SAIA® Process Control Devices

Motion control modules for servo drives

PCD2.H32x

SAIA-Burgess Controls Ltd. 2001. All rights reserved.
Edition 26/772 E1. 07.2001

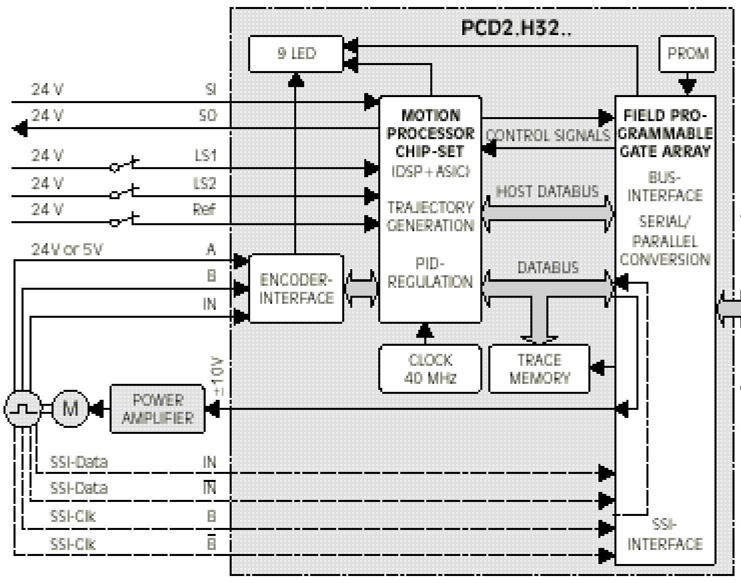
Subject to technical changes

TABLE OF CONTENTS

1. Introduction

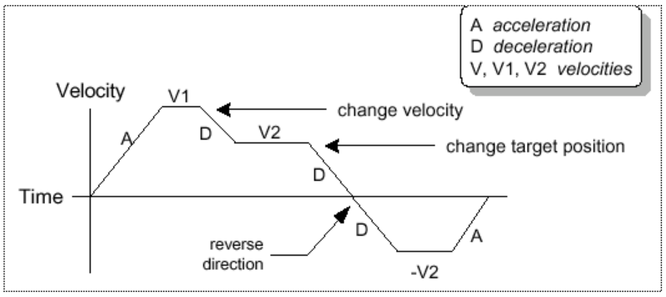
1.1 General

The PCD2.H32x. motion control module is an intelligent I/O module from the PCD2 series. This module can be used for positioning two independent axis with a variable speed control drive (servomotor) or two axis dependent of each other by a gearing factor. The servomotor can be any adjustable DC or AC motor having a power stage and incremental shaft encoder for the registration of position and speed. Position capture with an SSI absolute encoder is possible as well.

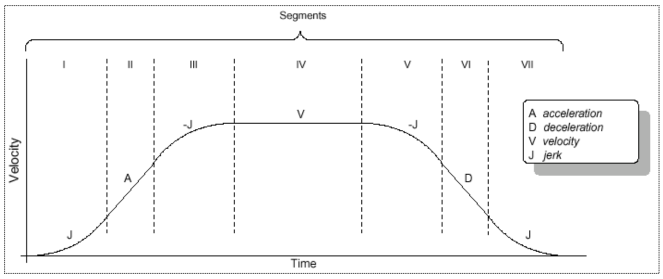


Block diagram of a servodrive for 1 axis (2 axes per module)

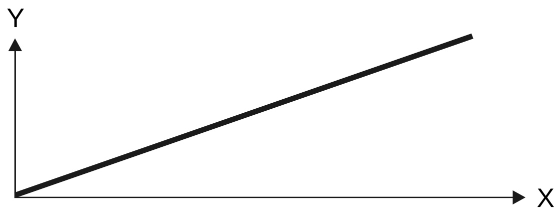
There are two ways generating trajectory profiles:
- trapezoidal
- S-Curve



Complex trapezoidal profile



S-curve profile



The gearing function allows synchronous operation of the 2 axes on-board.

4 different module types are available:

PCD2.H320	2 servo axis with 24V encoders
PCD2.H325	2 servo axis with 5V and SSI-absolute encoders
PCD2.H322	1 slave servo axis with 24V encoders
PCD2.H327	1 slave servo axis with 5V and SSI-absolute encoders

The PCD2.H322 and PCD2.H327 are special slave modules on which one axis (axis 2) is used as a master synchronisation input (without analog output).

1.2 Function and application

The H32x. module is used to position two axes with variable speed control DC or AC servomotors. This requires the drive unit to have a power stage and incremental shaft encoder for capturing position or speed. In a PCD1 system up to 2 motion control modules (4 axes) and in a PCD2 up to 4 H32x. modules (8 axes) can be plugged anywhere on the I/O bus and operated together with all base units. (With a PCD2.C1xx expansion unit up to 7 H32x modules (14 axes) can be used)

The motion control module contains a DSP processor that independently directs and “PID” controls every movement according to parameters supplied by the user program (velocity, acceleration and destination position). This enables each axis to execute independent moves, perform S-curve and trapezoidal motion profiles, change velocity and acceleration on the fly, perform flexible breakpoint functions and trace the axis during motion. Additionally the module is able to interpolate the two axes linearly.

Whenever possible, PCD2.H32x motion control modules should be plugged into the PCD2.M1xx base unit and not the PCD2.C1xx expansion unit. This is due to the relatively high current consumption.

1.3 Main characteristics

- Ideal for compact or economical machines
- PID controlled regulation of servomotor position and speed
- Velocity, acceleration and destination position can be modified during motion
- Analogue ±10V output with 12 bits (including sign bit) for triggering the motor power stage
- Per axis encoder signal inputs of 24 VDC (source operation) or 5 V/RS 422 (antivalent line driver)
- SSI-Interface for absolute position caption
- Per axis 2 digital inputs for limit switches
- Per axis 1 digital input for reference switch
- Per axis 1 digital input for synchronisation
- Per axis 1 digital output for triggering external events.

1.4 Typical areas of use

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

1.5 Programming

The availability of practical function blocks (FBs) means that only the desired parameters need to be entered for the various motion and travel functions. The present detailed manual contains descriptions of each function block, complemented with practical application examples. Programming is via the standard PG4 or PG5 programming tool from version V1.0 in IL (Instruction List).

A powerful commissioning tool is available.

Initialization instruction

INIT	FB, when it is called, 20 parameters are given with it.
	See section 7 and appendix A.

Execution instruction

EXEC	FB, always including 3 parameters. Over 50 different instructions can be executed.
	See section 7 and appendix A.

Home instruction

HOME	FB, for automatically seeking the reference switch. This FB has 7 parameters.
	See section 7 and appendix A.

Breakpoint instructions

SetBrkPoint	FB, to set and handle up to two individual breakpoints per axis. This FB has 6 parameters and several instructions to be executed. See section 7 and appendix A.

Diagnostics and error handling

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

See chapter 8.

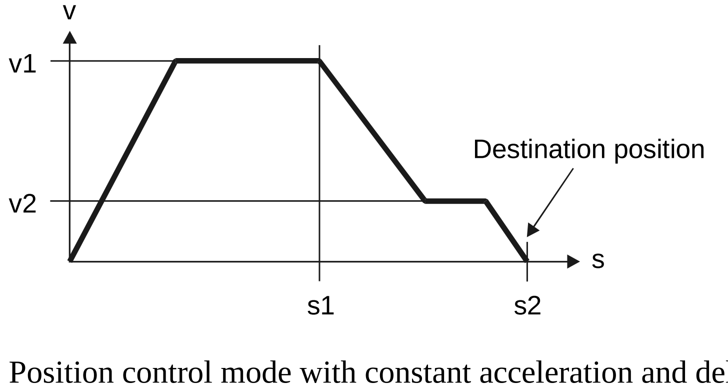
1.6 Operating modes

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

- position control mode
- speed control mode

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

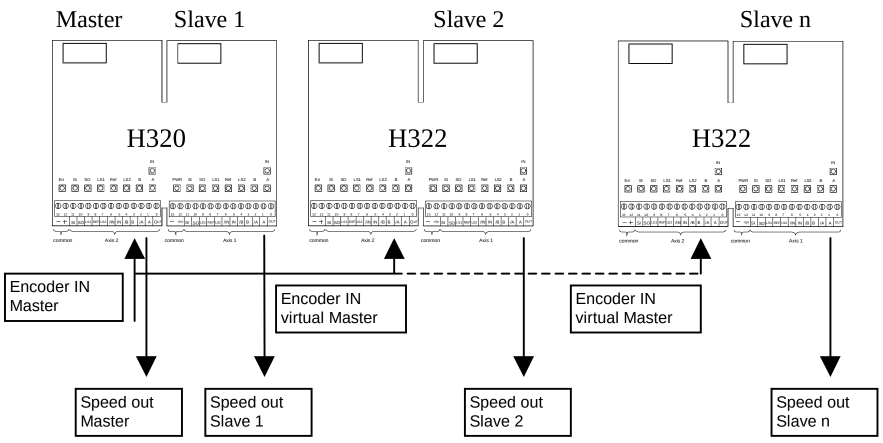
In speed control mode the desired velocity is attained with a preset rate of acceleration. Travel continues at this velocity in a controlled manner until there is a stop instruction. The desired velocity can be changed even during motion.



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

Slave modules PCD2.H322 and PCD2.H327

The PCD2.H322 and PCD2.H327 are special slave modules where axis 2 is used as a master synchronisation input and does not have an analog output.



The input filter of the synchronisation input (axis 2) has a higher impedance than the standard input filter (axis 1). This is necessary so that with the max. number of 6 slave modules the max. output current of the master encoder will not be exceeded.

The PCD2.H327 can be used with an SSI encoder on the slave axis 1, but the master encoder on axis 2 must be an incremental type.

When using H322 and H327 as slave modules the analogue output is on axis 2.

1.7 Commissioning

Convenient commissioning software is available as an Add-on tool for PG5 or as a stand-alone tool.

See chapter 10 for more details.

Notes

2. Technical Data

2.1 Hardware technical data

Internal power consumption (without encoder)

+5 V	typically 210 mA, max. 230 mA (250 mA in SSI mode)
V+	15...20 mA

External supply (Uext) and consumption
used to supply digital outputs SO

Terminals +/-	24 V (6 ... 32 V) smoothed, residual ripple max. 10% max. permissible ripple: 10%
Load	- 0…2 mA (without load current) - max. 1A for outputs
Over-voltage protection	TVS-diode 39V+/-10%
Wrong polarisation	not protected

Encoder inputs of the PCD2.H320/322 module

	Axis 1	Axis 2
Inputs	A B IN	A B IN
Number of inputs	3	3
Input voltage typ.	24 V	24 V
"Low" range	-30... +5 V	-30... +5 V
"High" range	+15 ... +32 V	+15 ... +32 V
Input current typ. H320 H322	7 mA 7mA	7 mA 2mA
Source operation (pos. logic)	x	x
F_{max}	125kHz 1)	125kHz 1)

1) 500kHz internal counting frequency

Encoder inputs of the PCD2.H325/327 module

	Axis 1	Axis 2
Inputs	A,/A B,/B IN,/IN	A,/A B,/B IN,/IN
Number of inputs	6	6
Input voltage typ.	RS422	RS422
Input impedance H325 H327	150Ohm 150Ohm	150Ohm 1500Ohm
F_{max}	250kHz 1)	250kHz 1)

1) 1MHz internal counting frequency

Digital inputs of all PCD2.H32x module per axis

Number of inputs:	1 reference input REF 2 Limit Switches LS1, LS2 1 Synch.Input SI
Input voltage:	typically 24 V
"Low" range:	-30... +5 V
"High" range:	+15 ... +32 V
Input current at 24 VDC:	7 mA (typical)
Circuit type	no galvanic separation
Input delay	300 µs

For safety reasons normally closed contacts or normally closed PNP sensors should be used for the limit and reference switches. Therefore these inputs work with negative logic. (LED is on, if 0V on input). Input SI works with positive logic.

Analoge outputs of the PCD2.H320/325 modules

	Axis 1	**Axis 2**
Outputs	Out	Out
Resolution (with sign bit)	12 bit	12 bit
Short-circuit protection	yes	yes
Galvanic isolation	no	no
Output voltage swing *)	± 10 V	± 10 V
Minimum load impedance	3 kΩ	3 kΩ

Analoge outputs of the PCD2.H322/327 modules

	Axis 1	**Axis 2**
Outputs	Out	n.c.
Resolution (with sign bit)	12 bit	-
Short-circuit protection	yes	-
Galvanic isolation	no	-
Output voltage swing *)	± 10 V	-
Minimum load impedance	3 kΩ	-

*) accuracy of adjustment ± 5 mV. Balancing is carried out in the factory and stored in a digital programmable potentionmeter.

Digital outputs of all PCD2.H32x modules

	Axis 1	**Axis 2**
Outputs	SO	SO
Suppply	Uext	Uext
Uext (typ. 24V DC)	6…32V DC	6…32V DC
Iout	5…500mA	5…500mA
Voltage drop @500mA	<0.3V	<0.3V
Short-circuit protection	yes 1)	yes 1)
Galvanic isolation	no	no

1) short circuit current limited to max. 1.6A

Supply for 5V encoder of the PCD2.H325/327 modules

Short-circuit protection	Yes
Electrical isolation	No
Output voltage	5 V
Max. load current	300 mA
Short-circuit current	400 mA

Important :

The maximum current available for the 5V encoders is 1.6A for a PCD2 or 750 mA for a PCD1.

Users of PCD2.H32x modules are urged to check the overall current consumption of **all** modules in a PCD2/1 **and** in any C100 or C150 expansion units to ensure that this maximum is not exceeded. (The current provided to supply the 5V encoder also comes from this source and must equally be taken into consideration).

When working with an expansion unit and up to 4 H32x modules, it is better to place the H32x modules in the base unit and to plug “normal” I/O modules into the expansion unit. This will eliminate any influences such as a possible voltage drop over the connecting cable from the expansion unit to the base unit.

Operating conditions

Ambient temperature	Operation: 0 ...+50°C without forced ventilation Storage: -20 ... +85°C
Noise immunity	CE mark, compliant with EN 61131-2, EN 50081-1 and EN 50082-2

General

Processor Chip-Set	PMD 2120
Programming	Based on PCD user programs (PG4/PG5) and supported by a library of FBs.

