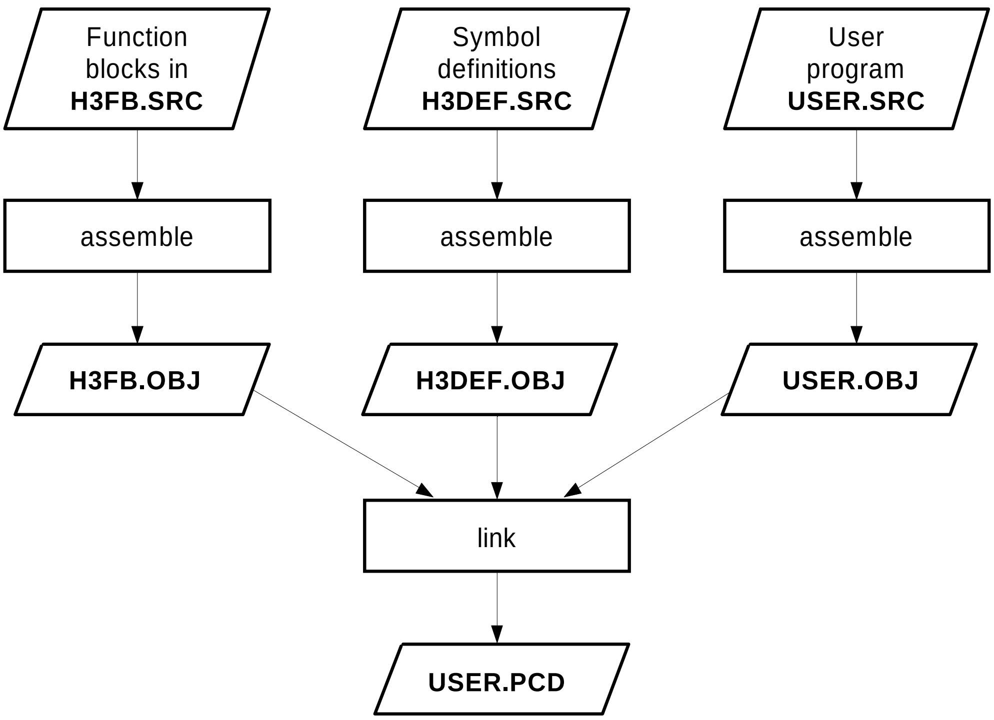
- H3FB.SRC : EXTNSYM EQU 1

If this method of assembly is chosen, all symbols used (from the H3DEF.SRC file) must be defined as EXTerNal in the user program. The register and flag block addresses ("RA1", "FA1" etc.) must be defined in the file, since the assembler can't add two external symbols (e.g. SET F FStart+FA1).
Definition of "RAi" and "FAi" in the user program (these are described Chapter 7.1.3):

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

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

For the same reason, the symbols "NoRfeA" and "NoFfeA" must also be defined. These definitions can be taken from the H3DEF.SRC file.

The diagram below shows how the files are assembled and linked.



7.1.3 Configuring H3 installation in the H3DEF.SRC file

The H3 installation must be configured by editing H3DEF.SRC before starting programming.
Configuration details include the base address of the first H3 module, the number of axes, base addresses of elements used etc. All definitions are set to default values, which can be changed if required.
The configuration data, found at the top of the H3DEF.SRC file, is listed below.

Symbol	Default value		Comment
FMAH3	EQU	0	; First Module Address H3 ; Defines the base address of the ; first H3 module. ; NOTE: All H3 modules must be ; inserted one after the other ; without a gap on the PCD4 bus.
IMode	EQU	6	; Initialization Mode ; Defined according to the output ; used for the set point (analogue/PMW). ; Modules H310,311,320,321: IMode=6 ; Modules H316,317,326,327: IMode=5
MNA	EQU	2	; Max. Number of Axes ; Defines the number of axes used. ; The required number of registers ; and flags are reserved according ; to this information.
BAF	EQU	2000	; Base Address of Flags
BAR	EQU	2000	; Base Address of Registers
BAC	EQU	1000	; Base Address of Counters
BAFB	EQU	900	; Base Address of Function Blocks

Symbol	Default value		Comment
RA1	EQU	0*NoRfeA	; Register block address for Axis 1. ; This constant indicates the first ; register of the register block ; occupied by axis no. 1. ; The constant "NoRfeA" defines the ; number of registers occupied ; per axis. ; Do not change "NoRfeA".
RA2	EQU	1*NoRfeA	; Register block address ; for Axis 2.
FA1	EQU	0*NoFfeA	; Flag block address for Axis 1. ; This constant indicates the first ; flag of the flag block occupied ; by axis no. 1. ; The constant "NoFfeA" defines ; the number of flags occupied ; per axis. ; Do not change "NoFfeA".
FA2	EQU	0*NoFfeA	; Flag block address for Axis 2

If more than two axes are used, further register and flag block addresses (RA1, RA2, FA1, FA2 ...) must be defined.

For example:

Axis i	RAi	FAi
1	0*NoRfeA	0*NoFfeA
2	1*NoRfeA	1*NoFfeA
3	2*NoRfeA	2*NoFfeA
4	3*NoRfeA	3*NoFfeA
5	4*NoRfeA	4*NoFfeA

etc.

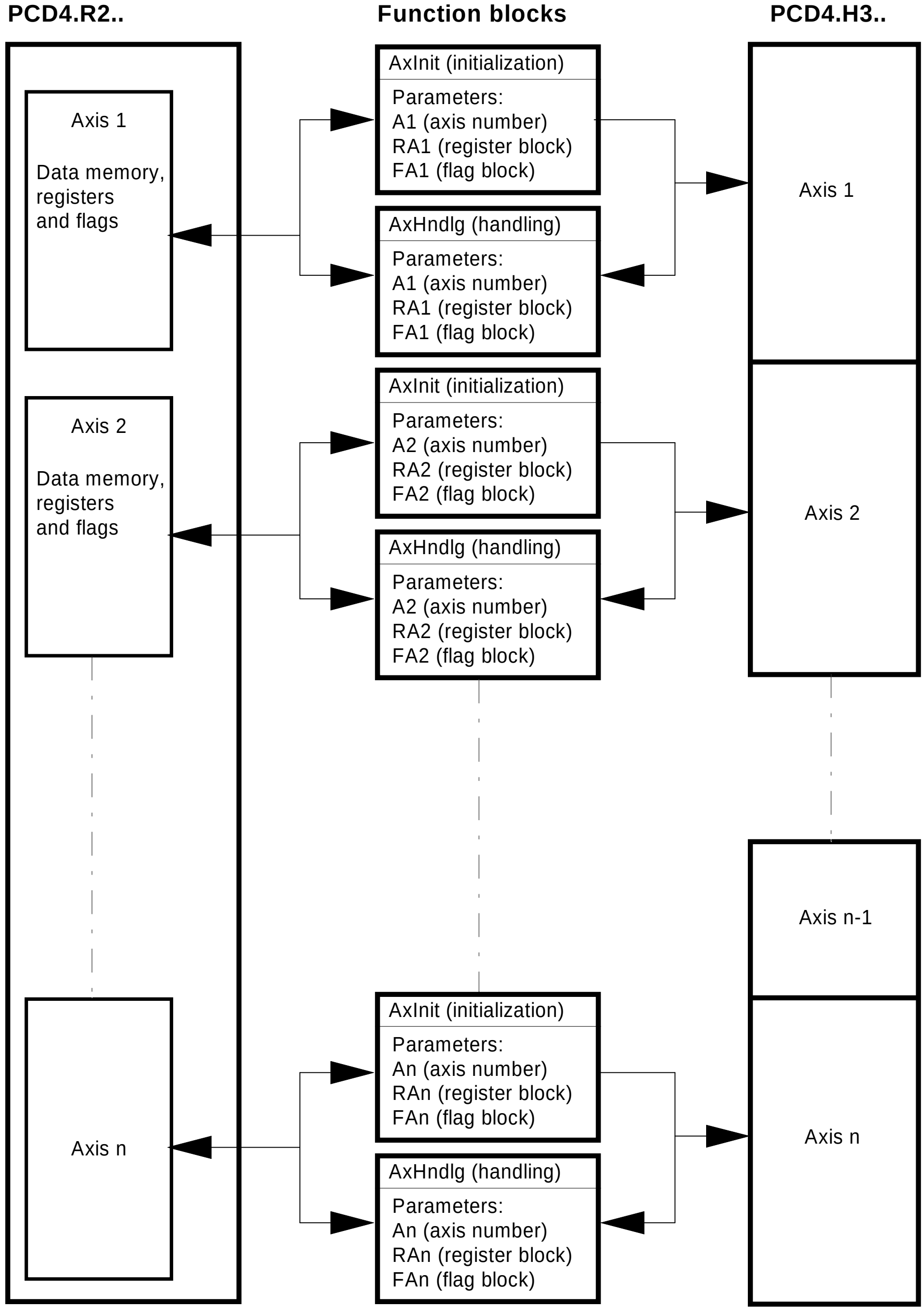
Following the configuration data are further symbol declarations which must NOT be changed.

After configuration, it is useful to know how many elements are used by the H3 software. At the end of H3DEF.SRC is a table of symbols which are assigned values by the assembler. From the H3DEF.LST listing file, it is possible to determine how many elements are used.

Elements used by the H3 software:

Symbol			Value	Comment
TFl	EQU	F	NBF-BAF	; Total used Flags
TCo	EQU	C	NBC-BAC	; Total used Counters
TRe	EQU	R	NBR-BAR	; Total used Registers
TFB	EQU		NFB-BAFB	; Total used Function Blocks
NFF	EQU	F	NBF	; Next free Flag
NFC	EQU	C	NBC	; Next free Counter
NFR	EQU	R	NBR	; Next free Register
NFFB	EQU		NFB	; Next free Function Block

7.1.4 Data transfer CPU <--> H3 module



The H3 module is controlled by two function blocks "AxInit" and "AxHndlg", and axis-specific data stored in data memory. The register and flag blocks defined for each axis store the operating parameters, which are read and written as data is exchanged with the H3 module. The correct axis and data blocks are used by specifying the axis number "Ai", the register block address "RAi" and the flag block address "FAi" when the function blocks are called.

7.1.5 Organization and access of data memory

Register allocation

Register block address

Base register address

Adress	Symbol	Designation
	Axis 1	
BAR+RA1+0	KProp	Proportional factor
BAR+RA1+1	KInt	Integral factor
BAR+RA1+2	KDer	Derivative factor
BAR+RA1+3	IntL	Integration limit
BAR+RA1+4	SampI	Sampling interval
BAR+RA1+5	MCFac	Motion control factor
BAR+RA1+6	PosEr	Position error
BAR+RA1+7	DestP	Destination position
BAR+RA1+8	BrkP	Break position
BAR+RA1+9	Veloc	Velocity
BAR+RA1+10	Accel	Acceleration
BAR+RA1+11	StaFRR	Status flag reset register
BAR+RA1+12	MCW	Motion control word
BAR+RA1+13	RActP	Actual position
BAR+RA1+14	RSetP	Set point position
BAR+RA1+15	RActV	Actual velocity
BAR+RA1+16	RSetV	Set point velocity
BAR+RA1+17	RIndP	Index position
BAR+RA1+18	RIntTS	Integration Term Sum
BAR+RA1+19	RSigB	Signal register
BAR+RA2+0 … BAR+RA2+19	Axis 2	
BAR+RA3+0 … BAR+RA3+19	Axis 3	
BAR+RAn+0 … BAR+RAn+19	Axis n	
	Shared registers	Used by function blocks as workspace memory

Flag allocation

Flag block address

Basis flag address

Adresse	Symbol	Designation
	Axis 1	
BAF+FA1+0	OnDest	Destination psn. reached
BAF+FA1+1	IPuls	Index pulse obtained
BAF+FA1+2	WrapOc	Position register overflow
BAF+FA1+3	ExcPEr	Position error
BAF+FA1+4	BrkPos	Break position reached
BAF+FA1+5	DplM	PCA2.D14 display mode
		Command flags:
BAF+FA1+6	FLdDR	Load destination relative
BAF+FA1+7	FLdDA	Load destination absolute
BAF+FA1+8	FLdVR	Load velocity relative
BAF+FA1+9	FLdVA	Load velocity absolute
BAF+FA1+34	FBackw	Backward at defined velocity
BAF+FA2+0	Axis 2	
BAF+FA2+34		
BAF+FA3+0	Axis 3	
BAF+FA3+34		
BAF+FAn+0	Axis n	
BAF+FAn+34		
	Shared flags	Used by function blocks as workspace memory.

Counter allocation

Only one counter is used, all axes share this counter.

User program access to registers and flags

Axis parameters are referenced by their symbol names(defined in H3DEF.SRC). The parameters for each axis share the same symbol names. To reference the actual parameter, the start address of the parameter block for the required axis is added to the symbol name. If only the symbol name is used, the parameters for axis 1 are referenced. The start addresses of the parameter blocks are defined as the constants "RAi" and "FAi" (the register and flag blocks for axis i). This is shown in the examples below.
For the sake of readability, symbol names are written using mixed upper and lower case letters in the H3 software. However, since the PCD assembler ignores the case, symbol names in the user program can be written with or without capital letters. E.g. "RA1" is the same as "ra1".

Before a parameter can be loaded into the H3 module, the correspon-ding register must first be loaded with the desired value.

Example: To load the "DestP" register (destination position) for the given axis:

```
- Axis 1   —>  LD   R   DestP+RA1
                        Value

- Axis 3   —>  LD   R   DestP+RA3
                        Value

- Axis i   —>  LD   R   DestP+RAi
                        Value
```

In order to load the destination position into the H3 module, the "Load Destination Position" command must be executed. By setting the flag "FLdDA" (Load Destination Absolute) the "DestP" register will be read by function block "AxHndlg" and loaded into the H3 module.

Example: To load the "destination absolute" data into the H3 module for the given axis

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

Once motion has started, it is necessary to wait until the destination position has been reached. To do this, the status flag "OnDest" must be polled.

Example: To poll the "OnDest" status flag for the given axis

- Axis 1	—>	STH F	OnDest+FA1
- Axis 3	—>	STH F	OnDest+FA3
- Axis n	—>	STH F	OnDest+FAn

7.2 Main function blocks: "AxInit" and "AxHndlg"

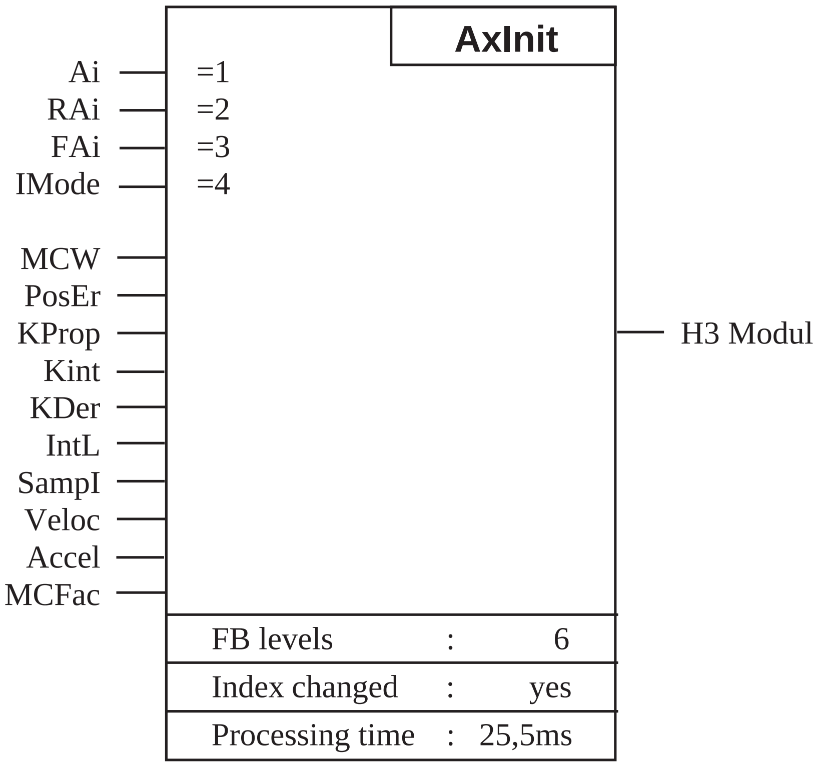
Communication with H3 modules is done exclusively by calling the two function blocks (FBs) "AxInit" (Axis Initialisation) and "AxHndlg" (Axis Handling).
When these FBs are called, the axis number, register block address and the flag block address are supplied, so that the corresponding data and axis is addressed. The FBs must be called separately for each axis.

AxInit

Function block: - **Ax**is **Init**ialisation

AxInit

Software package: PCD9.H3E1



Functional description:

This FB is used to initialize an axis of the H3 module. It must be called before an axis can be used. It is best called from the start-up XOB 16, so that it is called only once. This FB executes several commands, some of which can also be executed by the axis handling FB "AxHndlg", and are not described in detail here.

The following functions are executed by "AxInit":

1. Reset controller in H3 module

This resets to zero all the motion parameters (acceleration, velocity, destination position, and break position), and all the PID parameters (including the controlled output). The "actual position" is also the zero position.

2. Initializes the regulator output port

The output port is initialized according to the "IMode" parameters. The "IMode" constant must be defined in file H3DEF.SRC according to the H3 module used.

3. Selects the operating mode —> see command “FSelOM”

4. Loads the position error —> see command “FSetPEr”

5. Loads regulation parameters —> see command “FLdRP”

6. Updates regulation parameters —> see command “FUpDRP”

7. Loads acceleration —> see command “FLdAA”

8. Loads velocity —> see command “FLdVA”

9. Resets all command flags —> see FB “AxHndlg”

Example: To call the function block for axis 3

```
CFB     AxInit
        3           ; Axis number 3
        RA3         ; Register block for axis 3
        FA3         ; Flag block for axis 3
        IMode       ; Initialisation mode
```

Description of inputs and outputs

Unless otherwise specified, all symbol names used are as defined in H3DEF.SRC and must not be changed.

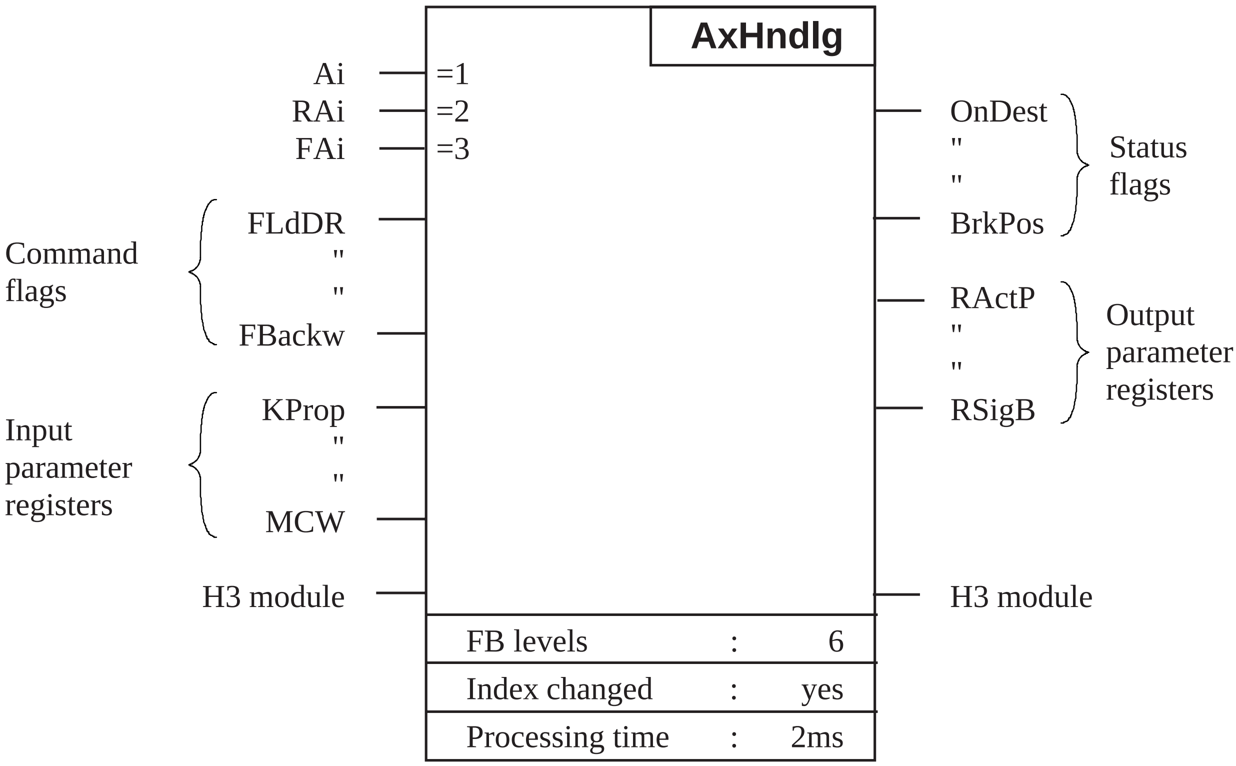
Symbol	Designation / Function	Para-meters	Data		
			Type	Format	Value
Ai	**A**xis number i Ai is not defined as symbol in H3DEF.SRC, an absolute value is used when calling the FB.	yes	K	Integer	1...32
RAi	**R**egister block address **A**xis i	yes	K	Integer	(i-1) * 20
FAi	**F**lag block address **A**xis i	yes	K	Integer	(i-1) * 35
IMode	**I**nitialization **Mode**	yes	K	Integer	5H/6H
MCW	**M**otion **C**ontrol **W**ord	no	R	Binary	-
PosEr	**Pos**ition **Er**ror (number of pulses)	no	R	Integer	0...32767
KProp	**Prop**ortional factor	no	R	Integer	0...32767
KInt	**Int**egral Factor	no	R	Integer	0...32767
KDer	**Der**ivative Factor	no	R	Integer	0...32767
IntL	**Int**egration **L**imit	no	R	Integer	0...32767
SampI	**Samp**ling **I**nterval	no	R	Integer	0...255
Veloc	**Veloc**ity	no	R	Integer	see command “FLdVA”
Accel	**Accel**eration	no	R	Integer	see command “FLdAA”
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Floating point	see command “FLdDA”

AxHndlg

Function block: - **Ax**is **Handl**in**g**

AxHndlg

Software package: PCD9.H3E1



Functional description:

After initialisation of the axis with "AxInit",communication with the H3 module is exclusively through this function block. Axis parameters read from and written to the H3 module, even motion commands are transmitted to the module via this FB.
FB inputs and outputs are divided into the following groups:

Inputs
- FB parameters
- Command flags
- Input parameter registers
- Data from H3 module

Outputs
- Status flag
- Output parameter registers
- Data to H3 module

FB parameters

The correct axis and data (command flags, status flags and parameter registers) is selected by supplying the axis number "Ai", the register block address "RAi" and the flag block address "FA1" when the FB is called.

Command flags

By setting a command flag from the user program, a specific command can be executed (e.g. load destination position, start motion etc.). The FB checks the command flags, and if a flag is set, the command is executed by calling the relevant sub-function block (which has the same name as the command flag, but without the letter "F"). After the command has been executed, the command flag is reset. If none of the command flags are set, the FB does nothing.
There are separate command flags for each axis. Chapter 7.1.5 shows how these flags are set from the user program.
To ensure that the H3 does nothing on start-up, the command flags are reset by the initialization FB "AxInit". See chapter 7.4 for details of each function.

Input parameter registers

The input parameter registers contain all the operating parameters. The relevant registers must be loaded by the user program before calling the FB to execute the commnand which copies the parameter from the register into the H3 module.

Status flags

Each time the FB is called, the status flags for that axis are copied into these flags. These flags can be polled by the user program. For a description of these flags, see the "FResSF" command.

Output parameter registers

Parameters read from the H3 module are copied into these registers.

H3 module

This refers to the controller for a particular axis within the H3 module.

Calling the function block:

Example: To call FB "AxHndlg" for axis 2

```
CFB     AxHndlg
        2           ; Axis number 2
        RA2         ; Register block for axis 2
        FA2         ; Flag block for axis 2
```

Program structure and application of function blocks "AxInit" and "AxHndlg"

The H3 module user program can be roughly divided into three parts:

- Initialisation
- Cyclic processing
- Controlling motion

a) Initialisation

The first step in an H3 Program is always the initialisation of the axes. Each axis must be initialized separately by calling FB "AxInit" from the start-up XOB 16. The FB reads pre-loaded values from the register block. Details about the drive must be known in order to determine what values to load into the registers before calling "AxInit", see chapter 9 "Application Examples" for details. The axis can be used after executing "AxInit".

b) Cyclic processing

All tasks which are executed regularly form part of the cyclic processing. They are therefore programmed into a COB, which is processed cyclically. The function block "AxHndlg" handles the entire data exchange between the program and the controller in the H3 module. The command flags inform the FB which tasks it has to execute, so the FB polls the command flags each time it is called, and executes any commands indicated. It is therefore natural to call the FB from a cyclic COB.

c) Controlling motion

Because motion control is always a sequential procedure, it is best represented by a GRAFTEC structured program.

In principle, the movements are always the same, and consist of the following steps:

- start the axis movement (--> Step)
- wait until the movement is completed (--> Transition)
- start the next movement
- wait for completion again
- and so on

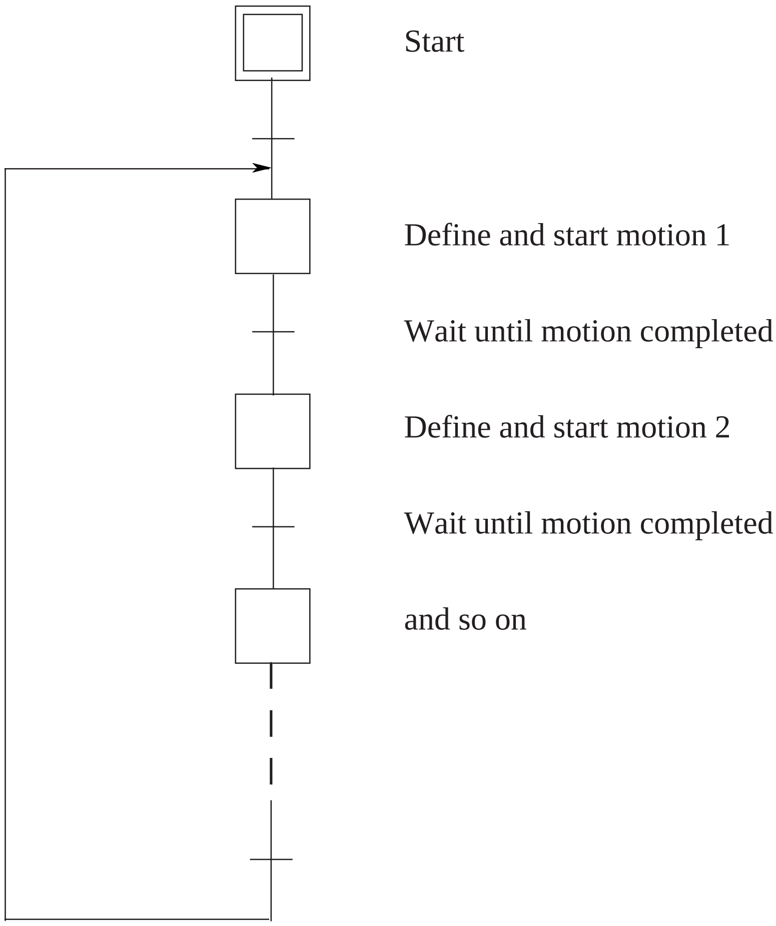
For every unfulfilled transition (while waiting for the axis to complete it's movement), the GRAFTEC program is exited and cyclic program execution continues.

--> The FB "AxHndlg" is called and fulfils the jobs which are defined by the GRAFTEC structure.
(Load motion parameters into H3 module and start the motion)

Example of H3 program structure:

```
XOB         16
  load initialisation registers here

  CFB       AxInit       ; Initialize
            1            ;  axis 1
            RA1
            FA1
            IMode
   :
   :
   :
  CFB       AxInit       ; Initialize
            n            ;  axis n
            RAn
            FAn
            IMode
EXOB
```

```
COB        0

CFB        AxHndlg     ; Process
           1           ; axis 1
           RA1
           FA1
  :
  :
  :
CFB        AxHndlg     ; Process
           n           ; axis n
           RAn
           FAn

CSB        0           ; Call motion control
                       ; (GRAFTEC)

now follow other tasks, such
as monitoring and display
output etc.

ECOB

SB         0
```




```
ESB
```

7.3. Command summary

The commands are executed by the axis handling FB "AxHndlg". The name given to each command indicates which flag is used to activate the command.

Operating parameters :

FSelOM	**Sel**ect **O**peration **M**ode
FSetPE	**Set P**osition **E**rror

Velocity profile parameters

FLdDA	**L**oa**d D**estination **A**bsolute
FLdDR	**L**oa**d D**estination **R**elative
FLdVA	**L**oa**d V**elocity **A**bsolute
FLdVR	**L**oa**d V**elocity **R**elative
FLdAA	**L**oa**d A**cceleration **A**bsolute
FLdAR	**L**oa**d A**cceleration **R**elative
FLdRP	**L**oa**d R**egulator **P**arameter
FUpDRP	**Up D**ate **R**egulator **P**arameter

Motion commands

FStart	**Start** motion
FStop	**Stop** motion
FMotOff	**Mot**or **off**
FSStepF	**S**ingle **Step F**orward
FSStepB	**S**ingle **Step B**ackwards
FForw	**Forw**ard with defined velocity
FBackw	**Backw**ards with defined velocity

Read data commands

FRdAP	**R**ea**d** **A**ctual **P**osition
FRdSP	**R**ea**d** **S**etpoint **P**osition
FRdAV	**R**ea**d** **A**ctual **V**elocity
FRdSV	**R**ea**d** **S**etpoint **V**elocity
FRdITS	**R**ea**d** **I**ntegration **T**erm **S**um
FRdIP	**R**ea**d** **I**ndex **P**osition
FRdSR	**R**ea**d** **S**ignal **R**egister

Miscellaneous commands

FResSF	**Res**et **S**tatus **F**lag
FLdBPA	**L**oa**d** **B**reak **P**osition **A**bsolute
FLdBPR	**L**oa**d** **B**reak **P**osition **R**elative
FSetIP	**Set** **I**ndex **P**osition
FSetZP	**Set** **Z**ero **P**osition

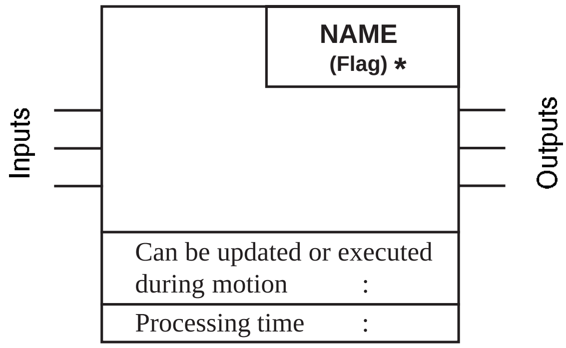
Notes :

7.4 Command description

For simplicity, all commands are described using the same format:

NAME Command : - text **NAME**

Software package : PCD9.H3E1



Functional description:

Inputs and outputs:

Other details ...

Commands on the following pages are in the same order as described in the summary of chapter 7.3.
An alphabetical list of all the names and symbols used can be found at the end of this manual.