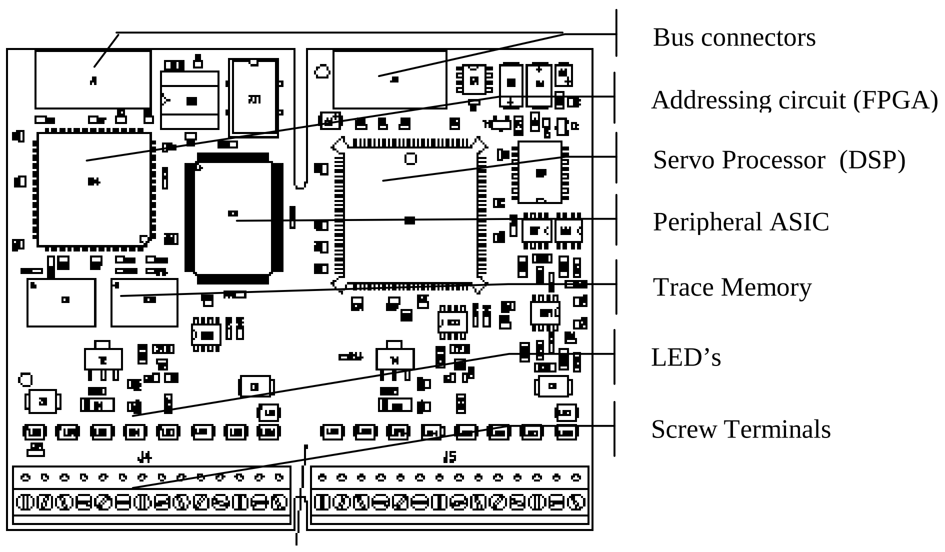
Ordering details

PCD2.H320	2 axis for encoder 24 VDC
PCD2.H325	2 axis for encoder 5 V/RS 422 + SSI
PCD2.H322	1 Slave (or single) axis for encoder 24 VDC
PCD2.H327	1 Slave or (1 single axis) for encoder 5 V/RS 422 + SSI
PCD9.H32E	Software library with function blocks (FBs) for programming in IL.
PCD8.H3xE	Software Tool for comfortable initialisation, programming and commissioning

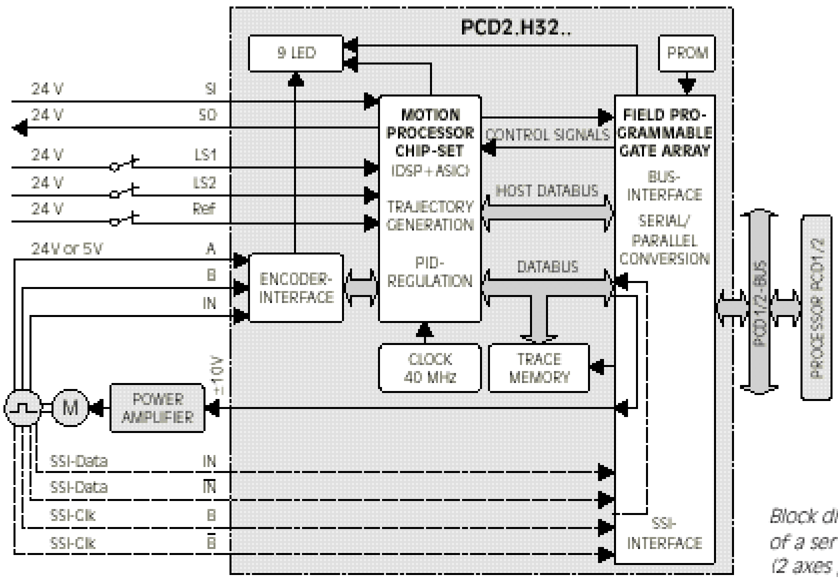
Notes

3. Presentation

Assembled module



Simple block diagram



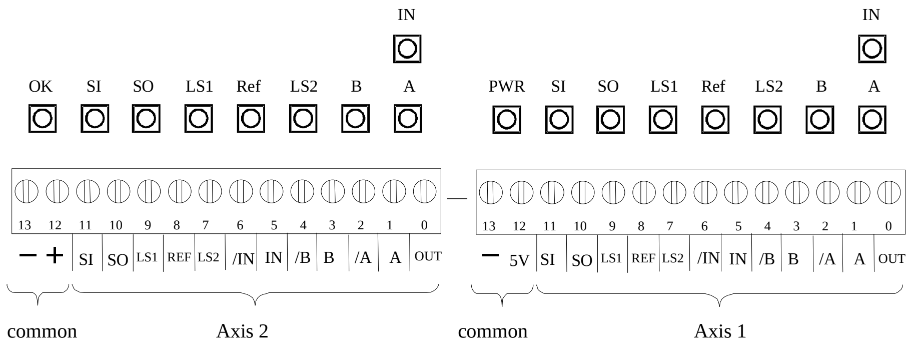
Block diagram of a servodrive for 1 axis (2 axes per module)

Notes

4. Terminals and meaning of LEDs

Terminals and LEDs of the PCD2.H32x

The picture shows the labelling of the printed circuit board. Both I/O connector blocks are numbered 0 to 13 from right to left.



IN
OK
SI
SO
LS1
Ref
LS2
B
A
PWR
13 12 11 10 9 8 7 6 5 4 3 2 1 0
− + SI SO LS1 REF LS2 /IN IN /B B /A A OUT
− 5V SI SO LS1 REF LS2 /IN IN /B B /A A OUT
common
Axis 2
Axis 1

Inputs: (per axis)

Module Type	H320	H322	H325	H327
Terminal 1 = "A"	Encoder signal "A"			
Terminal 2 = "/A"	*not connected*		Encoder signal "/A"	
Terminal 3 = "B"	Encoder signal "B"			
Terminal 4 = "/B"	*not connected*		Encoder signal "/B"	
Terminal 5 = "IN"	Encoder signal "IN"			
Terminal 6 = "/IN"	*not connected*		Encoder signal "/IN"	
Terminal 7 = "LS2"	Limit Switch 2			
Terminal 8 = "REF"	Reference Switch			
Terminal 9 = "LS1"	Limit Switch 1			
Terminal 11 = "SI"	Synchronisation Input			

Outputs: (per axis)

Module Type	H320	H322*	H325	H327*
Terminal 0 = "OUT"	Analog controller output *			
Terminal 10 = "SO"	Synchronisation output			

* only on (slave) axis 1

Supply: (per module)

Module Type	H320	H322	H325	H327
Terminalblock Axis 2				
Terminal 12 = "+"	External power supply input + 24 VDC, smoothed for SO			
Terminal 13 = "-"	user ground (PGND)			
Terminalblock Axis 1				
Terminal 12 = "5V"	Not connected		power supply output + 5VDC for encoder	
Terminal 13 = "-"	user ground (PGND)			

Meaning of LEDs

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

LED **IN** represents the encoder's index input.
This LED is "on" when the logic state is "H".

LED **REF** shows the logic state of the reference switch.
This LED is "on" if the switch is activated or when the switch has not been wired. By default this input is active low and for safety reasons, a N.C. switch or N.C. PNP sensor is recommended.

LEDs **LS1, LS2** shows the logic state of the limit switches.
This LED is "on" if the switches are activated or when the switches have not been wired. By default these inputs are active low and for safety reasons N.C. switches or N.C. PNP sensors are recommended.

LED **PWR** represents overall power supply indicator. This LED is "on" when the internal +/-15V is OK.
For PCD2.H325/27 it represents in addition whether the 5V for encoder supply is OK (no short-circuit).

LED **OK** shows the state of the on board motion controller and communication with PCD program.

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

Notes

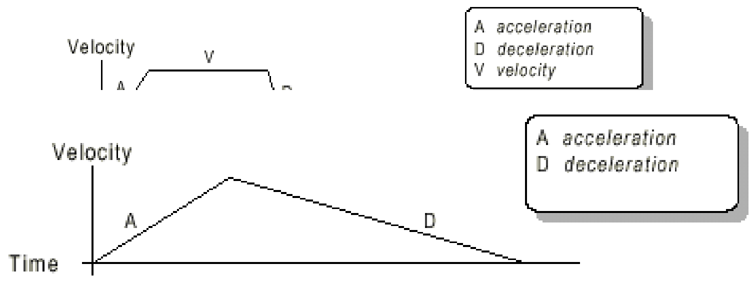
5. Function Description

5. 1 Trajectory Generation

The possible profiles are trapezoidal, S-curve, velocity contouring and electronic gearing. The axis profile is independent between axes, so two axes may have two different profiles. It is also possible to change the profile type during motion but the restriction is that to switch to S-curve from any other profile, the axis must be at rest.
Trapezoidal profile specifications:

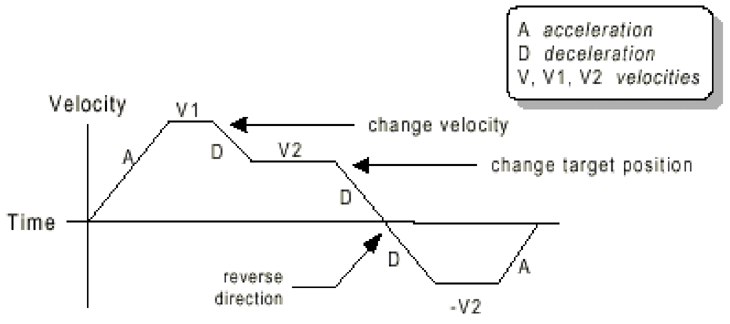
- Position is entered in units.
- Velocity is entered in units/s
- The acceleration and deceleration is entered in units/s²
- The user units are difined by the mechanic factor. (see chap. 7 & 9)
- The acceleration and deceleration can be different values.
- The acceleration and deceleration can be changed during motion.
- All parameters are absolute values, there is no relative positioning mode.

The profile gets its name from the resulting curve: the axis accelerates linearly (at the programmed acceleration value) until it reaches the programmed velocity. It continues in motion at that velocity, then decelerates linearly (using the deceleration value) until it stops at the specified position.



If deceleration must begin before the axis reaches the programmed velocity, the profile will have no constant velocity portion, and the trapezoid becomes a triangle.
The slopes of the acceleration and deceleration segments may be symmetric (if acceleration equals deceleration) or asymmetric (if acceleration is not equal to deceleration).
The acceleration parameter is always used at the start of the move. Thereafter, the acceleration value will be used when the absolute velocity is increasing, and deceleration will be used when the absolute velocity is decreasing. If no motion parameters are changed during the motion then the acceleration value will be used until the maximum velocity is reached, and the deceleration value will be used when ramping down to zero. When the direction is reversed, the deceleration parameter is used for acceleration to the target velocity.
It is acceptable to change any of the profile parameters while the axis is moving in this profile mode.

The profile generator will always attempt to remain within the legal bounds of motion specified by the parameters. If, during the motion, the destination position is changed in such a way that an overshoot is unavoidable, the profile generator will decelerate until stopped, then reverse direction to move to the specified position. Note that since the direction of acceleration/deceleration is fixed at the start of a move, the deceleration value will be used when ramping up velocity for the final move to the destination position. This is shown below.

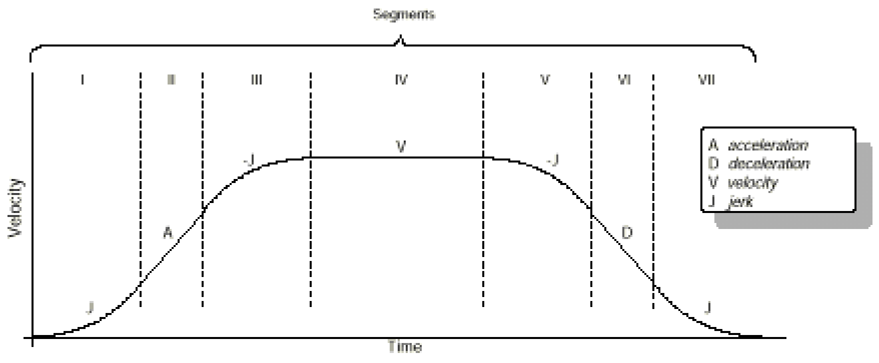


If a deceleration value of 0 (zero) is programmed (or no value is programmed leaving the chipset's default value of zero), then the value specified for acceleration (SetAcceleration) will automatically
be used to set the magnitude of deceleration.

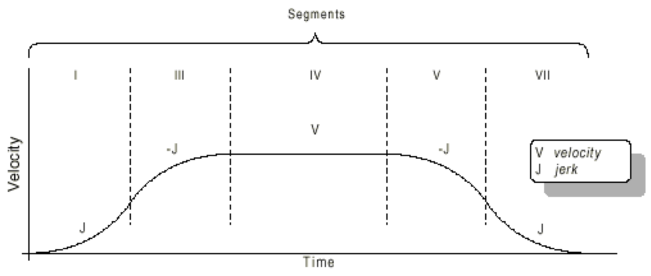
S-curve profile specifications:

- Position is entered in units.
- Jerk is entered in units/s³
- Velocity is entered in units/s.
- Acceleration and deceleration is entered in units/s²
- The user units are defined by the mechanical factor. (see chap. 7 & 9)
- The parameter „Jerk“ specifies the maximum change of acceleration and deceleration per second).
- The acceleration and deceleration must be the same, an asymmetric profile is not allowed.
- The S-curve profile does not support the change of any parameters during motion.

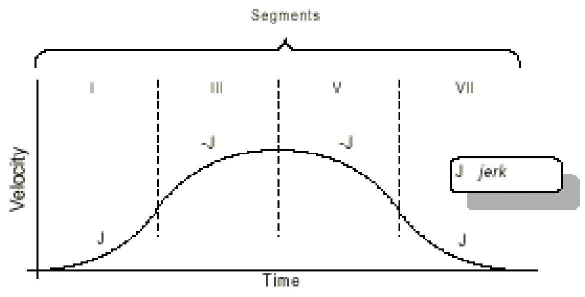
In this profile mode, the acceleration gradually increases from 0 to the programmed acceleration value, then the acceleration decreases at the same rate until it reaches 0 again at the programmed velocity. The same sequence in reverse brings the axis to a stop at the programmed destination position.



S-curve profile
The figure above shows a typical S-curve profile. In Segment I, the S-curve profile drives the axis at the specified jerk (J) until the maximum acceleration (A) is reached. The axis continues to accelerate linearly (jerk = 0) through Segment II. The profile then applies the negative value of the jerk to reduce acceleration to 0 during Segment III. The axis is now at maximum velocity (V), at which it continues through Segment IV. The profile will then decelerate in a manner similar to the acceleration stage, using the jerk value first to reach the maximum deceleration (D), and then to bring the axis to a halt at the destination.
An S-curve profile might not contain all of the segments. For example, if the maximum acceleration cannot be reached before the “halfway” point to or from the velocity, the profile would not contain a Segment II or a Segment VI. such as shown in the profile below.