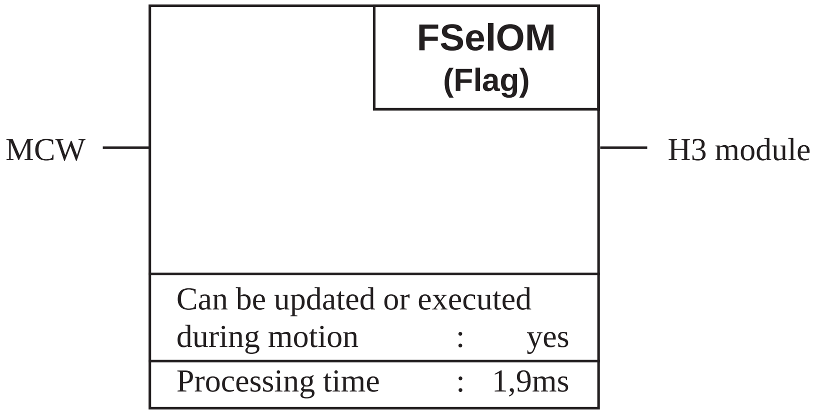
*) Indicates the symbol name for a flag, the command is executed by setting this flag.

FSelOM

Command : - **Sel**ect **O**peration **M**ode

FSelOM

Software package : PCD9.H3E1



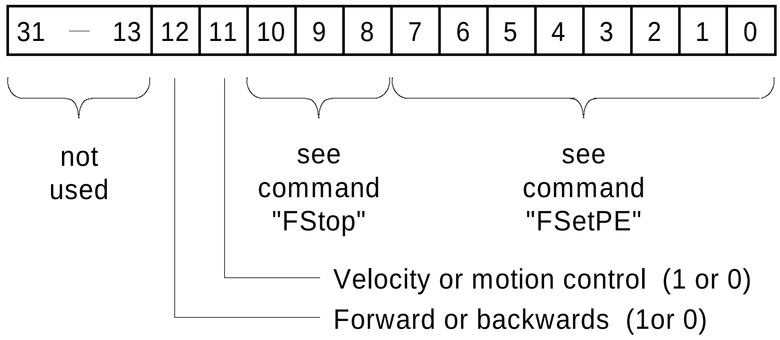
Functional description:

This command defines the operating mode for the axis (either motion or velocity control). The newly defined operating mode is used only after execution of the next start command "FStart".

Dscriptions of inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
MCW	**M**otion **C**ontrol **W**ord	no	R	Binary	see next page

Meaning of the motion control word "MCW":



The command reads only bits 11 and 12 of "MCW"

Bit 11: 0 --> Motion control
1 --> Velocity control

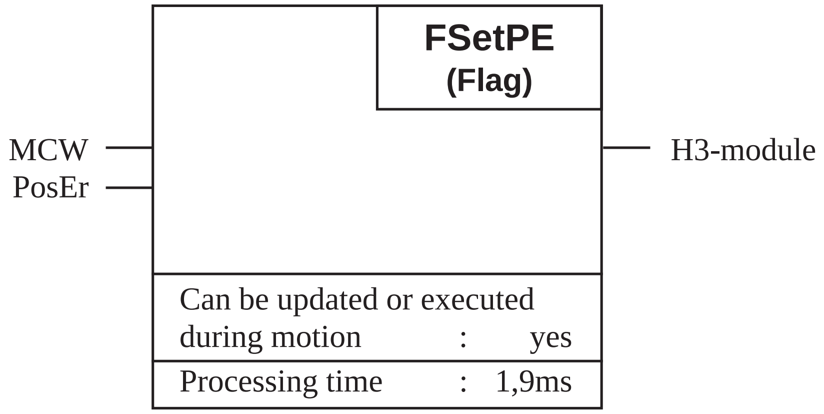
Bit 12: 0 --> backwards
1 --> forward

FSetPE

Command : - **Set P**osition **E**rror

FSetPE

Software package : PCD9.H3E1



Functional description:

This command loads the maximum allowable difference between the target and the actual positions. If the difference reaches this value, the status flag "ExcPEr" is set. Register "MCW" defines whether the status flag alone is set, or whether the controller is also switched off (output set to null).

A position error is a sign of serious problems and can therefore be monitored.

Description of inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
PosEr	**Pos**ition **Er**ror	no	R	Integer	0.. 32'767Imp.
MCW	**M**otion **C**ontrol **W**ord	no	R	Binary	see next page

Meaning of the motion control word "MCW":

31 — 13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used	see command "FSelOM"		see command "FStop"			1AH = status flag + regulator off 1BH = only status flag set							

The difference between set point and actual position is entered directly in encoder pulses. Note that in the position decoder, encoder pulses are quadrupled (by counting the pulse edges).

Example:

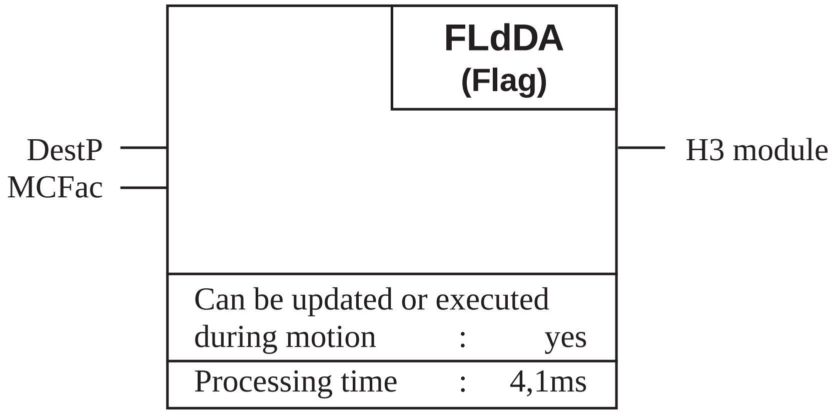
If the difference is to be a maximum of 500 encoder pulses, a value of 4*500 = 2000 steps must be loaded into register "PosEr".

FLdDA

Command : - **L**oa**d** **D**estination **A**bsolute

FLdDA

Software package : PCD9.H3E1



Functional description:

This command loads a new absolute destination position into the H3 module. "Absolute" means that the value is relative to position zero. The new position is used only when the next start command "FStart" is received.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
DestP	**Dest**ination **P**osition Value: $[-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	0.. $9{,}223371*10^{18}$

Maschine factor k in register "MCFac":

The k factor determines the units for entry of the destination position, velocity and acceleration. This factor is calculated from the encoder resolution and mechanical transmission. The k factor must be calculated and loaded into register "MCFac". This register is used by many commands to convert metric measurements into encoder pulses and vice versa.

The formula is: $k = \frac{4*In}{s}$

where In: pulses per revolution (encoder resolution)
s : distance per revolution (spindle gradient and gearing)

The unit for distance define the units for position, velocity and acceleration are determined at the same time as the units for distance.

Example:

Spindle with 3 mm gradient Encoder resolution 1000 pulses/rev.

A destination position of 60 mm should be approached and the input (and resolution) of the position should be given in µm.

$$k = \frac{4*In}{s} = \frac{4*1000 \text{ pul./rev.}}{3000 \text{ µm/rev.}} = 1{,}33333 \text{ pul./µm}$$

Input register "DestP" = 60000µm

Let the above example be used to give the position in units of 1/10mm:

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

Input register "DestP" = 600 1/10mm

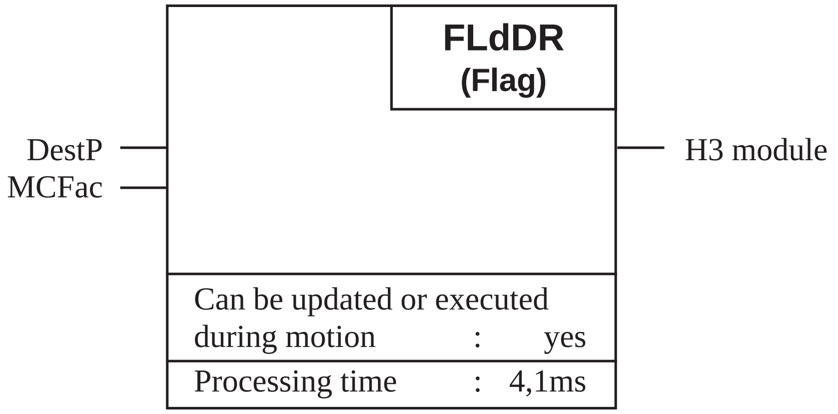
Note: To enter the position in pulses (fourway resolution of encoder partition), the value for "k" is 1.0 to be loaded in register "MCFac".

FLdDR

Function : - **Loa**d **D**estination **R**elative

FLdDR

Software package : PCD9.H3E1



Functional description:

This command loads the relative destination position into the H3 module. "Relative" means that the value refers to the current destination position. The new position is used only when the next start command "FStart" is received.

Description of the inputs and outputs:

Symbol	Description / Function	Parameters	Data Type	Format	Value
DestP	**Dest**ination **P**osition Value: $[-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

After initialization of the module, only an absolute destination position can be loaded for the first movement of the axis. If a relative position is loaded, the H3 controller produces a "command error".

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".

Example:

Spindle with 3mm gradient
Encoder resolution 1000 pulses/rev.

A relative distance of -60 mm should be covered and the input (and resolution) of the position should be given in µm.

$$k = \frac{4*In}{s} = \frac{4*1000 \text{ pul./rev.}}{3000 \text{ µm/rev.}} = 1{,}33333 \text{ pul./µm}$$

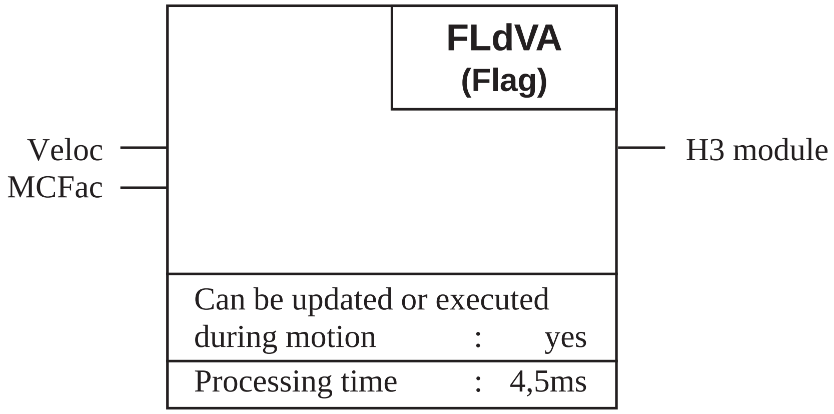
Input register "DestP" = -60'000 µm

FLdVA

Function : - **L**oa**d** **V**elocity **A**bsolute

FLdVA

Software package : PCD9.H3E1



Functional description:

This function is used to load an absolute velocity into an intermediate register in the H3 module. Absolute loading means that the value refers back to zero. The H3 module only takes the new velocity into the working register at the next start command.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
Veloc	**Veloc**ity Value: $[0..+(2^{30}-1)]/k*22348*10^{-6}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".

Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

Example:

Spindle with 3mm gradient
Encoder resolution 1000 pulses/rev.

A destination position should be approached at a velocity of 0.1 m/s, and the input (and resolution) should be given in mm/s.

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

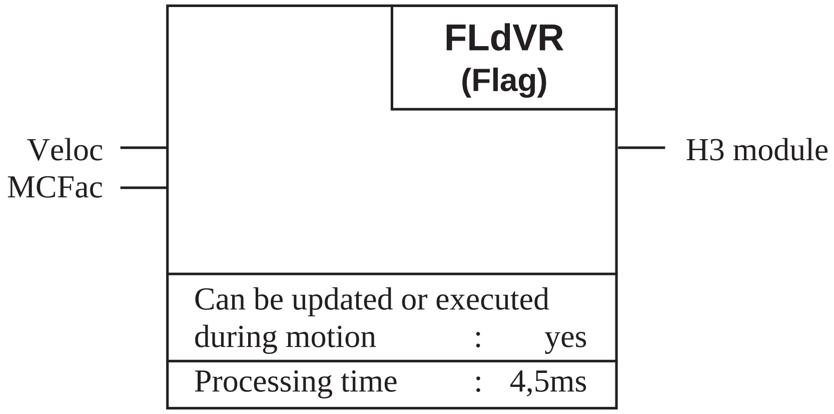
Input register "Veloc" = 100 mm/s

FLdVR

Function : - **Lo**a**d** **V**elocity **R**elative

FLdVR

Software package : PCD9.H3E1



Functional description:

This function is used to load a relative velocity into an intermediate register in the H3 module. Relative loading means that the value is relative to the current nominal velocity. The H3 module only takes the new velocity into the working register at the next start command.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
Veloc	**Veloc**ity Value: $[-2^{30}..+(2^{30}-1)]/k*22348*10^{-6}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

After initialisation of the module, only an absolute velocity may be loaded for the first motion of the axis. If a relative velocity is loaded, the H3 controller produces a "Command Error".

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".

Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

Example:

Spindle with 3mm gradient
Encoder resolution 1000 pulses/rev.

A relative velocity of -0.1 m/s is to be loaded and the input (and resolution) should be given in mm/s.

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

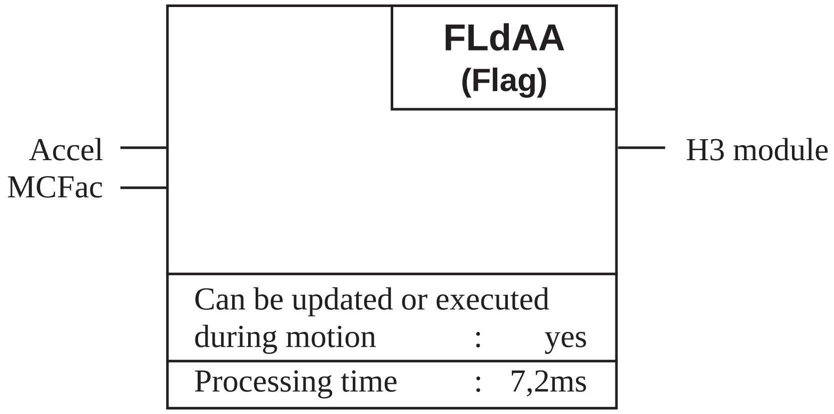
Input register "Veloc" = -100 mm/s

FLdAA

Function : - **L**oa**d** **A**cceleration **A**bsolute

FLdAA

Software package : PCD9.H3E1



Functional description:

This function is used to load an absolute acceleration into an intermediate register in the H3 module. Absolute loading means that the value refers back to zero. The H3 module only takes the new acceleration into the working register at the next start command. Although this function can be executed during a movement, the "FStart" start command to make the H3 controller operate with the new acceleration can only be executed after a completed motion (or as a result of a stop command).

NB: When this function is invoked, the "FMotOff" function is executed before loading the acceleration.

--> The regulator is switched off after the function has been executed.
--> Switch on the regulator again with "FStart".

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data Type	Data Format	Data Value
Accel	**Accel**eration Value: $[0..+(2^{30}-1)]/k*76206*10^{-9}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0.. 9,223371*10^{18}$

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".

Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

Example:

Spindle with 3mm gradient
Encoder resolution 1000 pulses/rev.

An acceleration of 0.005 m/s^2 is required and the input (and resolution) should be given in mm/s^2.