Similarly, if the position is specified such that velocity is not reached, there will be no Segment IV (There may also be no Segment II or Segment VI, depending on where the profile is “truncated.”).



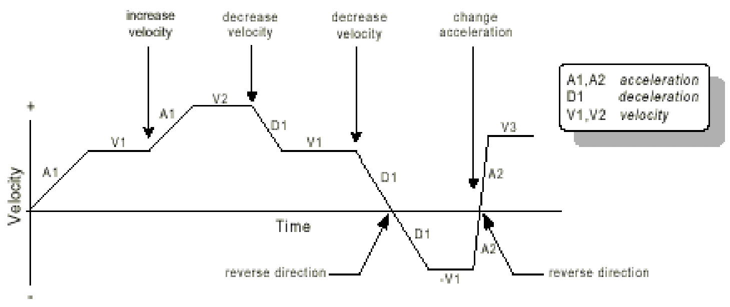
S-curve with no maximum-velocity segment
Unlike the trapezoidal profile mode, the S-curve profile mode does not support changes to any of the profile parameters while the axis is in motion.
An axis may not be switched into S-curve profile mode while the axis is in motion. It is however legal to switch from S-curve mode to any other profile mode while in motion.

Velocity contouring profile specifications:

- Velocity, acceleration and deceleration are in user units.
- The velocity is a signed value which gives the direction of the motion. There is no position given, this profile is entirely controlled by changing the velocity, acceleration and deceleration parameters while the profile is being executed.

In velocity contouring profile mode axis motion is not bounded by a destination. It is the host’s responsibility to provide acceleration, deceleration, and velocity values which result in a safe motion within acceptable position limits.
The trajectory is executed by continuously accelerating the axis at the specified rate until the velocity is reached. The axis starts decelerating when a new velocity is specified which has a smaller value (in magnitude) than the present velocity, or has a sign that is opposite to the present direction of travel.
A simple velocity-contouring profile looks just like a simple trapezoidal point-to-point profile.

The following figure illustrates a more complicated profile, in which both the velocity and the direction of motion change twice.

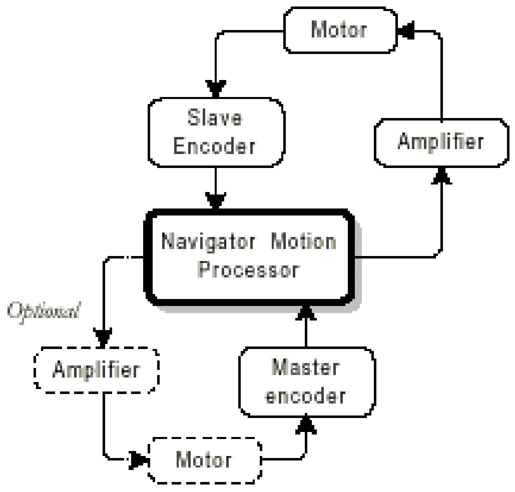


Velocity-contouring profile

Electronic-gear profile specifications:

- The master axis is the axis that will be the source of the position information used to drive the slave axis.
- The gear source can be either actual (encoder position of the master) or commanded (the commanded position of the master).
- The gear ratio specifies the direction and the ratio of the master gear counts to slave counts. The ratio can be changed during motion.

The figure below shows the arrangement of encoders and motor drives in a typical electronic gearing application.



Axes set up for electronic-gear profile
A positive gear ratio value means that when the master axis actual or commanded position is increasing the slave commanded position will also increase. A negative gear ratio value has the opposite effect; increasing the master position will result in decreasing the slave axis commanded position. For example, let us assume the slave axis is axis 1 (axes are counted 1, 2 for the module) and the master axis are set to axis 2. Also assume the source will be ‘actual’ with a gear ratio of - ½. Then for each positive encoder count of axis 2, axis 1 commanded position will decrease in

value by ½ count, and for each negative encoder count of axis 2, axis 1 commanded position will increase in value by ½ count.
If the master axis source is set to ‘actual’, this axis need not have a physical motor attached to it.
Frequently, it is used only for its encoder input, for example from a directly driven (open-loop) motor, or a manual control. It is possible, however, to drive a motor on the master axis by enabling the axis and applying a profile mode *other* than electronic gear to the axis. The effect of this arrangement is that both master and slave can be driven by the same profile, even though the slave can drive at a different ratio and in a different direction if desired.
The master axis will operate the same whether or not it happens to be the master for some other geared axis. The ‘optional’ components shown in the-figure above illustrate this arrangement. Such a configuration can be used to perform useful functions such as linear interpolation of two axes.
The gear-ratio parameter may be changed while the axis is in motion, but care should be taken to select ratios so that safe motion is maintained.
See page 5-21 for some information regarding Homing when using Ele-ectronic Gearing mode.

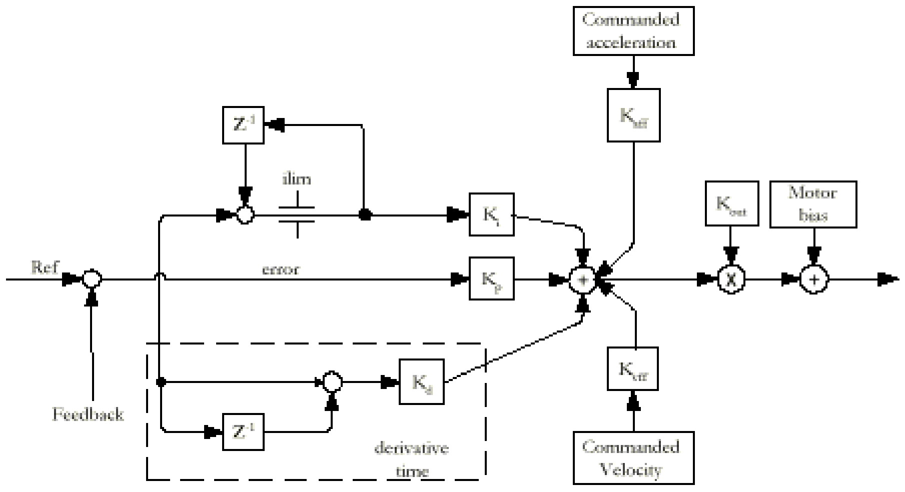
The SetStop command:

- This command is used to stop an axis during a „normal“ profile. The stop action can be smooth or abrupt.
- The smooth stop uses the defined deceleration ramp, the abrupt stop is an emergency stop which must be used carefully otherwise damage can occur to the equipment.
- Abrupt stop functions in all profiles, Smooth stop functions in all profiles except electronic-gear.

Motor mode:
If the motor mode is set to on then the trajectory generator is active. If the motor mode is set to off then the profile generator is disabled and the module enters into open loop mode. The use of motor off mode can be when a motion error occurs to place the motor in a safe state where no further moti-on can occur.
Setting the cycle time:
The module H32x calculates all trajectory and servo information on a fixed, regular interval. This interval is known as the cycle time (204.8µs).

5.2 The Servo Loop

The servo filter used is a Proportional-Integral-Derivative algorithm, with velocity and acceleration feed-forward terms and an output scale factor. An integration limit provides an upper boundary for the accumulated error. An optional bias value can be added to the filter calculation to produce the final motor output command. A limiting value for the filter output provides additional constraint.



Motor bias:
When an axis is subject to a net external force in one direction (such as axis pulled downward by gravity), the servo filter can compensate for it by adding a constant DC bias to the filter output.
Kout:
Defines the output scale factor of the digital servo filter. By default this value is set at 65535. If the optimum P-factor setting is not a whole number, then you can use this parameter to change the resolution of the output to obtain the desired setting. For example, if the optimum P-factor setting is 1.5 then you would set Kout to 32767 and enter 3 as your P-factor value. The instruction SetKout is located on page A-18 in Appendix A of this manuel.
Output limit:
The motor output limit prevents the filter output from exceeding a boundary magnitude in either direction. If the filter produces a value greater than the limit, the motor command takes the limiting value. The motor limit applies only in closed-loop mode.
Closed-loop & Open-loop mode:
In open-loop mode the servo filter does not operate and the motor command output value is set ‚manually' by executing an instruction in PCD program. With the motor „on" the module is in closed-loop mode and the motor command value is controlled automatically by the servo filter.
Limit switches do not work in open-loop mode.
The bias (offest volatage) applies in both modes.

5.3 Parameter Update and Breakpoints

Parameter buffering:
In some cases it may be desired that a set of parameters take effect at the exact same time to facilitate precise synchronised motion. To support this, all profile parameters and several other types of parameters such as servo parameters are loaded into the module using a buffer scheme. An update causes the buffered registers to be copied to the active registers, thereby causing the module to act on the new parameters.
There are two different possibilities to update the parameters:

- Using the command "Update". (specify axis 1, 2, or 1 & 2)
- Using breakpoints, which can generate an update command automatically when a pre-programmed condition becomes true.

Breakpoints:
Breakpoints are a convenient way of programming a module event upon some specific conditions. Each navigator axis has two breakpoints that may be programmed for it.
Each breakpoint has five parameters:

- The breakpoints axis is the axis on which the specified action is to be taken
- The source axis is the axis on which the triggering event is located. It can be the same as or different than the breakpoint axis.
- The trigger is the event that causes the breakpoint.
- The action is the sequence of operations executed by the module when the breakpoint is triggered.
- The comparison value is used in conjunction with the action to define the breakpoint event.

The breakpoint trigger conditions are:

- >= or <= commanded position.
- >= or <= actual position.
- Actual position crossed.
- Time.
- Event status (EventStatus register matches bit mask).
- Activity status (ActivityStatus register matches bit mask).
- Signal status (SignalStatus register matches bit mask).
- None (disable any previously set breakpoint).

5.4 Status Registers

The status registers combine a number of separate bit-oriented fields for the specified axis. These three 16-bit registers **are Event Status, Activity Status** and **Signal Status**. They can be used as a source of data for the AxisOut mechanism, which allows any bit within these three registers to be output as a hardware signal.

Event Status register:

Each of the bits in this register is set by the module and cleared executing the PCD FB: ResetEventStatus

Bit	**Description**
0	Motion complete
1	Position wraparound (set when the actual position exceeds the most positive position and wraps to the most negative position, or vice versa.
2	BreakPoint 1
3	Capture received (set when the high speed position capture hardware acquires a new position value)
4	Motion error
5	Positive limit
6	Negative limit
7	Instruction error
8-10	Reserved
11	not used
12 – 13	Reserved
14	BreakPoint 2
15	Reserved

Activity Status register:
Each of the bits in this register is continuously set and reset by the module:

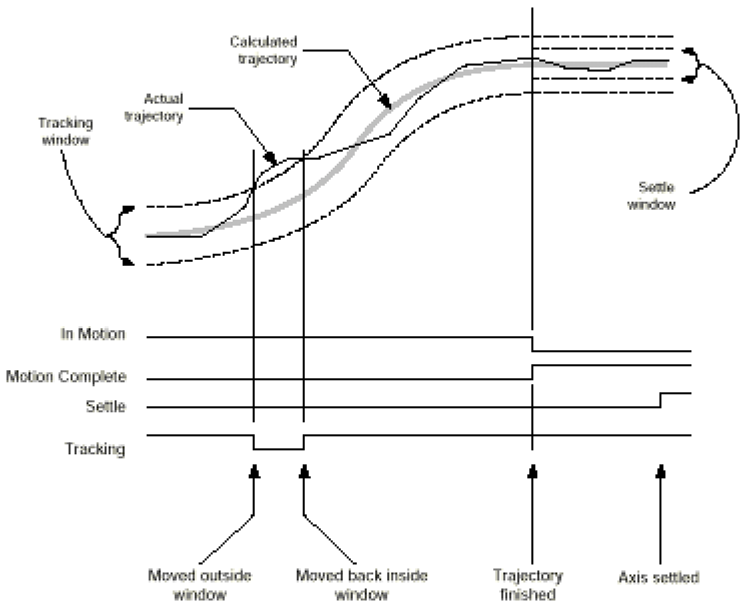
Bit	Description
0	not used
1	At maximum velocity (does not work in electro-nic-gear).
2	Position tracking (set when the servo is keeping the axis in the tracking window).
3 - 5	Current profile mode. 0: trapezoidal 1: velocity contouring 2: S-curve 3: electronic gearing
6	Axis settled (set when the axis has remained within the settled window for a specified period of time).
7	Reserved
8	Motor mode
9	Position capture (set when a new position value is available to read from the high speed capture hardware).
10	In-motion indicator (set when the trajectory pro-file commanded position is changing).
11	In positive limit
12	In negative limit
13 - 15	S-curve segment (indicates S-curve segment number using values 1-7).

Signal Status register:
The signal status register provides real time signal levels for various module I/O pins:

Bit	Description
0	A Encoder
1	B Encoder
2	Index encoder
3	Home (Ref. Switch)
4	Positive limit (LS1)
5	Negative limit (LS2)
6	AxisIn (generic axis input signal)
7 - 9	not used
10	Axis Out
11 - 15	Reserved

5.5 Monitoring Motion Performance

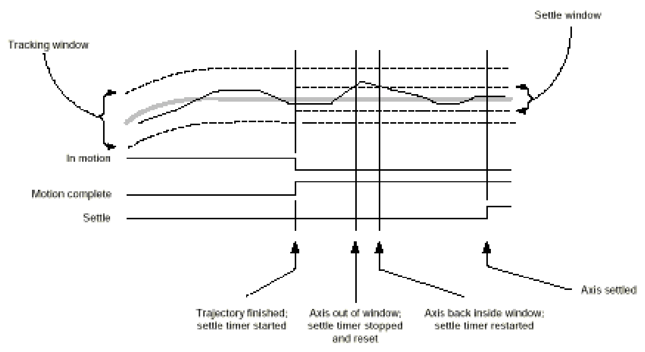
Motion error:
At the moment a motion error occurs, several events occur simultaneously. The motion error bit of the event status word is set. If automatic stop on motion error is enabled the motor is set off, which in turn disables the trajectory generator and the servo loop, placing the module in open-loop mode. If automatic stop on motion error is not set then only the motion error status bit is set. (EventStatus register bit 4)
Tracking window:
The tracking window functions similarly to the motion error, in that there is a programmable position error limit within which the axis must stay. The axis is not stopped when the actual trajectory goes outside the tracking window. To monitor the position error in reference to the tracking window, obeserve bit 2 of the activity status register. As long as the axis is within the window, the bit is set high. If it travels out of the window it is set low. This is displayed in the timing chart below.



Motion complete indicator:
Motion complete can indicate the end of the trajectory motion
in two ways:

- In commanded mode, the motion complete indicator is set based on the profile generator registers only. The motion is considered complete when the trajectory generator registers for commanded velocity and acceleration both become zero. This normally happen at the end of a move when the destination position has been reached. But it may also happen as the result of a stop command, a change of velocity to zero or when a limit switch event occurs.
- In actual mode, meaning the motion complete indicator is based on the actual encoder. The motion is considered complete when the profile generator (commanded) motion is complete **&** the difference between the actual position and the commanded position is less than or equal to the value of the settle window **&** when the to above conditions have been met continuously for the last N cycles, where N is the programmed settle time.

The motion complete bit (EventStatus register bit 0) is not active in electronic-gear mode.
In-motion indicator:
Bit 10 of the activity status register indicates if the axis is in motion or not. It does not work with electronic-gear mode.
Settled indicator:
The settled indicator appears in bit 6 of the activity status register. The axis is considered to be settled when the axis is at rest (not performing a trajectory profile) and when the actual motor position has settled in at the commanded position for the programmed settle time.
The settle time and the settled window used with the settled indicator are the same as for the motion complete bit when set to actual.


Settle window
Tracking window
In motion
Motion complete
Settle
Axis settled
Trajectory finished; settle timer started
Axis out of window; settle timer stopped and reset
Axis back inside window; settle timer restarted

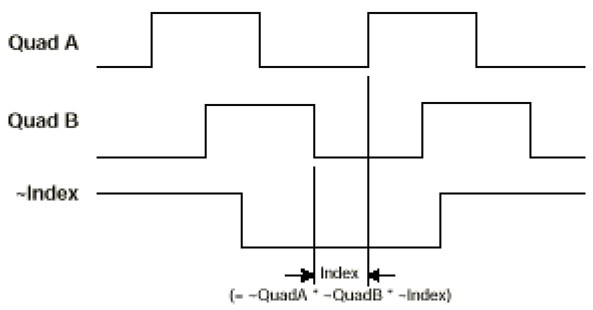
Data trace:
Data trace allows various module parameters and registers to be continuously captured and stored in an external memory buffer. The captured data can later be uploaded by the PCD using standard memory buffer access instructions(these instructions are only acessible via the commissioning tool).
There are 27 separate items within the module that can be stored such as actual position, event status register, position error etc.
The module can capture the value of the trace variables every single module cycle, every other cycle or at any programmed frequency.

The module can trace in two modes, one-time or rolling mode. This determines how the data is stored and whether the trace will stop automatically or whether it must be stopped by the commissioning tool.
One-time mode means that the trace mechanism will stop collecting data when the buffer is full. Rolling mode means that the trace mechanism will wrap around to the beginning of the trace buffer and continue storing data.
To allow precise synchronisation of data collection it is possible to define the start and stop conditions for a given trace.
It is possible to trace up to four module variables.
Host interrupt:
The events that trigger a host interrupt are the same as those that set the assigned bits of the event status register.
Using a 16-bit mask, the host may condition any or all of these bits to cause an interrupt:

Bit	**Description**
0	Motion complete
1	Position wraparound (set when the actual position exceeds the most positive position and wraps to the most negative position, or vice versa.
2	BreakPoint 1
3	Capture received (set when the high speed position capture hardware acquires a new position value)
4	Motion error
5	Positive limit (LS1)
6	Negative limit (LS2)
7	Instruction error
8 – 10	Reserved
11	not used
12 – 13	Reserved
14	BreakPoint 2
15	Reserved

5.6 Motor Interfacing

Incremental encoder input:
Incremental encoder feedback provides two square-wave signals:



Quadrature Encoder timing
The module continually monitors the position feedback signals and accumulates a 32-bit position value called the actual position. Upon power up the default actual position is zero. Particularly when using incremental feedback, the actual position is generally set shortly after power up, using a homing procedure to reference the actual position to a physical hardware location. The internal resolution is 4-times higher than the number of impulses given by one encoder input A or B.
The max. input frequency of the encoder signals is limited by an additional analogue input filter:
PCD2.H320/H322 (24V encoders) :
F_{max} = 125kHz (500kHz internal counting frequency)
PCD2.H325/H327 (5V encoders) :
F_{max} = 250kHz (1MHz internal counting frequency)

Encoder Supply (only H325/H327)
The supply voltage for the 5V encoders is taken from the PCD2 Bus (+5V supply). It is filtered against external perturbation and short circuit protected with a multifuse, but a short circuit on the 5V encoder supply can produce a CPU reset because the multifuse has to heat up before it limits the current.
The current for the two encoders is specified to 400mA per H325/H327.
The total current available from the 5V supply for the I/O-modules is limited to 1.6A for the PCD2.M and 750ma for the PCD1.M.
The supply current for the +5V encoder has to be taken into the calculation of the max. current consumption.
Encoder CableBreak detection (only H325/H327)
An encoder error detection is implemented on the PCD2.H325/H327.
By controlling the logic status of the differential encoder signals A, /A and B, /B a cable break, an encoder fail or a missing encoder supply can be detected. The Index signals IN and /IN are not controlled.
The encoder error signal is displayed on the PCD2 I/O-Bus and has to be controlled with the user program by checking the status of the Cablebreak_XsY input. (input 26) For checking status details see page 5 – 16.
Any action as a result of a cable break, has to be implemented in the user program. Nothing is done automatically.
Encoder error detection is not possible with H320/H322 (24V encoders).

SSI-Interface (only H325/H327)
The input on the controller is limited to 16 bits. SSI-encoders normally have 24 bits of data even if all 24 bits are not used for the position value. (depending on the resolution and the positioning range).
The number of SSI-clock pulses has to be defined by the user. The full position (26 bit) is read in the FB INIT after start up. This position is sent as an absolute position to the controller. The controller only uses the lower 16 bits of the SSI value and calculates the position wrap around itself. In this case the wrap around position has to be loaded to 32768 so that the controller can count the correct position with only 16 bits of data. The axis must be at rest during this initialisation.
SSI Configuration

- The encoder mode (incremental or SSI-absolute) can be selected independently for both axis.
- Other SSI configurations: no. of data bits, frequency and code is common for both axis.
- This must be done in the FB Init.

The number of SSI-clock pulses can be selected from 8 to 26.
The SSI-data can be coded in binary or in gray code. In case of gray data coming from the absolute encoder, a conversion gray →bin is done on the module.
The SSI clock frequency is selectable from 250kHz to 1MHz.
SSI-Functions

Resolution:	configurable 8 to 26 data bits
Clock frequency:	250kHz, 500kHz, 1Mhz
max. cable length:	100m (with 250kHz) 50m (with 500kHz) 20m (with 1MHz)
Data refreshing :	300µs (with 250kHz) 200µs (with 500kHz) 100µs (with 1MHz)
Data-Code:	configurable gray or binary
Read Mode:	normal mode (single read), no Ringregister mode
Galvanic separation:	Data input (the clock is separated in the encoder)
Encoder fail detection:	by timeout of 30µs detecded by the FPGA (no cable break detection except with H325/H327)
Connecting:	SSI-Clk (output) on Encoder B /SSI-Clk (output) on Encoder /B SSI-Data (input) on Encoder IN /SSI-Data (input) on Encoder /IN

Note: Shielded and twisted pair cables should be used.

Errors with SSI Absolute Encoders
After the initialisation in SSI mode, the absolute encoder is accessed continuesly. If the communication between the H325/H327 module and the encoder fails, a SSI timeout is generated after 30µs and then turns off the led "OK".
From the PCD, a SSI timeout can be detected by monitoring the SSI_timeout_XsY input. (input 23)
If a SSI timeout has occurred, the following is possible:
1. no power (power supply to encoder)
2. broken cable
3. connection wired incorrectly
4. defective encoder or not an SSI encoder

Once a SSI timeout has occurred, the axis must be reinitialised after the problem has been corrected. With the new initialisation, the led "OK" is turned back on, and the input SSI_timeout_XsY is reset.

AxisSelect_X_Y Output
There are seven elements that the axis must be designated using the AxisSelect _X_Y output. These elements are listed on page 7 – 26 of this manuel.
The status of these elements are on the PCD2 I/O bus and can be checked by using this element/output to designtate the axis you want to see the status of. For example, if you want to monitor the SSI timeout of axis 1 and then activate an output when a timeout occurs, you would program it is follows:

```
RES     AxisSelect_1_2      ;RES=Axis1 Set=Axis2
STH     SSI_timeout_1_2     ;If a timeout has occurred
Out     O 112               ;activate Output 112
```

Another example, if you wanted to monitor Axis 4 for a cable break and then if a broken cable is detected call PB 25, your program segement would be as follows:

```
SET     AxisSelect_3_4      ;RES=Axis3 Set=Axis4
STH     CableBreak_3_4      ;If a cable break is detected
CPB H   25                  ;call Progam Block 25
```

The above examples are very basic and are only to show how to access and utilise these 7 elements.

Capture Register

Each axis of the module supports a high-speed position capture register that allows the current axis location to be captured, triggered by an external signal. When a capture is triggered, the contents of the actual position register is transferred to the position capture register, and the capture-received indicator in the event status register is set along with the fCaptureTrig_n flag. The commands SetCaptureSource and GetCaptureValue are used in conjunction with three available signals to execute this functionality. The signals are located in the SignalStatus register and they are as follows:

Encoder Index: When using "Encoder Index" to trigger the high-speed capture of the actual axis position, both Channels A & B need to be low, preceded by a high to low transition on the Encoder Index input. Note, check the timing diagram of the encoder being used to determine the sequence for achieving this condition. A timing diagram example is on the preceding page.

Reference Switch: When using "Reference Switch" to trigger the high-speed capture of the actual axis position, the only condition that has to occur is that there has to be a High to low transition on the Reference switch input. The state of Channel A and Channel B are irrelevant. Keep in mind that the reference switch is active low so when the led is on the state of the input is low, and vice versa when the state of the input is high, the led is off.

Axis In: When using "Axis In" to trigger the high-speed capture of the actual axis position, it is the same conditions as with "Encoder Index". Channels A & B need to be low, preceded by a high to low transition on the Axis In input.

Notes:

1. Channel A, B, and Encoder Index are respectively bits 0, 1, and 2 of the SignalStatus register. The SignalSense register determines the active state (high or low) of the SignalStatus register's inputs and outputs. The SignalStatus register is masked by the SignalSense register, so it is important that when using the SetCaptureSource and GetCaptureValue commands that these 3 bits in the SignalSense register are low. (this is the default setting for the SignalSense register). If these bits are set high, the commands SetCaptureSource and GetCaptureValue will not function properly.
2. Once a capture has taken place, it is necessary to activate the GetCaptureValue command before another capture can take place. This command reloads the trigger mechanism.
3. The fCaptureTrig_n flag and bit 3 (capture-received indicator) of the EventStatus register are set high when a capture is completed. To reset these, you have to execute a "ResetEventStatus" command.

Signal Status Register – Axis In
The Axis In input can be used for 3 basic functions:

1. Trigger a high speed positon capture as explained above.
2. Trigger a breakpoint.
3. Used as a general purpose input. (the address is the H32x base address plus 30. Example base address 64 Axis In address 94.

When using Axis In to trigger a breakpoint, you use the command "TrgSignal" for parameter 5 of the SetBrkPoint FB. Parameter 6 is for the Value parameter and you have to enter a "two word mask," a high word and a low word. The high word is the mask that determines the bit of the register (in this case Axis In is bit 6) and the low word determines the sense or the active state of the bit high or low. If a 1 is entered in the sense for the chosen bit, then when that bit goes high it will trigger the breakpoint. If a 0 is entered then when it goes low it will trigger the breakpoint. To designate Axis In to trigger a break when it goes high, we would load 00400040H for the value. This value was determined because bit 6 of both the high word and the low word are set to 1. For more details on the SetBrkPoint FB, see page A-5 and A-6 in Appendix A of this manual.

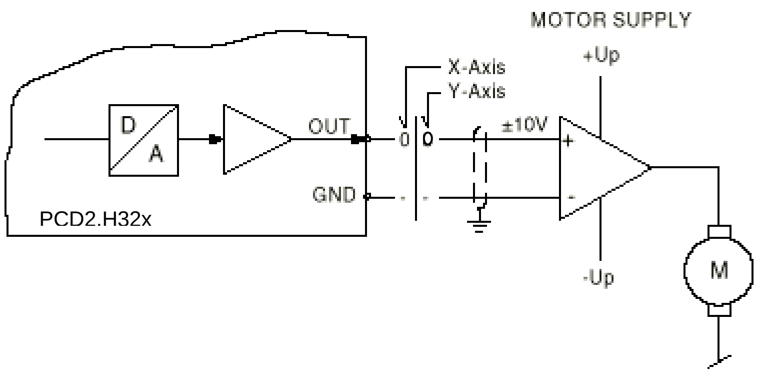
Signal Status Register – Axis Out
The Axis Out output is used in conjunction with the command "SetAxisOutSource" and is set to track the state of a designated bit of one of the Status registers. (Event, Activity, or Signal). When entering the CFB Exec for executing this command, parameter 3 is what determines the register, bit, and axis that the Axis Out pin is going to track. Reference page A – 43 in the Appendix A of this manual to follow the explanation below.
For example if we want to have the Axis Out pin track the Motion Complete bit of the Event Status register for axis 1, the FB instruction will be entered as follows:

```
LD  R 100
    100h
CFB Exec
K 1
SetAxisOutSource
R 100
```

The value 100h breaks down as follows:
1 = Event status register
0 = bit number
0 = Axis number

5.7 D/A Converter output

A 12 bit D/A converter has been used.
Analogue output connection:



Offset voltage
The max. offset voltage of the analogue outputs are adjusted to ±5mV on production by two digital potentiometers..
The user has no access to this offset adjusting but he can define a software offset with the command SetMotorBias.