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

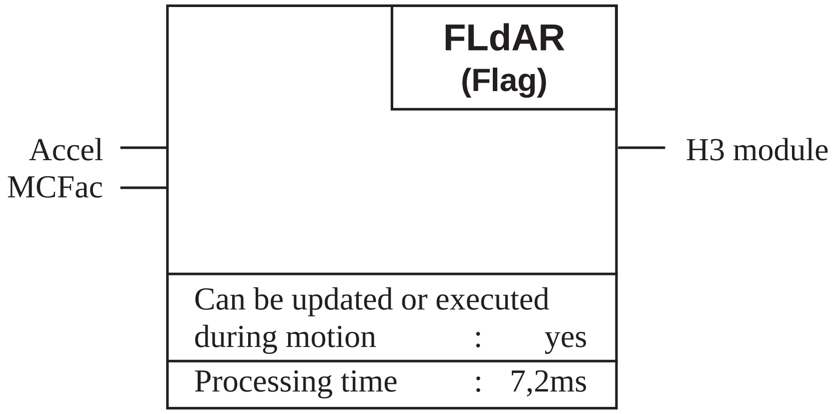
Input register "Accel" = 5 mm/s^2

FLdAR

Function : - **L**oa**d** **A**cceleration **R**elative

FLdAR

Software package : PCD9.H3E1



Functional description:

This function is used to load a relative acceleration into an intermediate register in the H3 module. Relative loading means that the value is relative to the current nominal acceleration. The H3 module only takes the new acceleration into the working register at the next start command. Although this function can be executed during a motion, the "FStart" start command to make the H3 controller operate with the new acceleration can only be executed after a completed motion (or as a result of a stop command).

NB: When this function is invoked, the command "FMotOff" is executed before loading the acceleration.

--> The regulator is switched off after this function has been executed.
--> Switch on the regulator again with "FStart".

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data Type	Data Format	Data Value
Accel	**Accel**eration Value: $[-2^{30}..+(2^{30}-1)]/k*76206*10^{-9}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0.. 9,223371*10^{18}$

After initialisation of the module, only an absolute acceleration may be loaded for the first motion of the axis. If a relative acceleration is loaded, the H3 controller produces a "Command Error".

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".

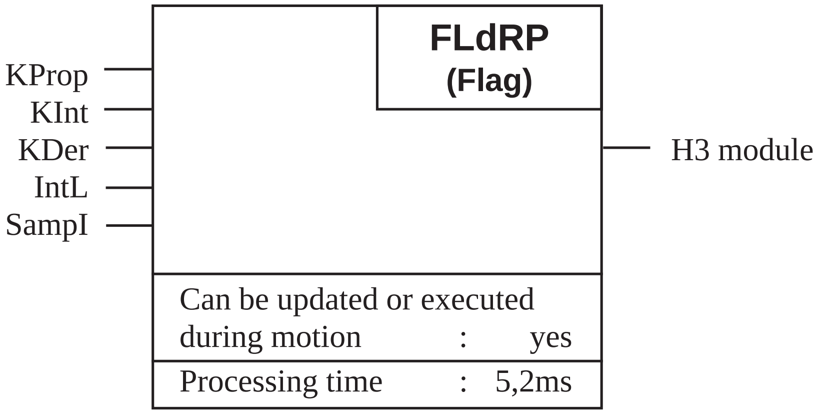
Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

FLdRP

Function : - **L**oad **R**egulator **P**arameter

FLdRP

Software package : PCD9.H3E1



Functional description:

This function is used to load the regulator parameters from the axis registers to intermediate registers in the H3 module. The regulator takes these values into the working registers once the function "FUpDRP" has been executed.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
KProp	**Prop**ortional Factor	no	R	Integer	0.. 32'767
KInt	**Int**egral Factor	no	R	Integer	0.. 32'767
KDer	**Der**ivative Factor	no	R	Integer	0.. 32'767
IntL	**Int**egration **L**imit	no	R	Integer	0.. 32'767
SampI	**Samp**ling **I**nterval derivative term	no	R	Integer	0.. 255

The derivative term sampling interval can be programmed in steps of 341.33 µs.

Sampling interval = (n+1) * 341,33 µs

The number n is loaded into register "SampI".

Example: Sampling interval should be 1024 µs

--> Register "SampI" = 2

The sampling interval for the proportional and integral terms is 341.33 µs and cannot be programmed.

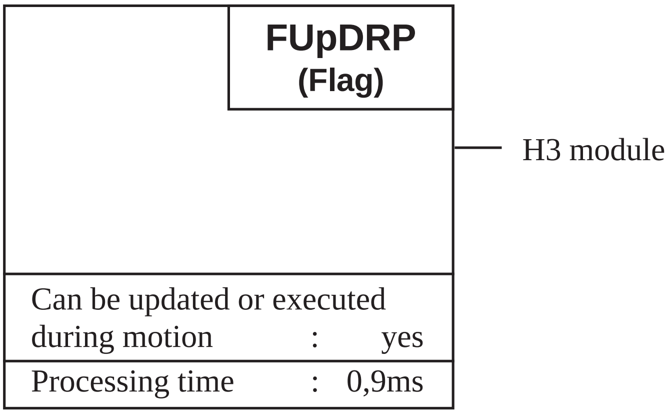
If a value outside the permitted range is loaded into the register, a command error is generated when the function is executed.

FUpDRP

Function : - **Up D**ate **R**egulator **P**arameter

FUpDRP

Software package : PCD9.H3E1



Functional description:

This function is used to update regulator parameters. The H3 module regulator takes the parameters loaded with function "FLdRP" from the intermediate registers into the working registers.

FStart

Function : - **Start** motion

FStart

Softwarepaket : PCD9.H3E1

FStart
(Flag)

H3 module

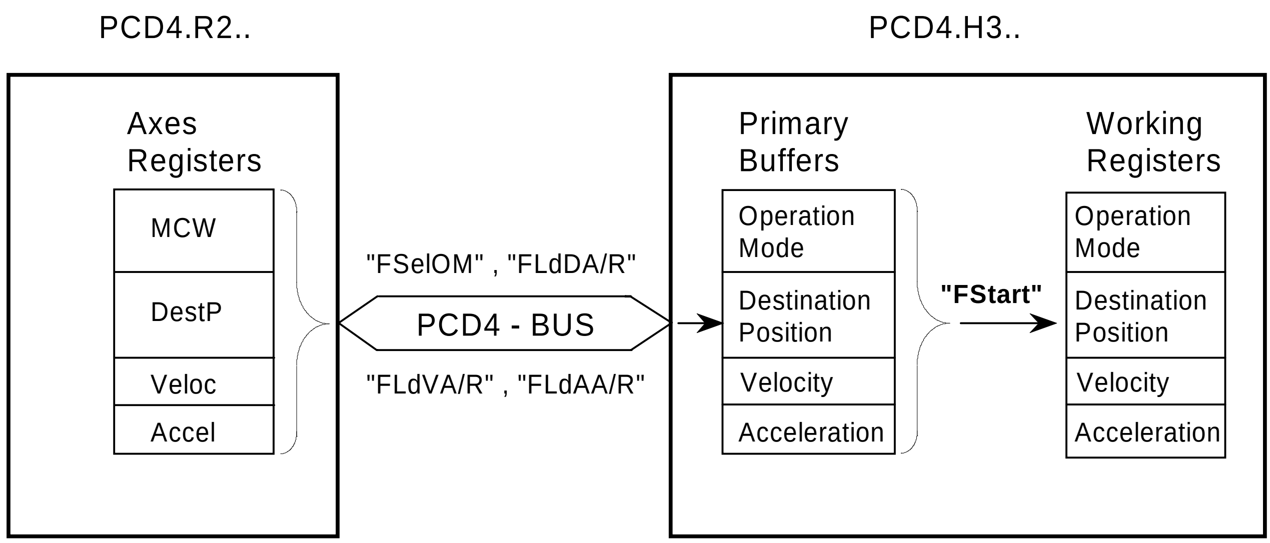
Can be updated or executed during motion	: yes
Processing time	: 1ms

Functional description:

This function can be used to start a motion, or to make the H3 module controller work with a newly loaded motion parameter (e.g. velocity).

If a new acceleration is loaded during a motion, a start command may only be executed after the motion has been completed.

The diagram below shows which motion parameters are only taken into the working registers of the H3 controller after a start command.

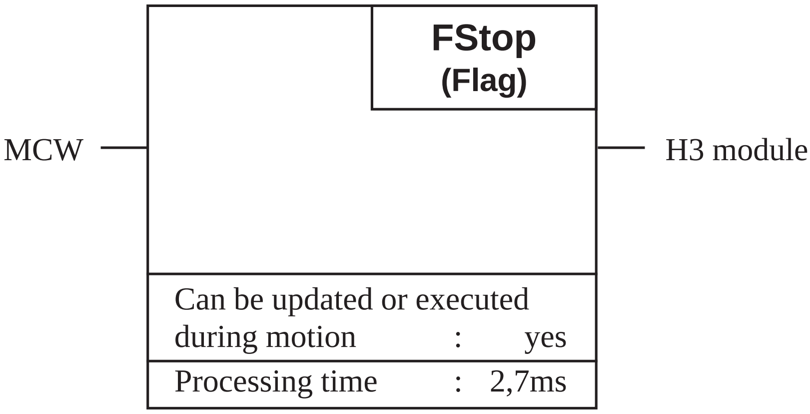


FStop

Function : - **Stop** motion

FStop

Software package : PCD9.H3E1



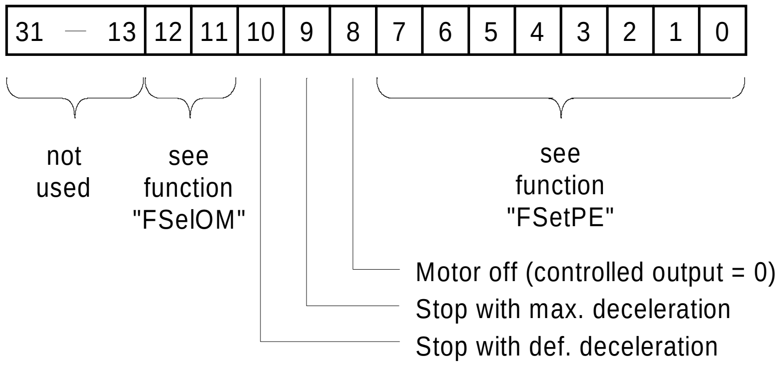
Functional description:

This function is used to stop a motion at any desired moment. The type of stop is in accordance with the definition in register "MCW". The "OnDest" status flag is set after execution of a stop.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
MCW	**M**otion **C**ontrol **W**ord	no	R	Binary	see next page

Meaning of the motion control word "MCW":



The function only reads bits 8 to 10 of register "MCW"

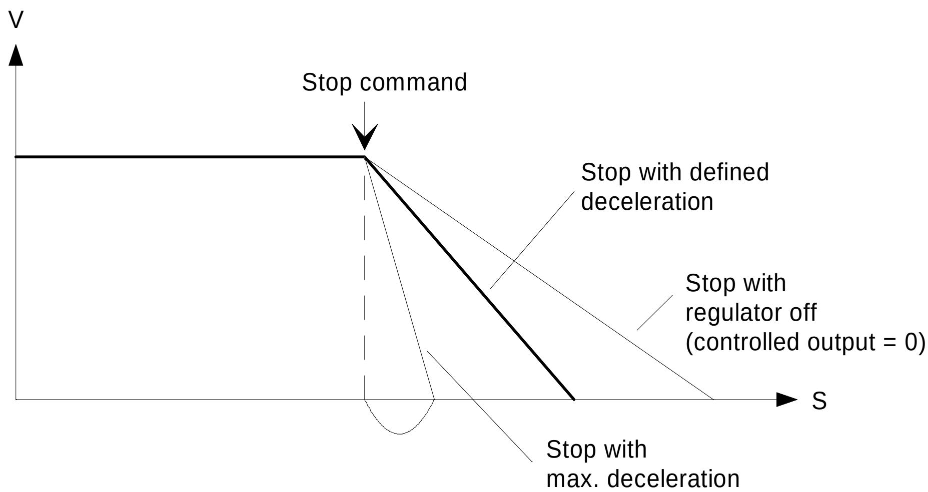
Bit 8 = "1" In case of a stop command, the regulator is switched off, i.e. the controlled output is set at zero.

Bit 9 = "1" In case of a stop command, maximum braking deceleration is used by settimg the setpoint position equal to the actual position in the working register of the H3 controller.

Bit 10 = "1" In case of a stop command, the defined braking deceleration (= negative acceleration) is used.

Only one of the three bits can be active ("1") at any time. After a stop, the controller does not lose the previously loaded destination position. However, before the interrupted motion can be continued without loading a new parameter (destination, velocity or acceleration), the operating mode must first be reloaded ("FSelOM").

The diagram below shows the three types of stop.



FMotOff

Function : - **Mot**or **Off**

FMotOff

Software package : PCD9.H3E1

FMotOff
(Flag)

H3 module

Can be updated or executed during motion	: yes
Processing time	: 1,8ms

Functional description:

This function is used to switch off the regulator, i.e. the controlled output is set at zero. The command "FMotOff" has the same function as the command "FStop" if Bit 8 in register "MCW" is active.

FSStepF

Function : - **S**ingle **Step F**orward

FSStepF

Software package : PCD9.H3E1

FSStepF (Flag)	
	H3 module
Can be updated or executed during motion : no	
Processing time : 4,5ms	

Functional description:

This function is used to move in a positive direction by a single pulse.
A destination position of +1 pulse is loaded relatively and the motion starts.
During execution, attention should be paid to the destination position value range. The function may not be executed if the absolute destination position has reached the positive limit of the value range.

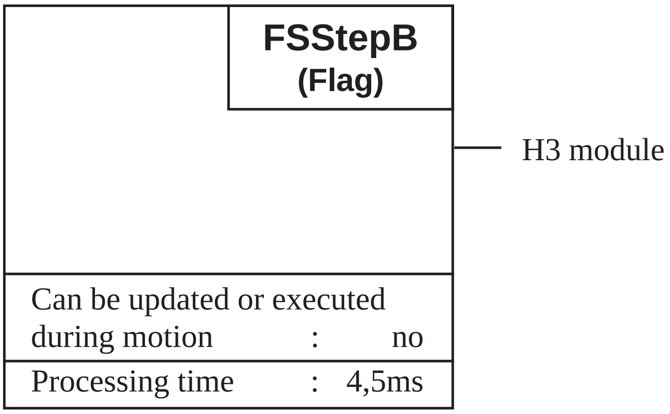
NB: 1 pulse corresponds to 1/4 encoder step
(pulse quadrupled by the position decoder).

FSStepB

Function : - **S**ingle **Step B**ackwards

FSStepB

Software package : PCD9.H3E1



Functional description:

This function is used to move in a negative direction by a single pulse.
A destination position of -1 pulse is loaded relatively and the motion starts.
During execution, attention should be paid to the destination position value range. The function may not be executed if the absolute destination position has reached the negative limit of the value range.

NB: 1 pulse corresponds to 1/4 encoder step
(pulse quadrupled by the position decoder).

FForw

Function : - **Forw**ard with defined velocity

FForw

Software package : PCD9.H3E1

FForw
(Flag)

H3 module

Can be updated or executed during motion	: yes
Processing time	: 4,5ms

Functional description:

This function is used to proceed in a positive direction at the previously loaded velocity. A manual stop command is required to stop the motion again.

The function is executed by loading the highest possible, positive destination position and then giving a start command.

FBackw

Function : - **Backw**ards with defined velocity

FBackw

Software package : PCD9.H3E1

FBackw
(Flag)

H3 module

Can be updated or executed during motion	: yes
Processing time	: 4,5ms

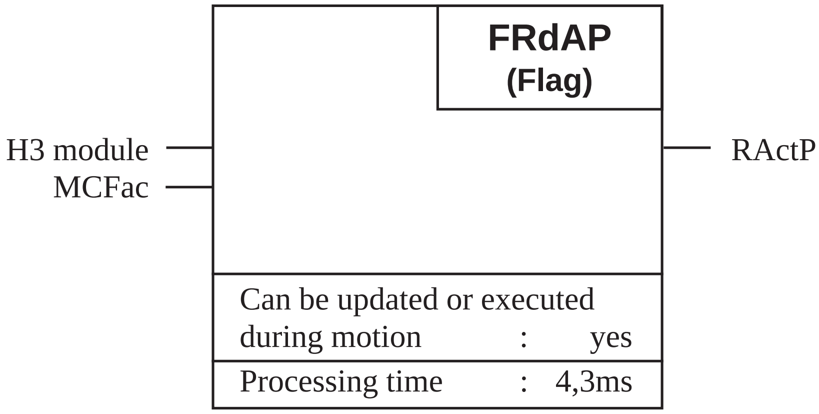
Functional description:

This function is used to proceed in a negative direction at the previously loaded velocity. A manual stop command is required to stop the motion again.