5.8 Complementary information: homing (FB Home)

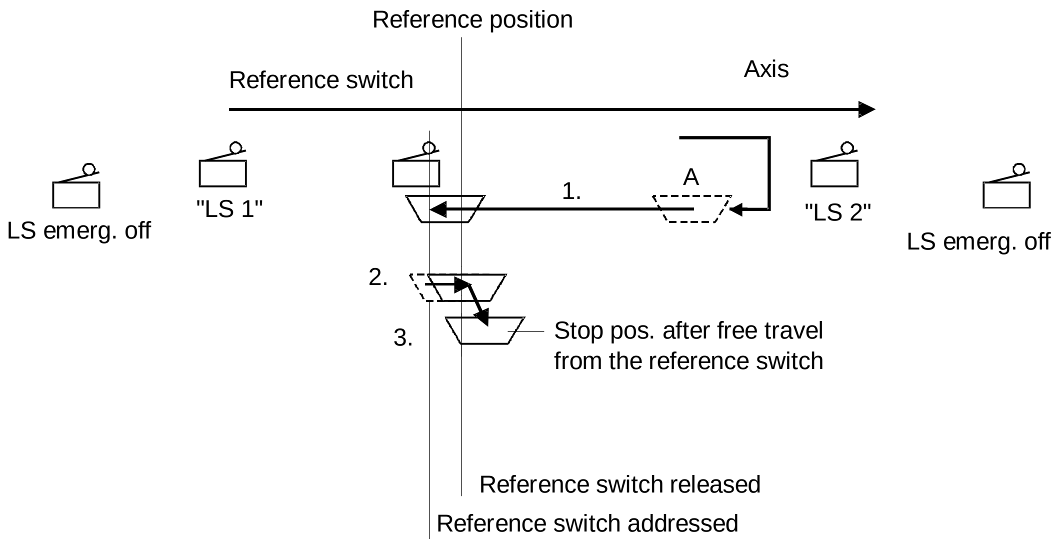
With the ‘Home’ FB, independent homing can take place. Seven parameters are required for definition of the home travel.
The axis to be referenced must have been initialized (FB Init).
For successful use of the ‘Home’ FB, the axis and reference switch must be located between ‘LS1’ and ‘LS2’.

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

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

3. Parameter 7 (encoder index) determines whether the homing stops when the reference switch is released or the first encoder index signal following the release of the reference switch. This position must now be defined by the user and is normally set as zero position.

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


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

Instructions:

- Several modules can be referenced on one PCD at the same time.
- Flags ‘fLS1_n’ and ‘fLS2_n’ are representations of the limit switches. The logic states of both flags are the result of querying both digital inputs.
- Normally closed contacts should be used for the switches.
- During the homing procedure, limit switches are switched off internally and are only used for reversing the direction of travel.
  If no reference switch is found, the error flag ‘fHomeErr_n’ (n = module no.) is set and the ‘Home’ FB is exited.
- When the ‘Home’ FB has finished (successfully or broken off by an error) the ‘fEndHome_n’ flag is set automatically.
- Since the ‘Home’ FB is only exited when homing has been successfully completed or an error has been detected, the FB can be broken off with a timeout (parameter 6). Its value corresponds to the time in seconds after which the ‘Home’ FB will be abandoned. In this case, as well as the error flag ‘fHomeErr_n’, the diagnostic register ‘rDiag’ is loaded with code 6 (for parameter 6) in the third byte (for FB Home). Parameter checking takes place as described in chapter 8.
- The acceleration and deceleration have to be defined before starting the FB Home.

Homing with Electronic Gearing

The homing process stops by the reference switch going high and then low, or when the encoder index marker is seen following this high low transition. In either case, at this point the value of this location must be defined. For trapazoidal or S-curve all that is needed is the "SetActualPosition" command. For the Electronic Gearing mode, you must:

1. Take the Slave Axis out of Electronic Gearing mode
2. Set the reference position on the Master Axis. (normally this is 0)
3. Put the Slave Axis back into Electronic Gearing mode.

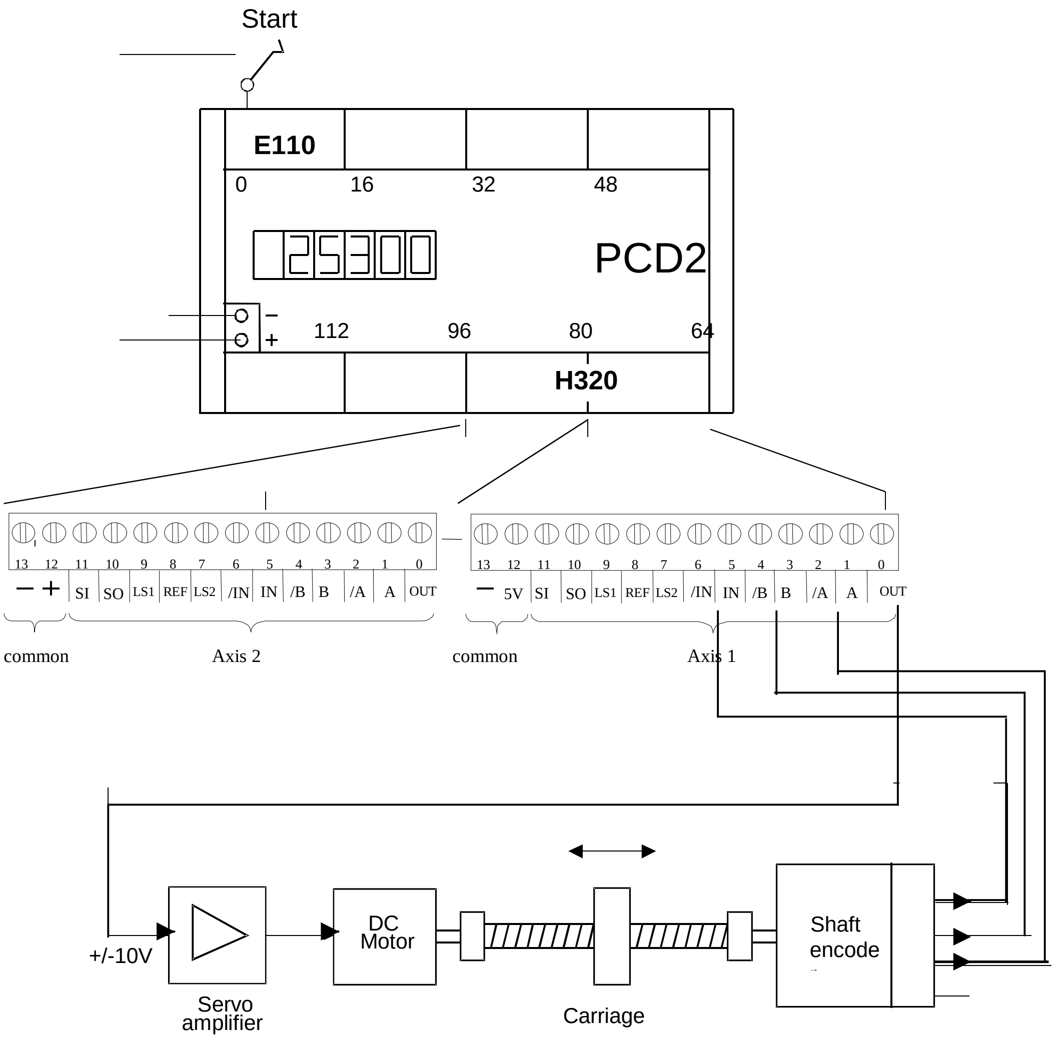
Following is an example of the code that would follow the FB Home Intialisation segement of your program to achieve this.
K 1 = Master
K 2 = Slave

```
;------Take Slave Axis out of Electronic Gearing Mode------

LD      rTemporary
        0                   ;0=Trapezoidal mode
cfb     Exec
        K 2                 ;Slave Axis
        SetProfileMode      ;Sets the profile mode
        rTemporary          ;on the Slave Axis

LD      rTemporary
        K 2


cfb     Exec
        K 2                 ;Slave Axis
        MultiUpdate         ;Update the status on
        rTemporary          ;the Slave Axis
```

```
;------Set Reference on Master Axis--------------------------

LD      rPosition
        0
cfb     Exec
        K 1                 ;Master Axis
        SetActualPosition   ;Set reference position
        rPosition           ;on Master Axis to zero

LD      rTemporary
        K 1                 ;Master Axis
cfb     Exec
        K 1                 ;Master Axis
        MultiUpdate         ;Update the status on
        rTemporary          ;the Master Axis

;------Place Slave Axis back into Electronic Gearing Mode----

LD      rTemporary
         3                   ;3=Electronic gearing mode
 cfb     Exec
         K 2                 ;Slave Axis
         SetProfileMode      ;Sets the profile mode
         rTemporary          ;on the Slave Axis

 LD      rTemporary
         K 2                 ;Slave Axis
 cfb     Exec
         K 2                 ;Slave Axis number
         MultiUpdate         ;Update the status on
         rTemporary          ;the Slave Axis
```

Notes

6. Quick Start



Minimal arrangement for position control using a PCD2.H32x, not including reference switch and limit switches.

The individual elements are:

- PCD1 or PCD2, equipped at least with 1 PCD2.H32x (with F510/530 display) 1 PCD2.E110
- Motion model with DC motor, spindle and incremental shaft encoder
- DC Motor with gears: approx. 500 rev/min at 10 VDC
  Spindle gradient: 1 mm/revolution
  Incremental shaft encoder: 1000 signals per revolution
- 4-quadrant servo amplifier
  e.g. with Op-Amp LM 12 (National Semiconductor)
  for details see: http://www.national.com/pf/LM/LM12.html
- PG4/5 and FBs
- For programming refer to chapter 7.

Notes

7. Programming

The PCD2.H32x motion control modules are programmable via SAIA's PG4 or PG5 software packages. If you are currently using PG3, you will have to upgrade to PG5 to be able to use the H32x series modules.
Since motion control tasks are always sequential processes, it is preferable if user programs are written in Graftec, using Instruction List programming with FBs (Function Blocks) for the code in the steps and transitions.
If you prefer you can program in straight Instruction List using FBs (Function Blocks) without using the Graftec structure.

In this chapter we are going to explain the following:

- Installation of the Function Blocks
- Structure of the Individual Function Blocks
- Module Base Address files (mba)
- Program Structures - Examples
- Elements
- Bus Interface

7.1 Installation of the FBs

The installation procedure for the H320 module is the same for both PG4 and PG5, however the format of the files that are installed is different between the two. PG5 requires that the files be compressed and with PG4 this is not necessary. The installation files are in two packages, Fb_Pg4 and Fb_Pg5.

There are 2 ways to get the FBs.
1. From the diskette PCD9.H32E
2. Download it from www.saia-burgess.com/pcdsupport

The installation package includes the following:

• APPSDIR	; Containing all the help files.
• FBs	; Containing the .SRC and .EQU files.
• D2H32x_B.mba	; Module base addresses file.
• Readme	; Information on installing this package in PG4 or PG5.
• Examples	; Demonstrations of functionality with very basic programs.

The installation procedure is as follows:
<Start><Programs><SAIA PG5 1.0><Setup Extra Files>
or
<SAIA PG4>

Use the browse window to locate and open the file folder (Fb_pg4 or Fb_pg5).
Click on Ok and then again on OK.
When asked if there are any more files click on No.
Click on Finished.

In your PG5 directory:
D2h320_b.Equ and D2h320_b.src files will be loaded into the FB file folder.
D2h320_b.mba will be loaded into the FB\MBA file folder.
D2h32x_b.hlp and Fb_Lib.hlp will be loaded into the main directory.
Examples: a project titled examples will be in the Project file folder.

In your PG4 directory, everything is the same as PG5 except that the mba file and the Examples file must be manually loaded. The mba file is normally copied into the FB file folder. The examples are normally copied into a PG4 project.

NOTES: Fb_Lib.hlp already exists, so when downloading a new FB, the existing one is overwritten. This can be a problem if you have a newer FB installed and then you need to install or reinstall an older one. If you are installing an older one the easiest thing to do is make a copy of the current one and then after running Setup extra files copy the current one back into the directory. If you are not sure, open the FB_Lib.hlp file and you should have a link to each of the FBs you have installed.

The project "Examples" loaded into the PG5 project file folder are functioning programs but are for demonstration purposes only. All programs (except Example 5) contain jumps which is not a recommended programming procedure.
All the Examples have an FB initialisation segment. MAKE SURE THAT THE PARAMETERS IN THE FB INITIALISATION SEGMENTS ARE CHANGED TO

MATCH THE USERS MOTOR AND ENCODER.
Examples 1, 2, 3, 5, also have a FB Homing segment. The Homing parameters also need to be changed to match the application.
The .mba file has the H320 module's base address at 64. If you are using a different location you will have to change this file accordingly.
The communication channel selected for these examples is S-Bus. If you are using a different channel than you will have to change the settings to match your application.
Descriptions of the programs are at the beginning of the programs. For example 5 because it is a Graftec example the description is at the beginning of the Initial Step.

.

7.2 Individual FBs

The FBs included in this package are:

FB INIT	Initialization	Parameters	20
FB HOME	Home position	Parameters	7
FB EXEC	Execution	Parameters	3
FB SetBrkPoint	Set Break Points	Parameters	6

The directive $include D2h320_B.EQU must be entered at the beginning of your program, or at least before any of the FBs in the program.
All FBs must have a H320 declaration. There are 2 possibilities for this:

1. Use the directives, $group H320 and $endgroup
2. Include *H320.* to precede the FB titles and instructions.

When using method one, you have to remember that all of the FBs for the H320 module must be between the directives, $group H320 and $endgroup. Whether you need to have a set of these for every FB, or if you want to include more than one FB in a set, depends on each individual user and their file structure. For example, you can have your FB Init and FB Home between one set of the $group directives, or you could have one set for the FB Init and one set for the FB Home.
When using method two, for the FB Init, FB Home, and FB SetbrkPoint, the call instruction has to be entered as follows:

```
CFB H320.Init
CFB H320.Home
CFB H320.SetbrkPoint
```

Some elements in the FB also need the *H320.* designation. If an element specifies a module number then it needs the *H320.* designation. E.g. *H320.fOnDest_**1.***
Also see the section "Elements" on page 7-27 of this chapter.