The function is executed by loading the lowest possible, negative destination position and then giving a start command.

FRdAP Function : - **R**ead **A**ctual **P**osition FRdAP

Software package : PCD9.H3E1



Functional description:

This function reads the actual position from the H3 module and copies it to register "RActP".

Description of the inputs and outputs:

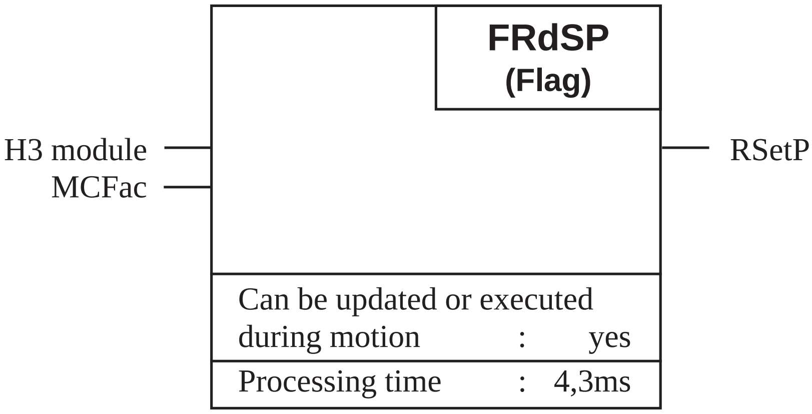
Symbol	Description / Function	Para-meters	Data Type	Format	Value
RActP	**R**egister **Act**ual **P**osition Value: $[-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9,223371*10^{18}$

Motion control factor k in register "MCFac":

Factor k has the same meaning as in function "FLdDA". Register "MCFac" is read by the above function and used to convert the position from a number of pulses to a metric measurement.

FRdSP Function : - **R**ea**d** **S**etpoint **P**osition FRdSP

Software package : PCD9.H3E1



Functional description:

This function reads the current setpoint position at the generator output for velocity profile and copies it to the register "RSetP". The difference between setpoint position and actual position is supplied to the PID regulator

Description of the inputs and outputs:

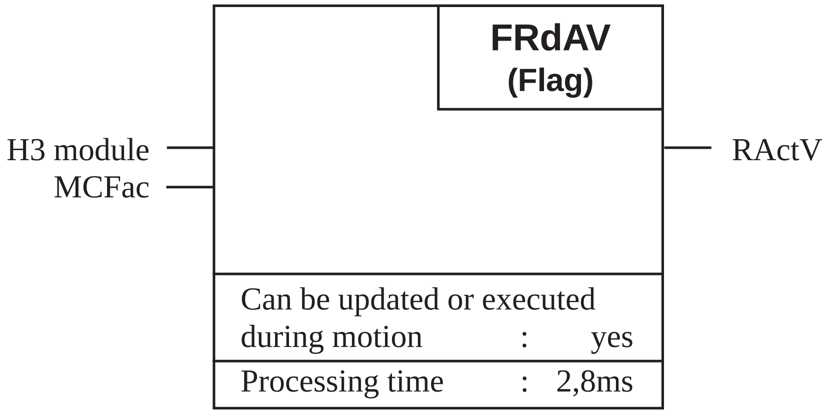
Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
RSetP	**R**egister **Set**point **P**osition Value: $[-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

Motion control factor k in register "MCFac":

Factor k has the same meaning as in function "FLdDA". Register "MCFac" is read by the above function and used to convert the position from a number of pulses to a metric measurement.

FRdAV Function : - **R**ea**d** **A**ctual **V**elocity FRdAV

Software package : PCD9.H3E1



Functional description:

This function reads the actual velocity of the axis from the H3 module and copies it to register "RActV".
However, only the 14 higher value bits can be read from the controller in the H3 module. For low velocities, therefore, no meaningful value can be read and it is recommended that the setpoint velocity is read instead of the actual velocity.

Description of the inputs and outputs:

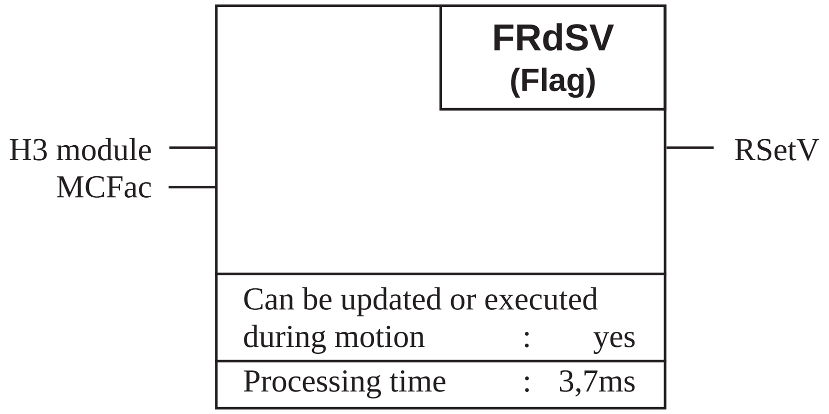
Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
RActV	**R**egister **Act**ual **V**elocity Value: $[0..+(2^{30}-1)]/k*22348*10^{-6}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9,223371*10^{18}$

Motion control factor k in register "MCFac":

Factor k has the same meaning as in function "FLdDA". Register "MCFac" is read by the above function and used to convert the velocity from a number of pulses/sec. to a metric measurement.

FRdSV Function : - **R**ea**d** **S**etpoint **V**elocity FRdSV

Software package : PCD9.H3E1



Functional description:

This function reads the current setpoint velocity from the profile generator and copies it to register "RSetV".

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data Type	Data Format	Data Value
RSetV	**R**egister **Set**point **V**elocity Value: $[0..+(2^{30}-1)]/k*22348*10^{-6}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

Motion control factor k in register "MCFac":

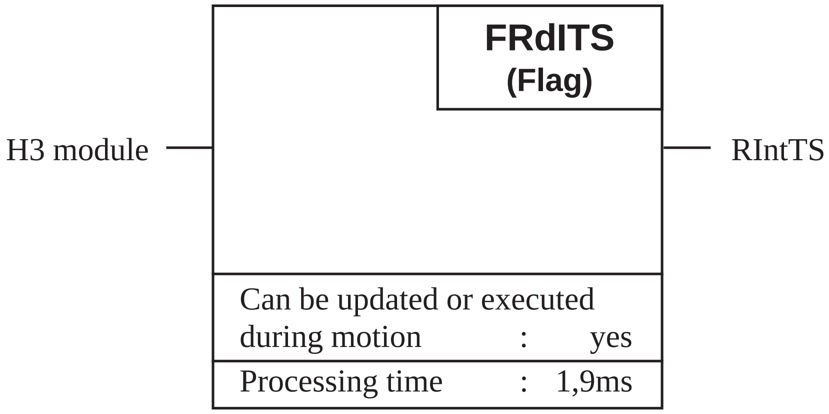
Factor k has the same meaning as in function "FLdDA". Register "MCFac" is read by the above function and used to convert the velocity from a number of pulses/sec. to a metric measurement.

FRdITS

Function : - **R**ea**d** **I**ntegration **T**erm **S**um

FRdITS

Software package : PCD9.H3E1



Functional description:

This function reads the integration value $\sum e(n)$ from the PID regulator and copies it to register "RIntS". The function is above all used for modulating regulator parameters during set up.

Description of the inputs and outputs:

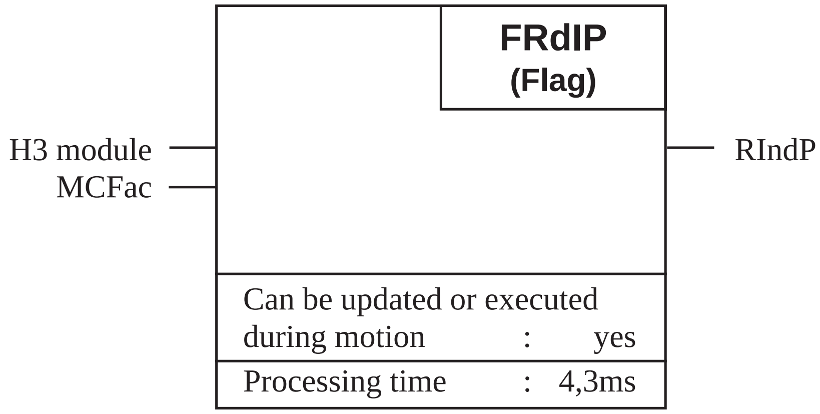
Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
RIntTS	**R**egister **Int**egration **T**erm **S**um Value: The value is inside the range of the integration term limit defined with function "FLdRP" (reg. "IntL")	no	R	Integer	--

FRdIP

Function : - **R**ea**d** **I**ndex **P**osition

FRdIP

Software package : PCD9.H3E1



Functional description:

This function reads the index position from the H3 module and copies it to register "RIndP" (see also function "FSetIP").

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
RIndP	**R**egister **Ind**ex **P**osition Value: $[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	0.. $9,223371*10^{18}$

Motion control factor k in register "MCFac":

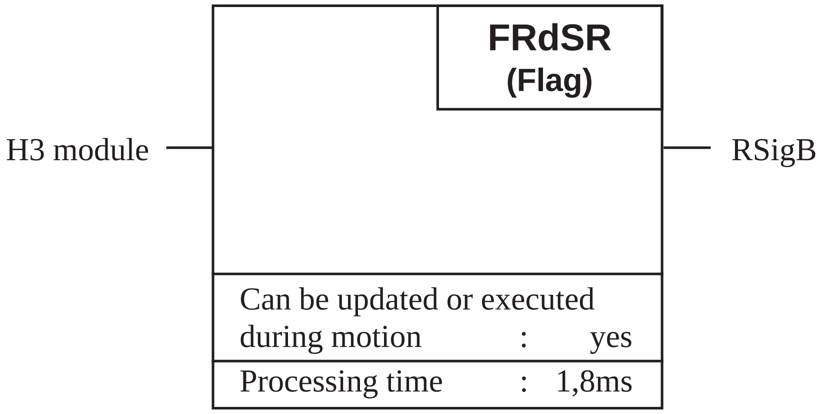
Factor k has the same meaning as in function "FLdDA". Register "MCFac" is read by the above function and used to convert the position from a number of pulses to a metric measurement.

FRdSR

Function : - **R**ea**d** **S**ignal **R**egister

FRdSR

Software package : PCD9.H3E1



Functional description:

This function is used to read the signal register of an axis from the H3 module.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
RSigB	**R**egister **Sig**nalisation **B**its	no	R	Integer	see next page

The individual bits in register "RSigB" have the following meaning:

Bit 0: Set to "1" after the function "SetIP" (set index position) has been executed. The bit is reset after entry of a subsequent index position (index pulse)

Bit 1: Not used

Bit 2 bis 6: Show condition of status flags (see function "FResSF")

Bit 7: Set to "1" if the regulator is switched off (controlled output = 0). The regulator is switched off by the following events:
- switching on the supply
- after processing FB "AxInit"
- if a position error has been exceeded (in case defined for this)
- executing function "MotOff" (motor off command)
- executing "FStop" function (if type of stop is defined for this)

The bit is reset by the next start command ("FStart")

Bit 8: Set to "1" when the supply is switched on or if the controlled output is defined as a PWM output by FB "AxInit". The bit is reset if the output is defined as a ±10V analogue output with FB "AxInit".

Bit 9: Shows the procedure defined if the maximum position error is exceeded.
"0" --> status flag "ExcEr" only is set
"1" --> status flag set and regulator switched off

Bit 10: Set to "1" when the generator has finished the calculated velocity profile. The bit is reset at the next start command ("FStart").

Bit 11: Shows the operating mode selected with function "FSelOM".
"0" --> position control
"1" --> speed control
The bit is only set and reset by a subsequent start command.

Bit 12: Shows the direction in speed control.
"0" --> forwards
"1" --> backwards
The bit is only set and reset by a subsequent start command.

Bit 13: Not used

Bit 14: Shows that a new velocity has been loaded with functions "FLdAA/R".
The bit is only reset by a subsequent start command.

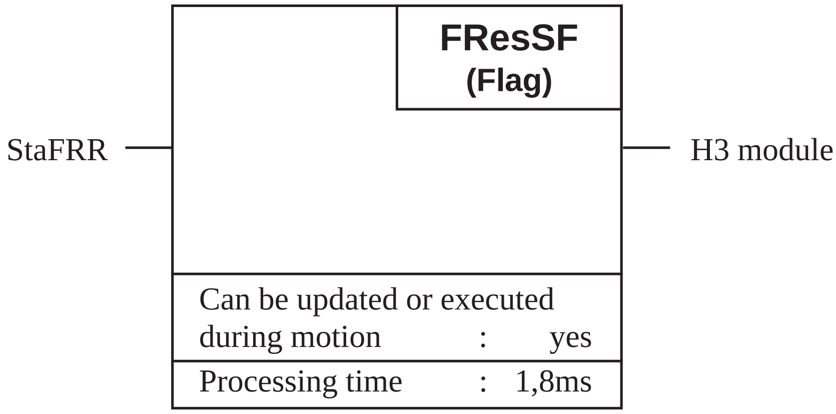
Bit 15 bis 31: Not used

FResSF

Function : - **Res**et **S**tatus **F**lag

FResSF

Software package : PCD9.H3E1



Functional description:

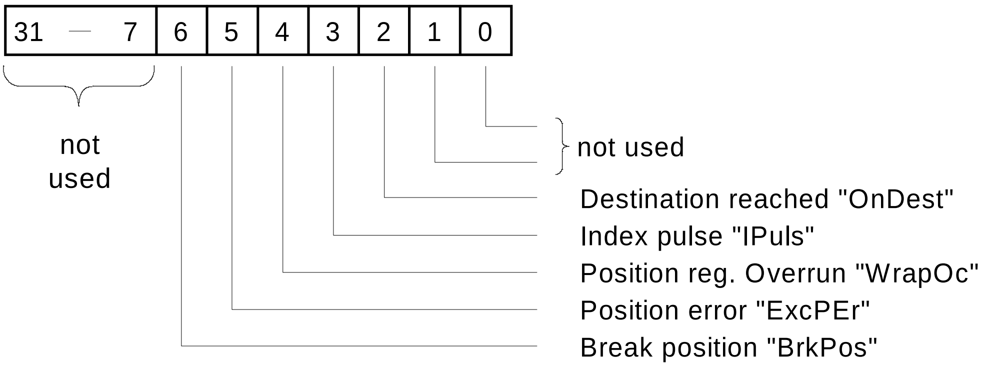
This function is used to reset the status flag of an axis.
The flags can be reset individually or all at the same time.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
StaFRR	**Sta**tus **F**lag **R**eset **R**egister	no	R	Binary	see next page

This function reads which status flags to reset from register "StaFRR".
The flag is reset if the corresponding bit in the register is zero.

Meaning of the register "StaFRR":



Meaning of the status flags:

"OnDest" **On Dest**ination (destination position reached)

Set to "1" by the following events:
- the generator has finished the calculated velocity profile. It can happen that, because of wrongly set parameters or mechanical problems, the flag is set even though the motor has not finally reached the destination position, as the generator has already ended the setpoint profile.
- the regulator is switched off (e.g. after the function "FMotOff")
- after a manual stop (function "FStop")

A start command (function "FStart") automatically resets the flag.

"IPuls" **I**ndex **Puls** noted

Set to "1" when an index pulse has been noted and the actual position written to the index position register in the H3 module (see also function "FSetIP").

"WrapOc" **Wrap** around **Oc**cured (position register overrun)

Set to "1" if the position register has been overrun.
An overrun is possible in speed control operation.

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

Set to "1" if the amount of error in positioning exceeds a value defined with function "FSetPE".

"BrkPos" **Br**ea**k** **Pos**ition

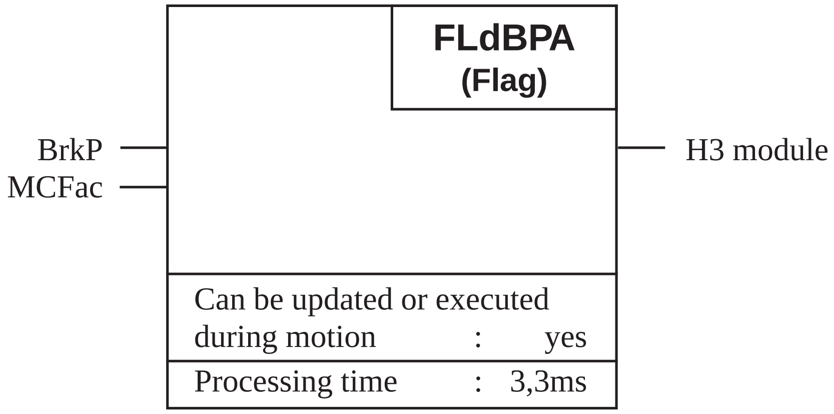
Set to "1" as soon as the actual position goes beyond the break position loaded with function "FLdBPA/R"

FLdBPA

Function : - **L**oa**d** **B**reak **P**osition **A**bsolute

FLdBPA

Software package : PCD9.H3E1



Functional description:

This function is used to load an absolute break position into the H3 module. Absolute loading means that the value refers back to zero. The H3 module takes the new position immediately into the working register. If the break position is reached, the status flag "BrkPos" is set. The flag can be reset with function "FResSF".

This function allows a message to be received at a particular position so that, for example, velocity or regulator parameters can be changed.

Description of the inputs and outputs:

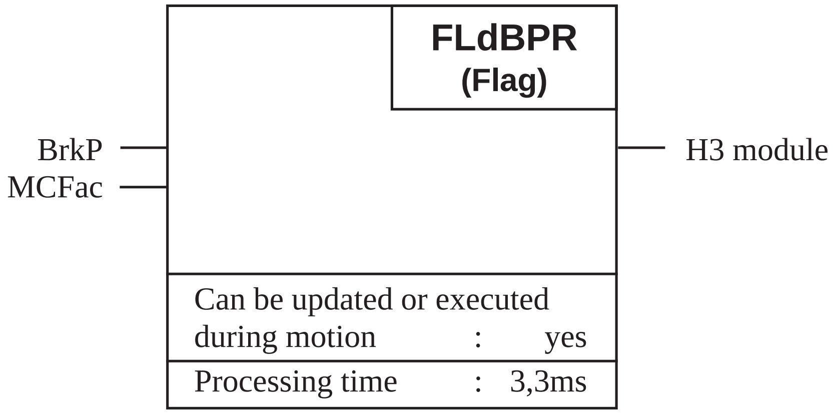
Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
BrkP	**Br**eak **P**osition Value: $[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA". Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

FLdBPR Function : - **L**oa**d** **B**reak **P**osition **R**elative FLdBPR

Software package : PCD9.H3E1



Functional description:

This function is used to load a relative break position into the H3 module. Relative loading means that the value refers to the current destination position. Care should be taken that the negative break position added to the destination position does not exceed the valid range of values for the destination position. The H3 module takes the new position immediately into the working register. If the break position is reached, the status flag "BrkPos" is set. The flag can be reset with function "FResSF".

This function allows a message to be received at a particular position so that, for example, velocity or regulator parameters can be changed.

Description of the inputs and outputs:

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
BrkP	**Br**ea**k** **P**osition Value: $[(-2^{30}..+(2^{30}-1)]/k*10^{-3}$ Unit: defined by k	no	R	Integer	--
MCFac	**M**otion **C**ontrol **Fac**tor	no	R	Fl.point	$0..\ 9{,}223371*10^{18}$

Motion control factor k in register "MCFac":

The k factor has the same meaning as for function "FLdDA".
Please note that this factor is read from the same register for destination position, velocity and acceleration. It is therefore recommended that the same units are selected for entering these parameters.

FSetIP

Function : - **S**et **I**ndex **P**osition

FSetIP

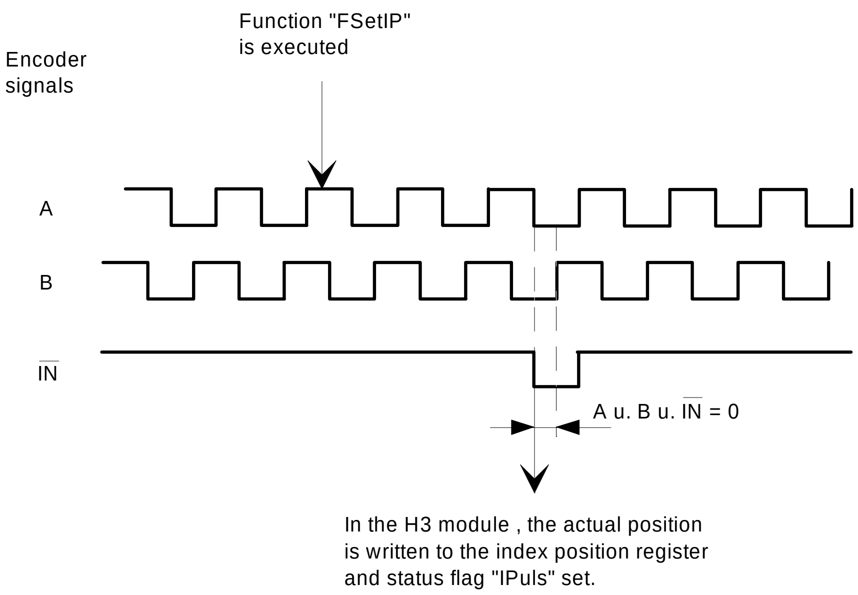
Software package : PCD9.H3E1

FSetIP (Flag)	
H3 module	
Can be updated or executed during motion	: yes
Processing time	: 0,9ms

Functional description:

Once this function has been executed, the next that encoder signals A, B and the index pulse input have status zero together, the actual position is written to the index position register in the H3 module. When the index position has been noted, the "IPuls" status flag is set. The index position can be read from the H3 module using the function "FRdIP".

The diagram below shows what happens in the H3 module regarding encoder signals once this function has been executed.



FSetZP

Function : - **Set** **Z**ero **P**osition

FSetZP

Software package : PCD9.H3E1

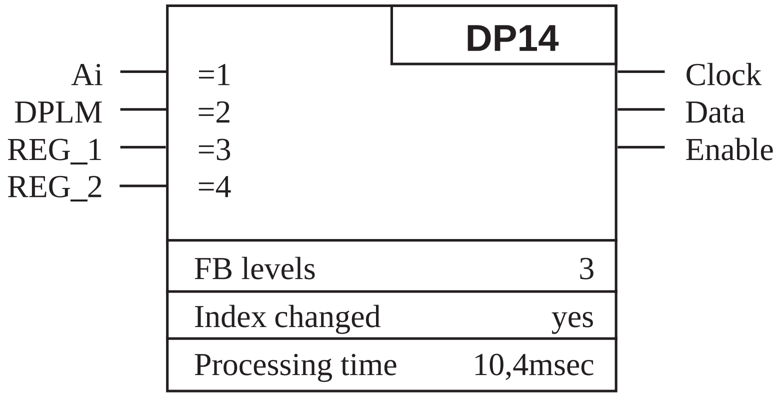
FSetZP (Flag)	
H3 module	
Can be updated or executed during motion	: yes
Processing time	: 0,9ms

Functional description:

This function defines the actual position as zero. If it is executed during a motion, the current destination is not affected, as long as no "FStart" start command is executed.

DP14 Function : - **D**isplay Contents of Register on PCA2.D**14** DP14

Software package : PCD9.H3E1



Functional description :

This command allows a register value of 1*10 digits or 2*6 digits to be displayed on the PCA2.D14 display module.
The following display formats are possible:

1*10 digits	b v 1 2 3 4	b = blank ; v = blank or "-"
(1 register)	5 6 7 8 9 10	1 .. 10 = digits

Display range = range of register values ± 2'147'483'647

2*6 digits	v x x x x x	w = contents of register	
(2 registers)	v x x x x x	0 ≤ w ≤ + 999'999	: v = MSD
		w > + 999'999	: v = "A"
		0 > w ≥ - 99'999	: v = " -"
		w < - 99'999	: v = "U"

Display range : - 99'999 ... + 999'999

The flag DPLM determines the type of display:

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

Parameters: REG_1 : Register for first display value
REG_2 : Register for second display value

Description of inputs and outputs :

Symbol	Description / Function	Para-meters	Data Type	Format	Value
Ai	Axis number i		K	Integer	1... 32
DPLM	Dislpay mode		F	Binary	0, 1
REG_1	first display value		R	Integer	0.. 10^9-1
REG_2	second display value		R	Integer	0.. 10^6-1
Clock	Output to D14		O	Binary	0, 1
Data	" "		O	Binary	0, 1
Enable	" "		O	Binary	0, 1

Program example to display actual position and setpoint velocity of axis 1 using a PCA2.D14:

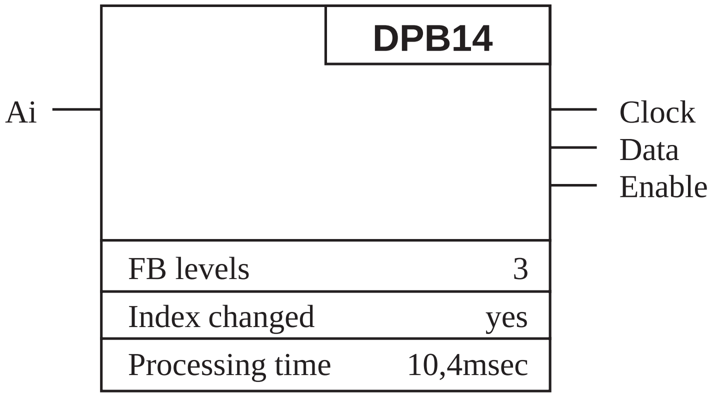
```
  ¦
ACC  H
SET  F    DplM+FA1  ; Display Mode = 2*6 Digits
CFB       DP14      ; Display on PCA2.D14
          1         ; Axis 1
     F    DplM+FA1  ; Display Mode
     R    RActP+RA1 ; Actual Position
     R    RSetV+RA1 ; Setpoint Position
  ¦
```

DPB14

Function : - Clear Display on PCA2.D14

DPB14

Software package : PCD9.H3E1



Functional description :

The "DPB14" command blanks over all characters on the PCA2.D14 display, deleting them.

Description of inputs and outputs :

Symbol	Description / Function	Para-meters	Data		
			Type	Format	Value
IR	Index register		K	Integer	$M_1, M_2 .. M_n$
Clock	Output to D14		O	Binary	0, 1
Data	" "		O	Binary	0, 1
Enable	" "		O	Binary	0, 1

8. Error recognition and handling

The H3 controller gives an error message by setting the command error flag. The flag is set if the controller cannot interpret received data or a command.

Usually the cause of a command error is a programming error on the part of the user, such as attempting to load a value into the H3 module which lies outside the permitted range of values.

Examples of errors:

- A negative velocity is loaded with function "FLdVA" (load absolute velocity).

- Motion control factor k was not loaded into register "MCFac" in floating point format. --> Can result in overrunning the value range when a parameter unit is calculated.

- A control value > 32767 is to be loaded into the H3 module.

- An attempt is made to load a relative motion parameter ("FLdDR", "FLdVR", "FLdAR" and "FLdBPR") for the first motion (before the first start command).

- A malfunction of the PCD4 bus.

Handling a command error

If an error occurs, the H3 controller ignores the command which has just been sent and sets the command error flag. This flag is independently monitored by all function blocks which communicate with the H3 module. The user does not have direct access to the error flag. If there is an error, the preceding command is repeated a maximum of two times. After the third successive error, the error handling FB "ComErS" is called. Once this has been processed, execution continues from the next program line. The FB "ComErS" is located in the file H3FB.SRC. The user can determine what actions are taken if an error occurs by placing his own code in this FB. In every case, if this FB is called, it means that a command has not been executed and that correct processing of the H3 program cannot be guaranteed. It is recommended that XOB 13 is called by forcing a divide-by-zero error.

Example:

```
FB          ComErS      ; Command Error Stop FB

DIV     R  0            ; Forces a divide-by-zero
        K  0            ; error, which calls XOB 13
        R  0            ;
        R  0            ;

EFB
```

If the instruction "DIAG" is programmed in XOB 13, it can very quickly be established where the error has occurred in the user program.

Example:

```
XOB 13

DIAG    R   1           ; Load Diagnostic Registers

Possibly switch off drive

HALT

EXOB
```

Procedure for locating an error

Establish which command triggered the error using the debugger and the contents of the Diagnostic Register.
Refer to the description of the PCD instruction "DIAG".

9. User examples for training

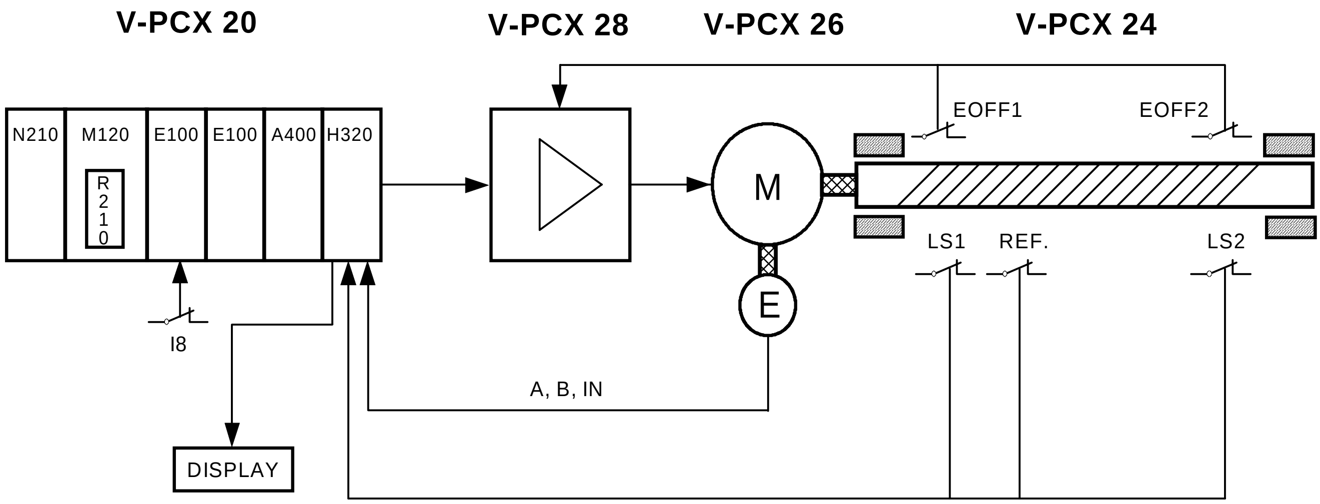
9.1 Example 1

The example concerns a very simple application with one axis. It is intended to show which steps must be made and in what sequence in order to run a simple motion.