For the FB Exec, the call instruction and the individual instruction (parameter 2) must have the *H320.* designation as follows:

```
CFB H320. Exec
K1
H320.GetActualPosition
R500
```

FB INIT:

The FB INIT has the following structure:
(values and symbols displayed are for presentation purposes only)

```
0$group H320
CFB  Init           ; Init H320 module
      K 1           ; Par. 1:  Axis Number (k1->k14)
      MecFactor_1   ; Par. 2:  Mechanic factor: 1.600000
      400           ; Par. 3:  Set proportional gain  (0..32767)
      25            ; Par. 4:  Set integral gain      (0..32767)
      10            ; Par. 5:  Set derivative gain    (0..32767)
      2000          ; Par. 6:  Integration limit      (0..65535)
      5             ; Par. 7:  Set derivative sampling time   (0..32767)
      200           ; Par. 8:  Set velocity feedforward gain  (0..32767)
      200           ; Par. 9:  Set accel feedforward gain     (0..32767)
      PosErrLimit_1 ; Par. 10: Position tolerance: 100  (User Unit)
      0             ; Par. 11: Position error behavior  (0:No stop,
                    ;                                    1:Stop movement)
      75            ; Par. 12: Tracking window  (User Unit)
      50            ; Par. 13: Settle window    (User Unit)
      100           ; Par. 14: Settle time      (0..10000 ms)
      0             ; Par. 15: Limit switch configuration (0:on, 1: off)
      0             ; Par. 16: Motion complete       (0:commanded, 1:actual)
      0             ; Par. 17: Profile mode          (0:Trapez.,1:Velocity,2
                    ;                          S-Curve,3:Elect. Gearing)
      0             ; Par. 18: SSI mode (0=no SSI; 1=250kHz;2=500kHz;3=1MHz)
      8             ; Par. 19: SSI nbr of bit  (8...26)
      0             ; Par. 20: SSI code(0=bin; 1=gray)
$endgroup
```

Unless otherwise stated, the unit of the values is based on the Mechanical Factor. (See Chapter 9)
The FB intialization file can be manually written, or can be generated automatically using the commissioning tool described in Chapter 10.
Additional information on the FB initialization file can be found in Appendix A of this manual. (page A-1)

FB HOME:

The FB HOME has the following structure:
(the values and symbols displayed are for presentation purposes only)

```
$group H320
CFB   Home          ; Homing H32x module
      K 1           ; Par. 1: Axis Number  (k1->k14)
      0             ; Par. 2: Reference search direction(0=down;1=up)
      1             ; Par. 3: Reference leave direction(0=down;1=up)
      HomeVelMin_1  ; Par. 4: Velocity minimum : 2000
      HomeVelMax_1  ; Par. 5: Velocity maximum : 5000



      10            ; Par. 6: Homing timeout : seconds
      1             ; Par. 7: Encoder index pulse.(0=No;1=Yes)
$endgroup
```

The FB homing file can be manually written, or can be generated automatically using the commissioning tool described in Chapter 10. The FB homing file has to be included in a block. (eg. COB, XOB, PB, etc)
Additional information on the FB homing file can be found in Appendix A of this manual. (page A-3)

FB EXEC:

There is several possibilities for the FB EXEC. Following is just a few examples:

```
$group H320
cfb    Exec
       K 1                 ; Par. 1: Axis number. k1->k14)
       GetActualPosition   ; Par. 2: Retrieve Actual position
       rActPos_1           ; Par. 3: Destination register for position

cfb    Exec
       K 1                 ; Par. 1: Axis number (k1->k14)
       GetEventStatus      ; Par. 2: Retrieve Event Status data
       rEventstatus_1      ; Par. 3: Destination for Event Status data

cfb    Exec
       k 1                 ; Par. 1: Axis number (k1->k14)
       StartMotion         ; Par. 2: Activates the move
       rTemporary          ; Par. 3: Axis where the move takes place
$endgroup
```

Three parameters must always be displayed even if only 2 are required. If a third parameter is not required, simply enter a dummy register.
A list of all instructions along with their functions for “FB Exec” can be found in Appendix A of this manual. (starting on page A-8)

FB SetBrkPoint:

The FB SetBrkPoint structure is as follows:

```
$group H320
cfb    SetBrkPoint
       k 1                  ; Par. 1: Axis the action takes place on
       SetBreakpoint_1      ; Par. 2: Breakpoint (1 or 2)
       SrceAxis_1           ; Par. 3: Axis the trigger event takes place on
       ActnUpdate           ; Par. 4: The action to be taken
       TrgPositiveActPos    ; Par. 5: Specifies the trigger event
       RPosition            ; Par. 6: Value of the trigger point
$endgroup
```

Up to two breakpoints can be set per axis.
The commissioning tool in Chapter 10 offers a feature for programming and testing breakpoints, then viewing the results on a performance curve.
A list of all instructions for “FB SetBrkPoint” can be found in Appendix A. (page A-6)

7.3 MBA File

The file is designated as D2H32x_B.MBA. (MBA = Module Base Address) and must be added manually to the program project file. The user must define the base addresses for the modules in the PCD. You will have to use a Text editor for this. (NotePad, TextPad, or even S-edit from PG5).
When you are assigning the base addresses for the H320, keep in mind the module takes 2 slots in the PCD. The PCD2 can support up to 7 modules (14 axis). The MBA file is displayed below:

```
; This file can be modified by the user
;;
; Basis addresses defined by the user
;; ----------------------------------
$group H320
NbrModules      EQU    1      ; No. of H320 modules used (0...7)


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

BA_1            EQU    64     ; Base address of module 1 (Axis 1 & 2)
BA_2            EQU    0      ; Base address of module 2 (Axis 3 & 4)
BA_3            EQU    0      ; Base address of module 3 (Axis 5 & 6)
BA_4            EQU    0      ; Base address of module 4 (Axis 7 & 8)
BA_5            EQU    0      ; Base address of module 5 (Axis 9 & 10)
BA_6            EQU    0      ; Base address of module 6 (Axis 11 & 12)
BA_7            EQU    0      ; Base address of module 7 (Axis 13 & 14)
$endgroup
```

In the above example, we have defined that 1 module is being used, and the base address is 64. Modules should be numbered consecutively starting with BA_1. The base addresses can be assigned as desired. For example:

```
NbrModules      EQU    3       ; No. of H320 modules used (0...7)

BA_1            EQU    64      ; Base address of module 1 (Axis 1 & 2)
BA_2            EQU    32      ; Base address of module 2 (Axis 3 & 4)
BA_3            EQU    96      ; Base address of module 3 (Axis 5 & 6)
BA_4            EQU    0       ; Base address of module 4 (Axis 7 & 8)
```

The base addresses for registers, flags, and FBs are assigned automatically and can be consulted in the resource list under “View” - “Resource List”.
Using the commissioning tool described in Chapter 10, the task of editing or creating the MBA file is greatly simplified.

7.4 Program Structure

The structure for the H320 section of your program is really dependent on the individual users method for entering FBs. Examples 1 and 2 are using “method one” and example 3 is using “method 2”, all of which are described under “Individual FBs” on page 7 – 2 of this chapter. Example 4 is the same as example 1, but we have added a SetBrkPoint FB to the program. **The examples are for presentation of the structure only. The parameters are according to the encoder and motor setup we have which can differ from the parameters that your setup requires. The programs are functional but they contain loops which is not a recommended programming procedure.**

Example 1: **Moving 1 axis forwards and backwards**

The basis of this program is that when Input 1 goes high, then a movement to the destination value 20000 and then back to 0 is activated. This movement will continue as long as Input 1 stays high. A high on Input 0 activates the homing segment.
Notice that all the FBs in the program are included between one set of $group H320 and $end-group directives. The example only has one module, but if you have several H320 modules, all the FBs for these modules can be placed in the same set of directives. It doesn’t matter as long as they are H320 modules. The important thing to remember is that all the FBs must be between a set of directives, whether it is a set specifically for each module or a set shared by multiple modules. The directive $include D2h320_b.equ must be at the beginning or at least prior to the module segment of your program.

```
$include D2h320_B.equ

;FB Initialisation of H320 module Axis 1-------------------------------------------

$group H320
$Xobseg 16

PosErrLimit_1 EQU  R                   ;;Position error limit value module nbr 1
MecFactor_1 EQU    R                   ;;Mecanic factor value module nbr 1

            LD     PosErrLimit_1       ;;Load position error limit value
                   100                 ;;Position error limit value

            LD     MecFactor_1         ;;Load Mecanic factor value
                   1.600000            ;;Mecanic factor value


            CFB    Init                ;;Init H320 module
                   K 1                 ;;Axis Number (k1->k14)
                   MecFactor_1         ;;Mecanic factor: 1.600000
                   100                 ;;Set proportional gain  (0..32767)
                   25                  ;;Set integral gain      (0..32767)
                   10                  ;;Set derivative gain    (0..32767)
                   1000                ;;Integration limit      (0..65535)
                   5                   ;;Set derivative sampling time     (0..32767)
                   150                 ;;Set velocity feedforward gain    (0..32767)
                   150                 ;;Set accel feedforward gain       (0..32767)
                   PosErrLimit_1       ;;Position tolerance: 100         (User Unit)
                   0                   ;;Position error behavior       (0:off, 1:on)
                   75                  ;;Tracking window    (User Unit)
                   50                  ;;Settle window      (User Unit)
                   1000                ;;Settle time        (0..10000 ms)
                   0                   ;;Limit switch configuration    (0:off, 1:on)
                   0                   ;;Motion complete mode(0:commanded,1:actual)
                   0                   ;;Profile mode     (0:Trapez.,1:velocity,2:S-
                                       ;;Curve,3:Elect. gearing)
                   0                   ;;SSI mode     (0=no SSI; 1=250kHz; 2=500kHz;
                                       ;; 3=1MHz)
                   8                   ;;SSI nbr of bit  (8...26)
                   0                   ;;SSI code  (0=bin; 1=gray)

$EndXobseg
```

```
;Homing Initailisation of H320 Module Axis 1-------------------------------------

ErrorDisplay EQU  PB                    ;;error handling for homing error
HomeVelMin_1 EQU   R                    ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                    ;;Ref Search velocity module nbr 1

            COB    0                    ;;call cob 0
                   0

            LD     HomeVelMin_1         ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1         ;;Load reference search velocity
                   3000

            STH    I 0                  ;;If input 0 is H call Homing FB
            JR     L Start              ;;If input = is L goto Main program
            RES    fEndHome_1           ;;Call FB Home Flag

            LDL    R 100                ;Acceleration
                   K 7000               ;Unit/s^2

            CFB    Exec
                   k 1                  ;Axis number
                   SetAcceleration      ;Acceleration for homing segment
                   R 100


Position:   CFB    Home                 ;;Homing H320 module
                   K 1                  ;;Axis Number
                   0                    ;;Reference search direction(0=left;1=right)
                   1                    ;;Reference leave direction(0=left;1=right)
                   HomeVelMin_1         ;;Velocity minimum : 1000
                   HomeVelMax_1         ;;Velocity maximum : 3000
                   15                   ;;Homing timeout
                   1                    ;;Encoder index pulse(0=No;1=Yes)
            STL    fEndHome_1           ;;If flag is L set Accu if H reset Accu
            JR     H Position           ;;If Accu is H wait for Homing to finish
            STH    fHomeErr_1           ;;Check Flag for Homing error
            CPB    H ErrorDisplay       ;;If Homing Error, display error message


;Main Program-------------------------------------------------------------------

Start:
            STH    I 1                ;start main program when I 1 goes High
            Jr     L End              ;wait for input 1 to go High

;Initial settings---------------------------------------------------------------
            LDL    R 100                ;Acceleration
                   K 10000              ;Unit/s^2

            CFB    Exec
                   k 1                  ;Axis number
                   SetAcceleration      ;Acceleration for main program
                   R 100

            LDL    R 101                ;Deceleration
                   K 10000              ;Unit/s^2
            CFB    Exec
                   k 1                  ;Axis number
                   SetDeceleration
                   R 101

            LDL    R 102                ;Velocity
                   5000                 ;Unit/s
            CFB    Exec
                   k 1                  ;Axis number
                   SetVelocity
                   R 102

;Set destination for forward movement-------------------------------------------

            LD     R 103
                   20000                ;destination value
CFB    Exec
                   K 1                  ;Axis number
                   SetPosition          ;Set destination
                   R 103
```

```
;Activate forward movement--------------------------------------------------------

            LDL    R 104
                   K 1                 ;axis for movement
            CFB    Exec
                   k 1                 ;Axis number
                   StartMotion         ;start movement
                   R 104

;Wait for destination to be reached-----------------------------------------------

Chkpnt:     CFB    Exec
                   K 1                 ;Axis number
                   GetEventStatus      ;read event status register
                   R 105               ;result of event status register

            STH    fOnDest_1           ;set H when destination is reached
            JR     L Chkpnt            ;wait for destination to be reached

;Set destination for reverse movement---------------------------------------------

            LD     R 106
                   0                   ;destination value
            CFB    Exec
                   K 1                 ;axis number
                   SetPosition         ;set destination
                   R 106

;Activate reverse movement--------------------------------------------------------

            LDL    R 104
                   K 1                 ;axis for movement
            CFB    Exec
                   k 1                 ;Axis number
                   StartMotion         ;start movement
                   R 104               ;

;Wait for destination to be reached-----------------------------------------------

Chkpnt1:    CFB    Exec
                   K 1                 ;Axis number
                   GetEventStatus      ;read event status register
                   R 105               ;results of event status register
            STH    fOnDest_1           ;set H when destination is reached
            JR     L Chkpnt1           ;wait for destination to be reached
End:            ECOB

;Handling of homing errors--------------------------------------------------------

            PB     ErrorDisplay
            Acc    H                   ;set acc H
            Out    O 112               ;output error code for homing error
            EPB
$endgroup
```