Task

Hardware:

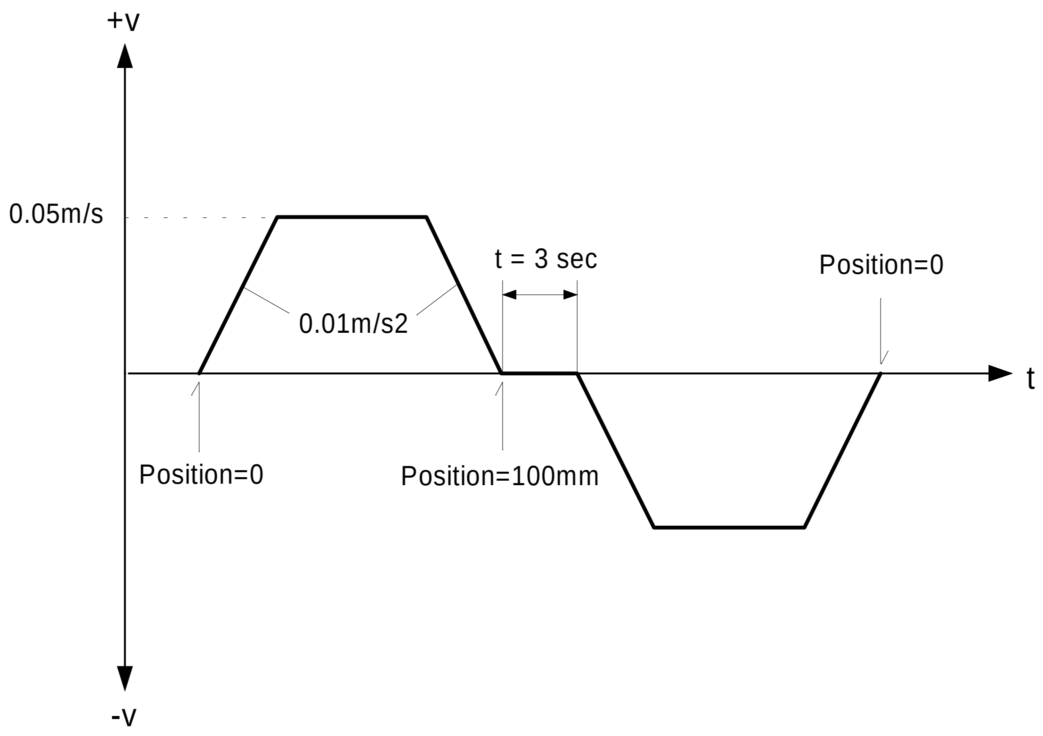
The example is based on hardware from the workshop models V-PCX 20, V-PCX 24, V-PCX 26 and V-PCX 28.



Axis data:

- Encoder 500 pul./rev.
- Spindle 2mm gradient
- Destination entered in 1/10mm resolution
- Maximum drive velocity 0.1m/s
- Maximum allowed acceleration 0.05m/s²

The motion should execute the following course:



The key connected to input 8 should be used to start the motion.

It can be accepted that the zero position has already been defined after switching on the controller and therefore that no reference course is necessary. This example should also do without any limit switch monitoring, as well as any display on the PCA2.D14 display module.

Solution

The hardware referred to in the summary of this task, as well as software package PCD9.H3E1, are available for it's solution. We can assume that all the hardware is installed and functioning. The steps described below should be taken into account when programming.

a) Software installation

The two H3 files H3DEF.SRC and H3FB.SRC are copied into the working directory. First of all, the H3 installation must be configured in file H3DEF.SRC as follows:

```
FMAH3   EQU   48          ; H3 module base address
IMode   EQU   6           ; initialize output port (analogue)
MNA     EQU   1           ; 1 axis set up
BAF     EQU   200         ; base address of flags
BAR     EQU   200         ; base address of registers
BAC     EQU   40          ; base address of counters
BAFB    EQU   900         ; base address of function blocks
RA1     EQU   0*NoRfeA    ; register block constant axis 1
FA1     EQU   0*NoFfeA    ; flag range constant axis 1
```

Since external symbol allocation is not being used, the symbol

```
PUBLSYM         EQU     0
```

is defined.

All the other symbols must remain unchanged.

In the H3FB.SRC file, symbol

```
EXTNSYM     EQU     0
```

is defined.

By means of this definition, the symbol definition file H3DEF.SRC is automatically included with the instruction $INCLUDE during assembly.
Including the definition file in the user file and H3FB.SRC file has the advantage that, immediately after assembly, the absolute addresses are available in the list file. These are required if a value must be shown with the debugger when testing the program.
On the other hand, if external symbol allocation was used, the absolute addresses would not be available until after the documentation file (.DOC) had been generated.

Located at the end of file H3FB.SRC is the FB "ComErS", which is called in case of a repeated command error. In this FB, the user can determine what measures should be taken in case of error. It is recommended that XOB 13 is called by forcing a divide-by-zero. The instruction "DIAG" can be programmed into XOB 13. In this way it can quickly be established where in the user program an error has occurred. For more details see Chapter 8.

```
FB      ComErS    ; Command Error Stop
DIV     R   0
        K   0     ; Force a divide-by-zero so
        R   0     ; that XOB 13 is called
        R   0
EFB
```

With the exception of these two changes, the file H3FB.SRC must not be changed.

In the next step the file is assembled to determine whether the changes mentioned have been carried out correctly. If assembly is successful, the file will not need to be re-assembled for the whole duration of the program and can later just be linked to the user program.

b) User program

To create the program, we refer to the program structure described in chapter 7.2:

1. Initialisation in XOB 16

In a first step, the values for the following initialisation registers must be determined:
(initialisation registers are read from FB "AxInit")

- Motion control word "MCW"

In this register the type of stop, operating mode and the measures in case of a position error are defined.

Type of stop: See also function "FStop"
In this first example, the type of stop is not important, as no manual stop command is allowed for in the program.
However, we define bit 10 = "1" -> stop with defined deceleration.

Operating mode: See also function "FSelOM"
We are operating in position control mode -> bits 11 and 12 = "0"

Position error: See also function "FSetPE"
In case of a position error, the status flag "ExcPEr" only should be set.
-> value for bits 0 to 7 = 1BH (hexadecimal)

Register "MCW"

Bit	31 ... 13	12	11	10	9	8	7 ... 0
	not used	0	0	1	0	0	1BH

-> Load register "MCW" with value 41BH (hexadecimal)

- Position error "PosEr" --> see function "FSetPE"

In this first example, the information for the position error is not important, as status flag "ExcPEr" is not monitored from the user program (for example, to switch off the drive). Since the position error must still be loaded into the H3 module during initialisation, and must be within the permitted value range, we define the maximum difference between a setpoint and an actual position as one revolution, which will set status flag "ExcPEr".

--> Load register "PosEr" with value 2000 = 4*500 pul./rev. (encoder pulse slope evaluation)

- PID factors --> see function "FLdRP"

For the sake of simplicity, the procedure for determining PID factors is shown with another example. We assume that the factors have already been given.

Registers must be loaded with the following values:

Proportional factor "KProp": 150

Integral factor "KInt": 50

Derivative factor "KDer": 50

Integration limit "IntL": 500

Sampling interval derivative term "SampI": 15 (5.46ms=16*0.341ms)

- Motion control factor k "MCFac": --> see function "FLdDA"

The motion control factor is used to determine the units of input and output values for position, velocity and acceleration. The task requires a resolution of 1/10 mm for entry of the destination position.

$$\longrightarrow k = \frac{4 * In}{s} = \frac{4 * 500 \text{ pul./rev.}}{20 * 1/10\text{mm/rev.}} = 100 \text{ pul. per } 1/10\text{mm}$$

—> Load register "MCFac" with value 100 (floating point format).

- Velocity "Veloc" —> see function “FLdVA”

Since for many applications only one velocity is often used, an absolute velocity is loaded at the initialisation stage.

For our example, let the velocity be 0.05 m/s.

—> Load register "Veloc" with value 500 (unit 1/10 mm/s)

- Acceleration "Accel" —> see function “FLdAA”

Acceleration for this task is 0.01 m/s^2

—> Load register "Accel" with value 100 (unit 1/10 mm/s^2)

The values which have now been entered must be loaded into the registers inside XOB 16 before calling FB "AxInit".

2. Cyclical axis handling

In COB 0 only FB "AxHndlg" is called for handling the axes.

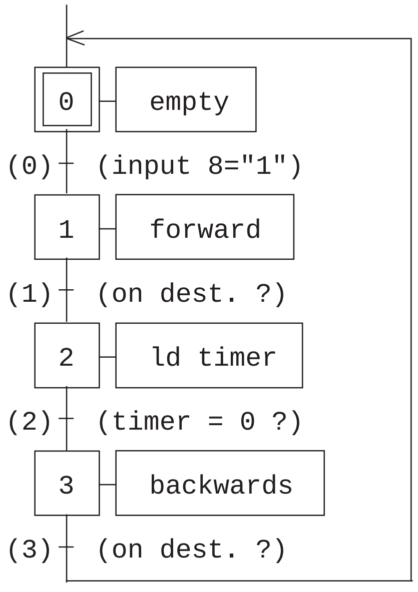
3. Definition of the motion control program

The motion control program is written according to the given velocity/ time profile within a GRAFTEC structure (SB 0).

The following pages show the source file (BSP01.SRC) of the user program for this example.

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

$ include H3DEF.SRC

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

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

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

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

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



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

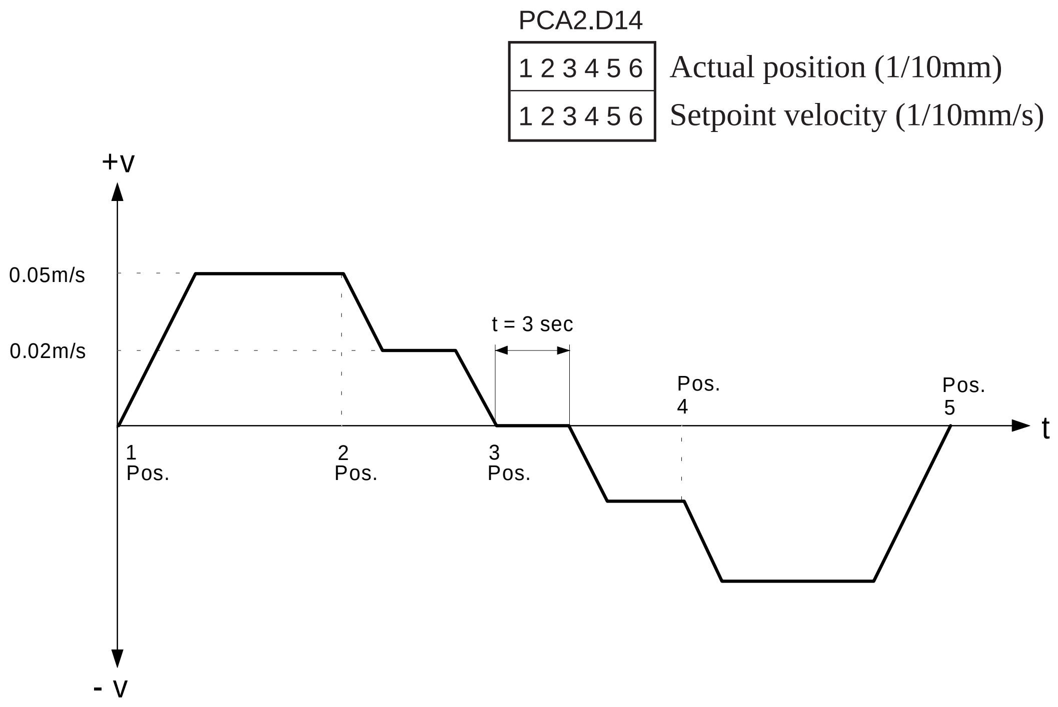
9.2 Example 2

The example builds on what has been learned in example 1.
It has been expanded with an additional change of velocity during the motion, and an output to display module PCA2.D14.

Task

Hardware: Workshop models as in example 1.

The motion should execute the following course:



Position 1:	0mm	(start position)
Position 2:	70mm	(break position)
Position 3:	100mm	(destination)
Position 4:	70mm	(break position)
Position 5:	0mm	(destination)

- Velocity for slow and fast backwards motion same as for forwards motion.
- Constant acceleration of 0.01 m/s2.
- The motion should be started with the key connected to input 8.
- The PCA2.D14 display should show the actual position above and the setpoint velocity below.

Solution

a) Software installation

We assume that hardware and software has already been installed as for example 1.
(Files H3DEF.SRC and H3FB.SRC can be taken in unchanged.)

b) User program

1. Initialisation in XOB 16

Can be taken from example 1 unchanged.

2. Cyclical axis handling

Further to example 1, functions "FRdAP" and "FRdSV" are used here to read the actual position and setpoint velocity from the H3 module, to be output to the PCA2.D14 display module with function block "DP14".

3. Definition of motion control program

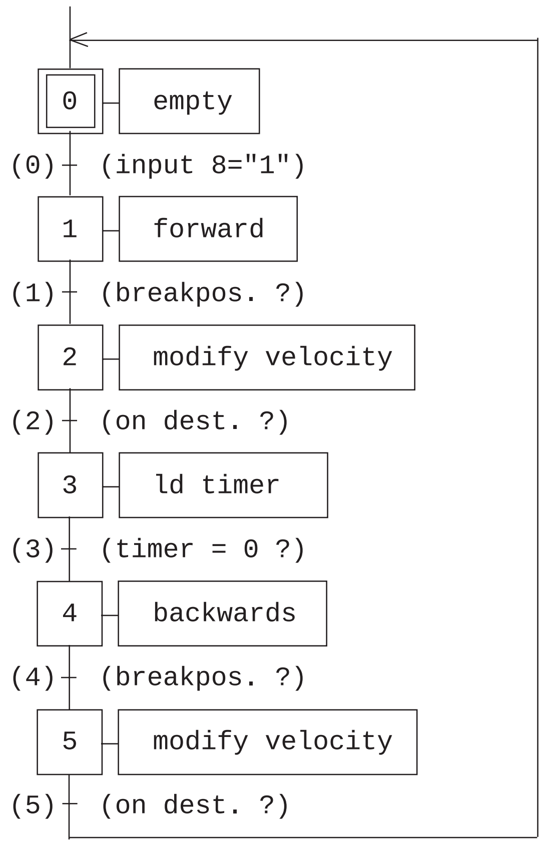
The motion control program is written according to the given velocity/ time profile within a GRAFTEC structure (SB 0). If the velocity is to be changed during the motion at a particular position, this can be achieved by loading a break position and polling the status flag "BrkPos". After loading the break position, care should be taken to reset the status flag "BrkPos" with function "FResSF".
See also the description of functions "FLdBPA" and "FResSF".

The following pages show the source file (BSP02.SRC) of the user program for this example.

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

$ include H3DEF.SRC

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

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

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

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

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



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

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

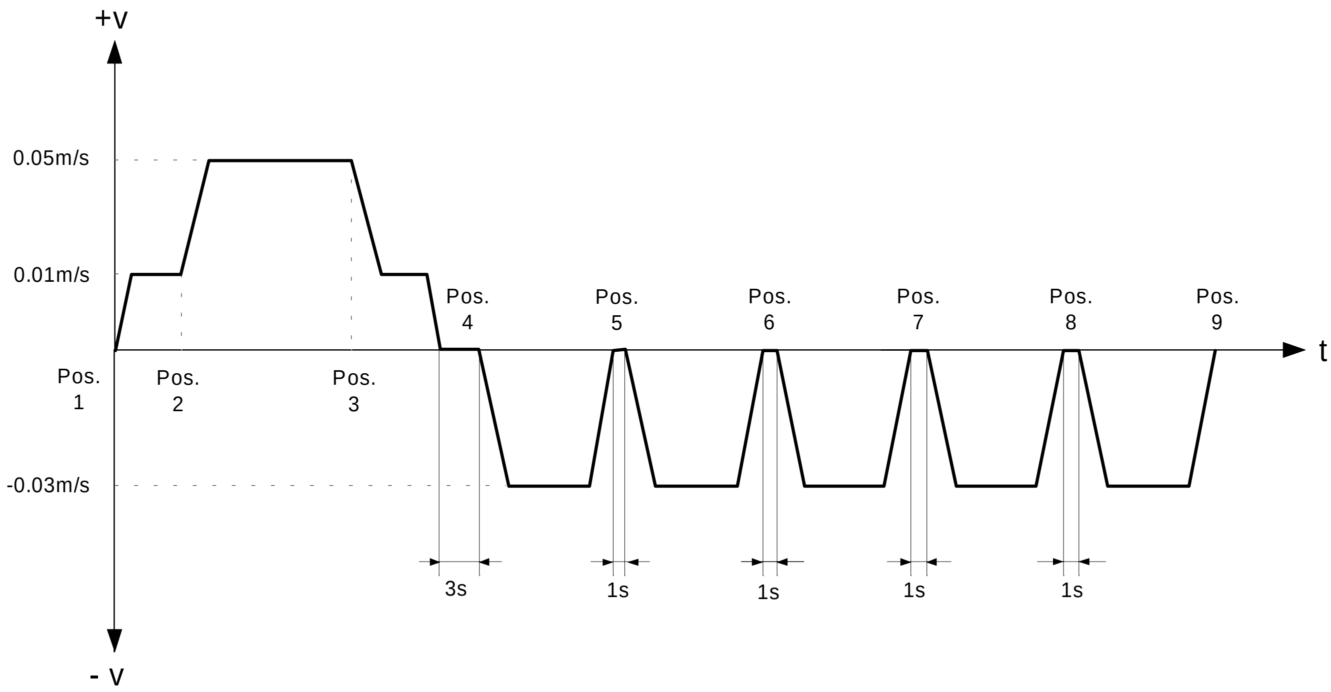
9.3 Example 3

The example builds on what has been learned in example 2.
It has been expanded with a more extensive motion control program with an automatic and a manual control term.

Task

Hardware: Workshop models as in example 1.

In automatic control, the motion should execute the following course:



Position 1:	0mm	(start position)
Position 2:	30mm	(break position
Position 3:	110mm	(break position)
Position 4:	150mm	(destination)
Position 5:	120mm	(destination)
Position 6:	90mm	(destination)
Position 7:	60mm	(destination)
Position 8:	30mm	(destination)
Position 9:	0mm	(destination)

Conditions:

- Constant acceleration of 0.01 m/s2.
- At any desired moment, it should be possible to switch between automatic and manual control.
- After a key depression has started the automatic program, it will run continuously until manual control is switched on, or until interrupted by the stop key.
- At any desired moment, it should be possible to stop the axis with a key depression.

Manual control:

- By means of a key depression, it should be possible to run in positive and negative directions at a velocity of +/- 0.02 m/s. As soon as the key is released, a stop of the defined type should ensue.
- The actual position should be definable as the zero position by means of a key depression.

Input allocation:

Input 0	-->	Switching to automatic/manual control (0/1)
Input 8	-->	Start automatic program
Input 9	-->	Forwards in manual control
Input 10	-->	Backwards in manual control
Input 11	-->	Define zero position
Input 15	-->	Stop

Solution

a) Software installation

We assume that hardware and software has already been installed as for example 2.
(Files H3DEF.SRC and H3FB.SRC can be taken in unchanged.)

b) User program

1. Initialisation in XOB 16

Can be taken from example 2 unchanged.

2. Cyclical axis handling

Further to example 2, control of operating mode is also achieved here. To allow switching between automatic and manual control at any desired moment, activating the switch at input 0 restarts the GRAFTEC program from step 0. In case of switching between automatic and manual control, this means that the current motion is continued until the drive stops.

If the stop key at input 15 is activated, the drive is stopped and afterwards the GRAFTEC program restarted from step 0.

3. Definition of the motion control program

The motion control program is written as required by the task in a GRAFTEC structure (SB 0).

Automatic program:

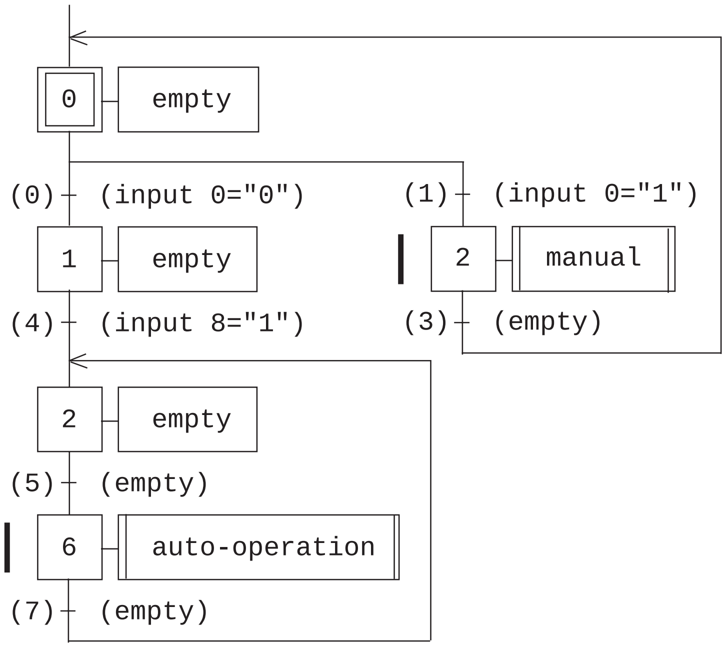
Since each time positions 5 to 9 are approached, the same course is run at the same velocity, these motions can be programmed in a loop (destination position loaded relative).

The following pages show the source file (BSP03.SRC) of the user program for this example.

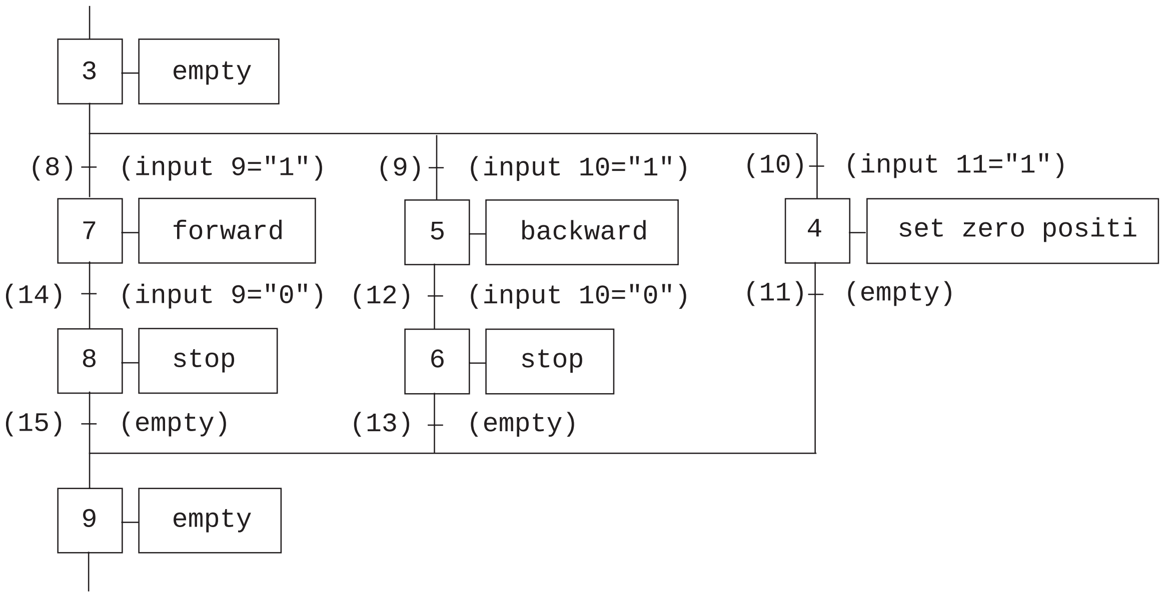
```
; Demo programm for the motion control module PCD4.H3..
; ======================================================
; Name  : BSP03.SRC
; U. Jäggi 27.08.90

$ include H3DEF.SRC

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

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

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

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

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

        CSB         0           ;Call of the motion control program

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

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

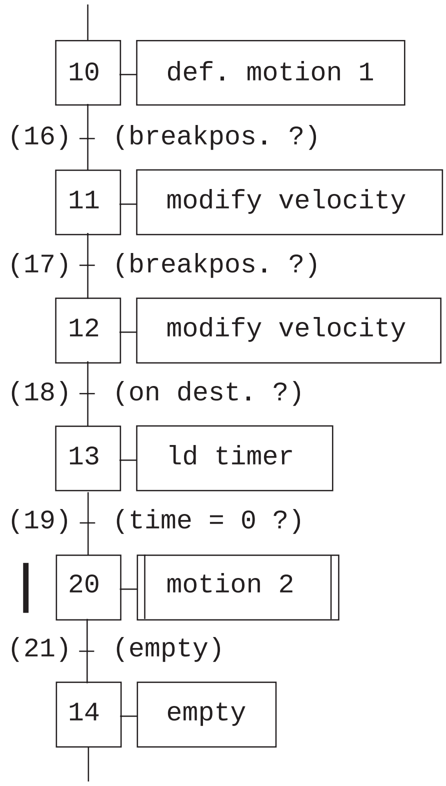
```
*** SAIA PCD GRAFTEC EDITOR $113 ***                               PAGE: 1
FILE: BSP03.GLS  (29.08.90  11.08)                  PRODUCED: 29.08.90  11.11
***   SAIA AG   -   CH-3280 MURTEN   ***
SB-NUMBER:  0
PAGE-NB:  0
```



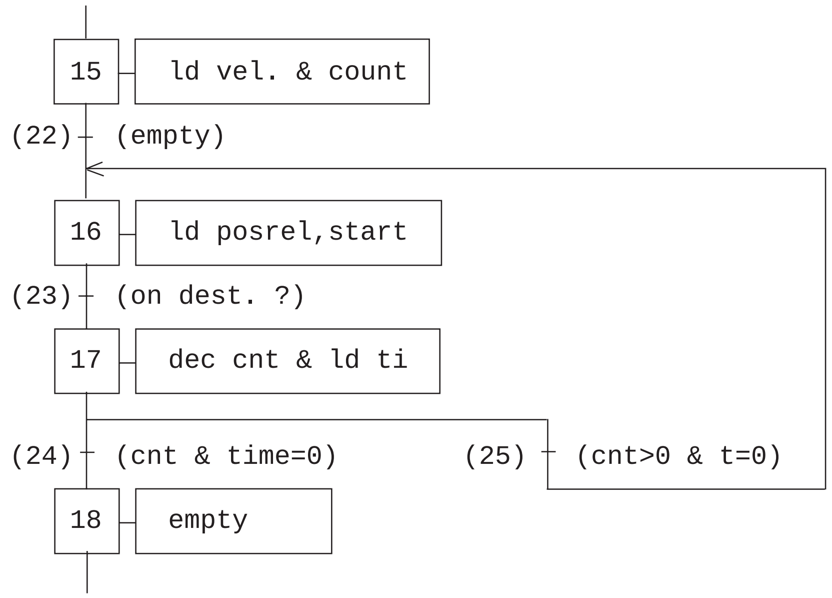
PAGE-NB: 2   manual



PAGE-NB: 6 auto-operation



PAGE-NB: 20 motion 2



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

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

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

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

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

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

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

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

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

ESB
```

10. Command and symbol summary

Operating parameters

FB Command Flags	Designation / Function	Processing Time	Can be updated or executed during motion	Page	Input / Output Values			
					Symbol	Type	Format	Designation / Function
FSelOM	Select Operation Mode	1,9ms	yes	7-24	MCW	R	Binary	Motion Control Word
FSetPE	Set Position Error	1,9ms	yes	7-25	PosEr MCW	R R	Integer Binary	Position Error Motion Control Word

Motion commands

FB Command Flags	Designation / Function	Processing Time	Can be updated or executed during motion	Page	Input / Output Values			
					Symbol	Type	Format	Designation / Function
FStart	Start Motion	1ms	yes	7-43	--			
FStop	Stop Motion	2,7ms	yes	7-44	MCW	R	Binary	Motion Control Word
FMotOff	Motor Off	1,8ms	yes	7-46	--			
FSStepF	Single Step Forward	4,5ms	no	7-47	--			
FSStepB	Single Step Backwards	4,5ms	no	7-48	--			
FForw	Forward with def. Velocity	4,5ms	yes	7-49	--			
FBackw	Backwards with def. Velocity	4,5ms	yes	7-50	--			

Velocity profile parameters

FB Command Flags	Designation / Function	Pro-cessing Time	Can be up-dated or exe-cuted during motion	Page	Input / Output Values			
					Symbol	Type	Format	Designation / Function
FLdDR	Load Destination Position Relative	4,1ms	yes	7-30	DestP MCFac	R R	Integer Fl. point	Destination Position Motion Control Factor
FLdDA	Load Destination Position Absolute	4,1ms	yes	7-28	DestP MCFac	R R	Integer Fl. point	Destination Position Motion Control Factor
FLdVR	Load Velocity Relative	4,5ms	yes	7-34	Veloc MCFac	R R	Integer Fl. point	Velocity Motion Control Factor
FLdVA	Load Velocity Absolute	4,5ms	yes	7-32	Veloc MCFac	R R	Integer Fl. point	Velocity Motion Control Factor
FLdAR	Load Acceleration Relative	7,2ms	conditionally	7-38	Accel MCFac	R R	Integer Fl. point	Acceleration Motion Control Factor
FLdAA	Load Acceleration Absolute	7,2ms	conditionally	7-36	Accel MCFac	R R	Integer Fl. point	Acceleration Motion Control Factor
FLdRP	Load Regulator Parameter	5,2ms	yes	7-40	KProp KInt KDer IntL Sampl	R R R R R	Integer Integer Integer Integer Integer	Proportional Factor Integral Factor Derivative Factor Integration Limit Sampling Interval Derivative Term
FUpDRP	Up Date Regulator Parameters	0,9ms	yes	7-42	--			

Read data commands

FB Command Flags	Designation / Function	Processing Time	Can be updated or executed during motion	Page	Input / Output Values			
					Symbol	Type	Format	Designation / Function
FRdAP	Read Actual Position	4,3ms	yes	7-51	RActP MCFac	R R	Integer Fl. point	Actual Position Motion Control Factor
FRdSP	Read Setpoint Position	4,3ms	yes	7-52	RSetP MCFac	R R	Integer Fl. point	Setpoint Position Motion Control Factor
FRdAV	Read Actual Velocity	2,8ms	yes	7-53	RActV MCFac	R R	Integer Fl. point	Actual Velocity Motion Control Factor
FRdSV	Read Setpoint Velocity	3,7ms	yes	7-54	RSetV MCFac	R R	Integer Fl. point	Setpoint Velocity Motion Control Factor
FRdITS	Read Integration Term Sum	1,9ms	yes	7-55	RIntTS	R	Integer	Integration Term Sum
FRdIP	Read Index Position	4,3ms	yes	7-56	RIndP MCFac	R R	Integer Fl. point	Index Position Motion Control Factor
FRdSR	Read Signal Register	1,8ms	yes	7-57	RSigB	R	Binary	Signalisation Bits

Miscellaneous commands

FB Command Flags	Designation / Function	Processing Time	Can be updated or executed during motion	Page	Input / Output Values			
					Symbol	Type	Format	Designation / Function
FResSF	Reset Status Flag	1,8ms	yes	7-60	StaFRR	R	Binary	Status Flag Reset Register
FLdBPR	Load Break Position Relative	3,3ms	yes	7-64	BrkP MCFac	R R	Integer Fl. point	Break Position Motion Control Factor
FLdBPA	Load Break Position Absolute	3,3ms	yes	7-62	BrkP MCFac	R R	Integer Fl. point	Break Position Motion Control Factor
FSetIP	Set Index Position	0,9ms	yes	7-66	--			
FSetZP	Set Zero Position	0,9ms	yes	7-67	--			

Notes :

From : Company : Department : Name : Address : Tel. : Date :	Send back to : SAIA-Burgess Electronics Ltd. Bahnhofstrasse 18 CH-3280 Murten (Switzerland) http://www.saia-burgess.com BA : Electronic Controllers Manual PCD4.H3xx Motion control modules for servo drives

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

Your suggestions :