Example 2: Moving 1 axis with a H320 module and 1 axis with a H310 module

The basis of this program is that when Input 1 goes high, then a movement to the destination value 20000 and then back to 0 is activated. This movement will continue as long as Input 1 stays high. A high on Input 0 activates the homing segment. For the H310 module the destination values are the same. When Input 3 goes high the movement is activated and a high on Input 4 activates the homing segment.
When you are using two different types of modules in the same PCD, you have to designate a set of $group and $endgroup directives for each module type. Like with the example above, you can have all the H320 FBs in one set of directives, and all the H310 FBs in one set of directives. However in this example we are going to be using multiple sets of the $group and $endgroup directives. Also notice that at the beginning of the program, we have an $include directive for each module type.

```
$include D2h310_b.equ
$include D2h320_b.equ
```

```
$Xobseg 16

;FB Initialisation of H320 Module Axis 1-----------------------------------------

$group H320

PosErrLimit_1 EQU  R                    ;;Position error limit value module nbr 1
MecFactor_1 EQU    R                    ;;Mecanic factor value module nbr 1

           LD      PosErrLimit_1        ;;Load position error limit value
                   101                  ;;position error limit value

           LD      MecFactor_1          ;;Load Mecanic factor value
                   1.600000             ;;Mecanic factor value


           CFB     Init                 ;;Init H320 module
                   K 1                  ;;Axis Number (k1->k14)
                   MecFactor_1          ;;Mecanic factor: 1.600000
                   100                  ;;Set proportional gain  (0..32767)
                   25                   ;;Set integral gain      (0..32767)
                   10                   ;;Set derivative gain    (0..32767)
                   1000                 ;;Integration limit      (0..65535)
                   5                    ;;Set derivative sampling time     (0..32767)
                   150                  ;;Set velocity feedforward gain    (0..32767)
                   150                  ;;Set accel feedforward gain       (0..32767)
                   PosErrLimit_1        ;;Position tolerance: 100        (User Unit)
                   0                    ;;Position error behavior       (0:off, 1:on)
                   75                   ;;Tracking window    (User Unit)
                   50                   ;;Settle window      (User Unit)
                   1000                 ;;Settle time        (0..10000 ms)
                   0                    ;;Limit switch configuration    (0:off, 1:on)
                   0                    ;;Motion complete mode(0:commanded,1:actual)
                   0                    ;;Profile mode     (0:Trapez.,1:velocity,2:S-
                                        ;;Curve,3:Elect. gearing)
                   0                    ;;SSI mode     (0=no SSI; 1=250kHz; 2=500kHz;
                                        ;;3=1MHz)
                   8                    ;;SSI nbr of bit  (8...26)
                   0                    ;;SSI code  (0=bin; 1=gray)

$endgroup
;FB Initialisation for H310 Module Axis 1-----------------------------------------

$group H310

           LD      R 1000               ; Mechanic Factor
                   1.6                  ;
           LD      R 1001               ; Initial absolute speed
                   5000                 ;
           LD      R 1002               ;Initial absolute Accel.
                   10000

           CFB     Init                 ;Initialisation of FB
                   K 1                  ;axis number
                   100                  ; Proportional Factor
                   25                   ; Integrative Factor
                   10                   ; Derivative Factor
                   1000                 ; Integrative Limit
                   5                    ; Derivative sampling time
                   100                  ; Position Tolerance
                   0                    ; Action upon position error
                   R 1000               ;Mechanic Factor
                   R 1001               ; Initial velocity
                   R 1002               ; Initial accel.

$endXobseg
$endgroup

;Homing Initialisation for H320 Module Axis 1-------------------------------------

$group H320


ErrorDisplay1 EQU PB                    ;;Homing error handling
HomeVelMin_1 EQU   R                    ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                    ;;Ref Search velocity module nbr 1

           COB     0                    ;;call cob 0
                   0
```

```
            LD     HomeVelMin_1        ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1        ;;Load reference search velocity
                   3000

            STH    I 0                 ;;If input 0 is H call Homing FB
            JR     L Start1            ;;If input = is L goto Main program
            RES    fEndHome_1          ;;Call FB Home Flag


            LDL    R 100               ;Acceleration
                   K 7000              ;Unit/s^2

            CFB    Exec
                   k 1                 ;Axis number
                   SetAcceleration     ;acceleration for the homing segment
                   R 100


Position1:  CFB    Home                ;;Homing H310 module
                   K 1                 ;;Axis Number
                   0                   ;;Reference search direction(0=left;1=right)
                   1                   ;;Reference leave direction(0=left;1=right)
                   HomeVelMin_1        ;;Velocity minimum : 1000
                   HomeVelMax_1        ;;Velocity maximum : 3000
                   15                  ;;Homing timeout
                   1                   ;;Encoder index pulse(0=No;1=Yes)
            STL    fEndHome_1          ;;If flag is L set Accu if H reset Accu
            JR     H Position1         ;;If Accu is H wait for Homing to finish
            STH    fHomeErr_1          ;;Check Flag for Homing error
            CPB    H ErrorDisplay1     ;;If Homing Error, display error message
```
$endgroup

```
;Homing Initialisation for H310 Module Axis 1------------------------------------
```

$group H310

```
ErrorDisplay2 EQU PB                   ;;Homing error handling
HomeVelMin_1 EQU   R                   ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                   ;;Ref Search velocity module nbr 1

            LD     HomeVelMin_1        ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1        ;;Load reference search velocity
                   3000
RES    fEndHome_1           ;;Call FB Home Flag
            STH    I 4                 ;;If input 4 is H call Homing FB
            JR     L Start2            ;;If input = is L goto Main program

Position2:  CFB    H Home              ;;Homing H310 module
                   K 1                 ;;Axis Number
                   0                   ;;Reference search direction(0=left;1=right)
                   1                   ;;Reference leave direction(0=left;1=right)
                   HomeVelMin_1        ;;Velocity minimum : 1000
                   HomeVelMax_1        ;;Velocity maximum : 3000
                   15                  ;;Homing timeout
                   I 43                ;;Reference Input
            STL    fEndHome_1          ;;If flag is L set Accu if H reset Accu
            JR     H Position2         ;;If Accu is H wait for Homing to finish
            STH    fHomeErr_1          ;;Check Flag for Homing error
            CPB    H ErrorDisplay2     ;;If Homing Error, display error message
```

$endgroup

```
;Main Program for H320 Module Axis 1--------------------------------------------
```

$group H320

```
Start1:
            STH    I 1                 ;start main program when I 1 is High
            Jr     L Start2               ;wait for input 1 to go High

;Initial settings-------------------------------------------------------------------

            LDL    R 100               ;Acceleration
                   K 10000             ;Unit/s^2
            CFB    Exec
                   k 1                  ;Axis number
                   SetAcceleration      ;acceleration for the main program
                   R 100

            LDL    R 101                ;Deceleration
                   K 10000              ;Unit/s^2

            CFB    Exec
                   k 1                  ;Axis number
                   SetDeceleration
                   R 101

            LDL    R 102                ;Velocity
                   5000                 ;Unit/s
            CFB    Exec
                   k 1                  ;Axis number
                   SetVelocity
                   R 102

;Set destination for forward movement----------------------------------------------

            LD     R 103
                   20000                ;destination value
            CFB    Exec
                   K 1                  ;Axis number
                   SetPosition          ;Set destination
                   R 103

;Activate forward movement---------------------------------------------------------

                   LDL    R 104
                   K 1                  ;axis for movement

            CFB    Exec
                   k 1                  ;Axis number
                   StartMotion          ;start movement
                   R 104                ;

;Wait for destination to be reached------------------------------------------------

Chkpnt1:    CFB    Exec
                   K 1                  ;Axis number
                   GetEventStatus       ;read event status register
                   R 105                ;results of status register

            STH    fOnDest_1            ;set H when destination is reached
            JR     L Chkpnt1            ;wait for destination to be reached

;set destination for reverse movement----------------------------------------------

            LD     R 106
                   0                    ;destination value

                   CFB    Exec
                   K 1                  ;axis number
                   SetPosition          ;set destination
                   R 106

;Activate reverse movement---------------------------------------------------------

            LDL    R 104
                   K 1                  ;axis for movement
            CFB    Exec
                   k 1                  ;Axis number
                   StartMotion          ;start movement
                   R 104                ;

;Wait for destination to be reached------------------------------------------------

Chkpnt11:   CFB    Exec
                   K 1                  ;Axis number
                   GetEventStatus       ;read event status register
                   R 105                ;results of status register
            STH    fOnDest_1            ;set H when destination is reached
            JR     L Chkpnt11           ;wait for destination to be reached
$endgroup
```

```
;Main Program for H310 Module Axis 1------------------------------------------

$group  H310

Start2:
            STH    I 3
            Jr     L End

;Set destination for forward movement-----------------------------------------

            LD     R 107
                   20000              ;destination value

            CFB    Exec
                   K 1                ;Axis number
                   LdDestAbs          ;Set destination
                   R 107
;Activate forward movement----------------------------------------------------

            CFB    Exec
                   K 1                ;Axis number
                   StartMot           ;activate the movement
                   rNotUsed           ;


;Wait for destination to be reached-------------------------------------------

Chkpnt2:    CFB    Exec
                   K 1                ;Axis number
                   RdStatRg           ;read contents of status register
                   R 108              ;results of status register

            STH    fOnDest_1          ;set high when destination is reached
            JR     L Chkpnt2          ;wait until destination is reached

;Set destination for reverse movement-----------------------------------------

            LD     R 107
                   0                  ;destination value
            CFB    Exec
                   K 1                ;axis number
                   LdDestAbs          ;set destination
                   R 107

;Activate reverse movement----------------------------------------------------

            CFB    Exec
                   K 1               ;Axis number
                   StartMot          ;start movement
                   rNotUsed          ;

;Wait for destination to be reached-------------------------------------------

 Chkpnt22:  CFB    Exec
                   K 1               ;Axis number
                   RdStatRg          ;read contents of status register
                   R 108             ;results of status register

                   STH    fOnDest_1   ;set high when destination is reached
                   JR     L Chkpnt22  ;wait for destination to be reached
End:
           ECOB
;Error handling for homing errors---------------------------------------------

                   PB   ErrorDisplay2 ;error handling PB H310
                   Acc     H          ;set Acc H
                   Out     O 113      ;error for Homing module
                   EPB
$endgroup

$group H320
                   PB  Error Display1 ;error handling PB H320
                   Acc       H        ;set Acc H
                   Out       O 112    ;error for Homing module
                   EPB
$endgroup
```

Example 3: Alternate programming of 1 axis moving forwards and backwards

Example 3 is the same as example 1 except in example 3 we are using method 2 for entering the program code. (method 2 is described on page 7-2 of this chapter)

```
$include D2h320_B.equ

ErrorDisplay EQU  PB                  ;;error handling for homing error
HomeVelMin_1 EQU   R                  ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                  ;;Ref Search velocity module nbr 1
PosErrLimit_1 EQU  R                  ;;Position error limit value module nbr 1
MecFactor_1 EQU    R                  ;;Mecanic factor value module nbr 1

;------------------------------------------------------------------------------
;FB Initialisation of H320 module Axis 1

$Xobseg 16

            LD     PosErrLimit_1      ;;Load position error limit value
                   101                ;;position error limit value

            LD     MecFactor_1        ;;Load Mecanic factor value
                   1.600000           ;;Mecanic factor value


            CFB    H320.Init          ;;Init H320 module
                   K 1                ;;Axis Number (k1->k14)
                   MecFactor_1        ;;Mecanic factor: 1.600000
                   100                ;;Set proportional gain  (0..32767)
                   25                 ;;Set integral gain      (0..32767)
                   10                 ;;Set derivative gain    (0..32767)
                   1000               ;;Integration limit      (0..65535)
                   5                  ;;Set derivative sampling time     (0..32767)
                   150                ;;Set velocity feedforward gain    (0..32767)
                   150                ;;Set accel feedforward gain       (0..32767)
                   PosErrLimit_1      ;;Position tolerance: 100        (User Unit)
                   0                  ;;Position error behavior      (0:off, 1:on)
                   75                 ;;Tracking window    (User Unit)
                   50                 ;;Settle window      (User Unit)
                   1000               ;;Settle time        (0..10000 ms)
                   0                  ;;Limit switch configuration    (0:off, 1:on)
                   0                  ;;Motion complete mode(0:commanded,1:actual)
                   0                  ;;Profile mode     (0:Trapez.,1:velocity,2:S-
                                      ;;Curve,3:Elect. gearing)
                   0                  ;;SSI mode     (0=no SSI; 1=250kHz; 2=500kHz;
                                      ;; 3=1MHz)
                   8                  ;;SSI nbr of bit  (8...26)
                   0                  ;;SSI code  (0=bin; 1=gray)

$EndXobseg
;-------------------------------------------------------------------------------
;Homing Initialisation of H320 Module Axis 1

            COB    0                  ;;call cob 0
                   0

            LD     HomeVelMin_1       ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1       ;;Load reference search velocity
                   3000

            STH    I 0                ;;If input 0 is H call Homing FB
            JR     L Start            ;;If input = is L goto Main program
            RES    H320.fEndHome_1    ;;Call FB Home Flag

            LDL    R 100                ;Acceleration
                   K 7000               ;Unit/s^2

            CFB    H320.Exec
                   K 1                  ;Axis number
                   H320.SetAcceleration  ;aceleration for the homing segment
                   R 100


Position:   CFB    H320.Home          ;;Homing H310 module
                   K 1                ;;Axis Number
                   0                  ;;Reference search direction(0=left;1=right)
                   1                   ;;Reference leave direction(0=left;1=right)
                   HomeVelMin_1        ;;Velocity minimum : 1000
                   HomeVelMax_1        ;;Velocity maximum : 3000
                   15                  ;;Homing timeout
                   1                   ;;Encoder index pulse(0=No;1=Yes)
            STL    H320.fEndHome_1     ;;If flag is L set Accu if H reset Accu
            JR     H Position          ;;If Accu is H wait for Homing to finish
            STH    H320.fHomeErr_1     ;;Check Flag for Homing error
            CPB    H ErrorDisplay      ;;If Homing Error, display error message


;-----------------------------------------------------------------------------------
;Main Program

Start:
            STH    I 1                   ;start main program when I 1 goes High
            Jr     L End                 ;wait for I 1 to go High



;Initial settings------------------------------------------------------------------

            LDL    R 100                 ;Acceleration
                   K 10000               ;Unit/s^2

            CFB    H320.Exec
                   K 1                   ;Axis number
                   H320.SetAcceleration  ;Acceleration for the main program
                   R 100

            LDL    R 101                 ;Deceleration
                   K 10000               ;Unit/s^2

                   CFB    H320.Exec
                   K 1                   ;Axis number
                   H320.SetDeceleration
                   R 101


            LDL    R 102                 ;Velocity
                   5000                  ;Unit/s
            CFB    H320.Exec
                   k 1                   ;Axis number
                   H320.SetVelocity
                   R 102


;set the destination for forward movement------------------------------------------

            LD     R 103
                   20000                 ;destination value
            CFB    H320.Exec
                   K 1                   ;Axis number
                   H320.SetPosition      ;Set destination
                   R 103

;Activate forward movement---------------------------------------------------------

            LDL    R 104
                   K 1                   ;axis for movement
            CFB    H320.Exec
                   K 1                   ;Axis number
                   H320.StartMotion      ;start movement
                   R 104                 ;
;Wait for destination to be reached------------------------------------------------

Chkpnt:     CFB    H320.Exec
                   K 1                   ;Axis number
                   H320.GetEventStatus   ;read event status register
                   R 105

            STH    H320.fOnDest_1        ;set H when destination is reached
            JR     L Chkpnt              ;wait for destination to be reached

;Set destination for reverse movement----------------------------------------------

            LD     R 106
                   0                      ;destination value
            CFB    H320.Exec
                   K 1                    ;axis number
                   H320.SetPosition       ;set destination
                   R 106

;Activate reverse movement---------------------------------------------------

            LDL    R 104
                   K 1                    ;axis for movement
            CFB    H320.Exec
                   k 1                    ;Axis number
                   H320.StartMotion       ;start movement
                   R 104                  ;

;Wait for destination to be reached------------------------------------------

Chkpnt1:    CFB    H320.Exec
                   K 1                    ;Axis number
                   H320.GetEventStatus    ;read event status register
                   R 105                  ;results event status register

            STH    H320.fOnDest_1         ;set H when destination is reached
            JR     L Chkpnt1              ;wait for destination to be reached


End:
            ECOB
;Error handling for homing errors--------------------------------------------

            PB  ErrorDisplay
            Acc      H                    ;set Acc H
            Out      O 112                ;output error code for homing error
            EPB
```

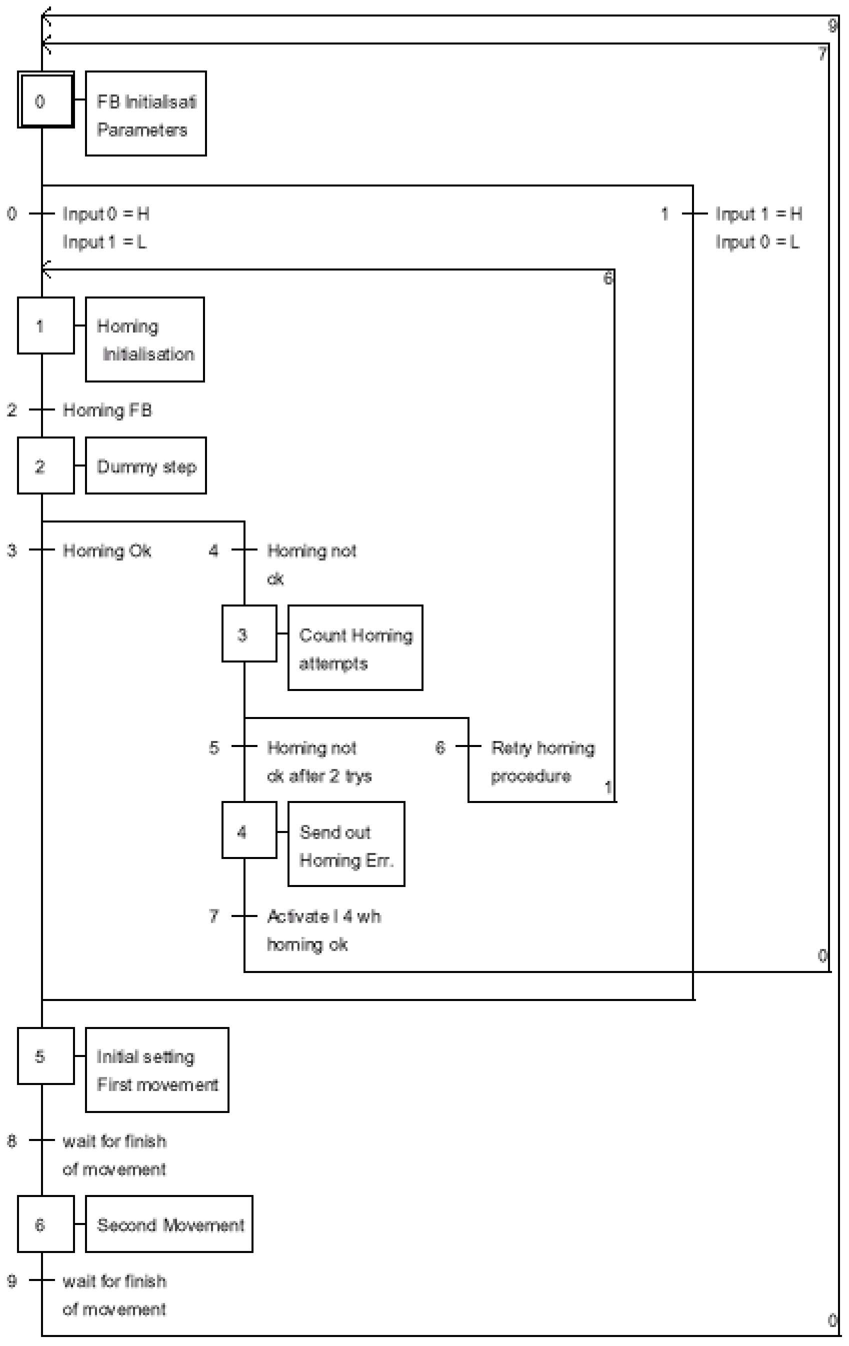
Example 4: Introducing Breakpoints

In this example we have taken the same program as example 1, changed the destination value, and added a couple of breakpoints to it. The first breakpoint is setup to change velocity from 5000 to 20000 (um/s) when the movement reaches 10 mm on its way to 40 mm. The second breakpoint is setup to change velocity from 5000 to 20000 (um/s) when the movement reaches 35 mm starting from 40 mm on its way to 0. For a list of possible instructions and additional information on the SetBrkPoint FB, see Appendix A, page A – 5 & 6 of this manual.

```
$include D2h320_B.equ

;----------------------------------------------------------------------------
;FB Initialisation of H320 module Axis 1

$group H320
$Xobseg 16

PosErrLimit_1 EQU  R                   ;;Position error limit value module nbr 1
MecFactor_1 EQU    R                   ;;Mecanic factor value module nbr 1

            LD     PosErrLimit_1       ;;Load position error limit value
                   100                 ;;position error limit value

            LD     MecFactor_1         ;;Load Mecanic factor value
                   1.600000            ;;Mecanic factor value


            CFB    Init                ;;Init H320 module
                   K 1                 ;;Axis Number (k1->k14)
                   MecFactor_1         ;;Mecanic factor: 1.600000
                   100                 ;;Set proportional gain  (0..32767)
                   25                  ;;Set integral gain      (0..32767)
                   10                  ;;Set derivative gain    (0..32767)
                   1000                ;;Integration limit      (0..65535)
                   5                   ;;Set derivative sampling time     (0..32767)
                   150                 ;;Set velocity feedforward gain    (0..32767)
                   150                 ;;Set accel feedforward gain       (0..32767)
                   PosErrLimit_1       ;;Position tolerance: 100         (User Unit)
                   0                   ;;Position error behavior       (0:off, 1:on)
                   75                  ;;Tracking window    (User Unit)
                   50                  ;;Settle window      (User Unit)
                   1000                ;;Settle time        (0..10000 ms)
                   0                   ;;Limit switch configuration    (0:off, 1:on)
                   0                   ;;Motion complete mode(0:commanded,1:actual)
                   0                   ;;Profile mode     (0:Trapez.,1:velocity,2:S-
                                       ;;Curve,3:Elect. gearing)
                   0                   ;;SSI mode     (0=no SSI; 1=250kHz; 2=500kHz;
                                       ;; 3=1MHz)
                   8                   ;;SSI nbr of bit  (8...26)
                   0                   ;;SSI code  (0=bin; 1=gray)

$EndXobseg

;---------------------------------------------------------------------------------
;Homing Initailisation of H320 Module Axis 1

ErrorDisplay EQU   PB
HomeVelMin_1 EQU   R                   ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                   ;;Ref Search velocity module nbr 1

            COB    0                   ;;call cob 0
                   0



            LD     HomeVelMin_1        ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1        ;;Load reference search velocity
                   3000

            STH    I 0                 ;;If input 0 is H call Homing FB
            JR     L Start             ;;If input = is L goto Main program
            RES    fEndHome_1          ;;Call FB Home Flag


            LDL    R 100               ;Acceleration
                   K 7000              ;Unit/s^2

            CFB    Exec
                   k 1                 ;Axis number
                   SetAcceleration     ;acceleration for the homing segment
                   R 100

Position:   CFB    Home                ;;Homing H310 module
                   K 1                 ;;Axis Number
                   0                   ;;Reference search direction(0=left;1=right)
                   1                   ;;Reference leave direction(0=left;1=right)
                   HomeVelMin_1        ;;Velocity minimum : 1000
                   HomeVelMax_1        ;;Velocity maximum : 3000
                   15                  ;;Homing timeout
                   1                   ;;Encoder index pulse(0=No;1=Yes)
            STL    fEndHome_1          ;;If flag is L set Accu if H reset Accu
            JR     H Position          ;;If Accu is H wait for Homing to finish
            STH    fHomeErr_1          ;;Check Flag for Homing error
            CPB    H ErrorDisplay      ;;If Homing Error, display error message

;---------------------------------------------------------------------------------
;Main Program

Start:
            STH    I 1                 ;start main program when I 1 goes High
            Jr     L End               ;wait for input 1 to go High

;Initial settings-----------------------------------------------------------------

            LDL    R 100               ;Acceleration
                   K 10000             ;Unit/s^2

            CFB    Exec
                   k 1                 ;Axis number
                   SetAcceleration     ;acceleration for the main program
                   R 100

            LDL    R 101               ;Deceleration
                   K 10000             ;Unit/s^2
            CFB    Exec
                   k 1                 ;Axis number
                   SetDeceleration
                   R 101
            LDL    R 102               ;Velocity
                   5000                ;Unit/s
            CFB    Exec
                   k 1                 ;Axis number
                   SetVelocity
                   R 102

;Set the destination for forward movement------------------------------------------

            LD     R 103
                   40000               ;destination value
            CFB    Exec
                   K 1                 ;Axis number
                   SetPosition         ;Set destination
                   R 103



;Define Breakpoint parameters------------------------------------------------------

            LD     R 200
                   10000               ;define value for 1st breakpoint
            LD     R 201
                   SrceAxis_1          ;define register for Source axis
            LD     R 202
                   ActnUpdate          ;define register for Action
            LD     R 203
                   TrgActPosCrossed    ;define register for trigger point
            LD     R 204               ;define value for 2nd breakpoint
                   35000

;Activate the forward movement-----------------------------------------------------

            LDL    R 104
                   K 1                 ;axis for movement

            CFB    Exec
                   k 1                 ;Axis number
                   StartMotion         ;start movement
                   R 104

;Set velocity that will be updated bythe first breakpoint----------------------------

            LD     R 302               ;20mm/s
                   20000
            CFB    Exec
                   k 1                 ;Axis number
                   SetVelocity
                   R 302

;Configure the first breakpoint-------------------------------------------------------

            CFB    SetBrkPoint
                   K 1                 ;Axis number
                   SetBreakpoint_1     ;set breakpoint 1
                   R 201
                   R 202
                   R 203
                   R 200

;Wait for destination of forward movement to be reached----------------------------

Chkpnt:     CFB    Exec
                   K 1                 ;Axis number
                   GetEventStatus      ;read event status register
                   R 105               ;result of event status register
            STH    fOnDest_1           ;set H when destination is reached
            JR     L Chkpnt            ;wait for destination to be reached

;Set the velocity for the beginning of the reverse movement------------------------

            CFB    Exec
                   k 1                 ;Axis number
                   SetVelocity
                   R 102

;Set the destination for the reverse movement

            LD     R 106
                   0                   ;destination value
            CFB    Exec
                   K 1                ;axis number
                   SetPosition        ;set destination
                   R 106
;Activate the reverse movement-----------------------------------------------------

            LDL    R 104
                   K 1                ;axis for movement
            CFB    Exec
                   k 1                ;Axis number
                   StartMotion        ;start movement
                   R 104              ;

;Set the velocity that will be updated by the second breakpoint-----------------------

            CFB    Exec
                   k 1                ;Axis number
                   SetVelocity
                   R 302
;Configure the second breakpoint------------------------------------------------------

            CFB    SetBrkPoint
                   k 1                ;Axis number
                   SetBreakpoint_1    ;set breakpoint 1
                   R 201
                   R 202
                   R 203
                   R 204

;Wait for destination of reverse movement to be reached---------------------------

Chkpnt1:    CFB    Exec
                   K 1                ;Axis number
                   GetEventStatus     ;read event status register
                   R 105              ;results of event status register

            STH    fOnDest_1          ;set H when destination is reached
            JR     L Chkpnt1          ;wait for destination to be reached

End:
            ECOB

;Error handling for homing errors--------------------------------------------------

            PB     ErrorDisplay
            Acc    H                  ;set acc H
            Out    O 112              ;output error code for homing error
            EPB
$endgroup
```

7.5 Program Structure (Graftec)

As mentioned at the beginning of the chapter, because motion control tasks are sequential processes, it is preferable that you write them in Graftec.
For the example displayed below, we have taken “example 1” from the previous section, and put it into a Graftec structure. The $Include D2h320_b.EQU and the $Group H320 have been placed in the Initial Step. The $endgroup is in the last transition. For this example, we have also added some additional segments for the handling of a Homing Error.
The structure is displayed first, followed by a listing of the code.



```
SB     0
;-------------------------------
            IST    0                    ;FB Initialisation Parameters
                   I 9                  ;wait for finish of movement
                   I 7                  ;Activate I 4 when homing ok
                   O 0                  ;Input 0 = H Input 1 = L
                   O 1                  ;Input 1 = H Input 0 = L
$include D2H320_b.EQU
$include D2h110_b.equ
$group H320
;------------------------------------------------------------------------------------
;FB Initialisation of H320 module Axis 1

HomeVelMin_1 EQU   R                    ;;Ref leave velocity module nbr 1
HomeVelMax_1 EQU   R                    ;Ref search velocity module nbr 1
PosErrLimit_1 EQU  R                    ;;Position error limit value module nbr 1
MecFactor_1 EQU    R                    ;;Mecanic factor value module nbr 1

            LD     PosErrLimit_1        ;;Load position error limit value
                   100                  ;;position error limit value

            LD     MecFactor_1          ;;Load Mecanic factor value
                   1.600000             ;;Mecanic factor value


            CFB    Init                 ;;Init H320 module
                   K 1                  ;;Axis Number (k1->k14)
                   MecFactor_1          ;;Mecanic factor: 1.600000
                   150                  ;;Set proportional gain  (0..32767)
                   25                   ;;Set integral gain      (0..32767)
                   10                   ;;Set derivative gain    (0..32767)
                   1000                 ;;Integration limit      (0..65535)
                   5                    ;;Set derivative sampling time     (0..32767)
                   150                  ;;Set velocity feedforward gain    (0..32767)
                   150                  ;;Set accel feedforward gain       (0..32767)
                   PosErrLimit_1        ;;Position tolerance: 100         (User Unit)
                   0                    ;;Position error behavior       (0:off, 1:on)
                   75                   ;;Tracking window    (User Unit)
                   50                   ;;Settle window      (User Unit)
                   1000                 ;;Settle time        (0..10000 ms)
                   0                    ;;Limit switch configuration    (0:off, 1:on)
                   0                    ;;Motion complete mode(0:commanded,1:actual)
                   0                    ;;Profile mode     (0:Trapez.,1:velocity,2:S-
                                        ;;Curve,3:Elect. gearing)
                   0                    ;;SSI mode     (0=no SSI; 1=250kHz; 2=500kHz;
                                        ;; 3=1MHz)
                   8                    ;;SSI nbr of bit  (8...26)
                   0                    ;;SSI code  (0=bin; 1=gray)

            LDL    R 100                ;Acceleration
                   K 35000              ;Unit/s^2

            CFB    Exec
                   k 1                  ;Axis number
                   SetAcceleration
                   R 100

            LDL    R 101                ;Deceleration
                   K 35000              ;Unit/s^2
            CFB    Exec
                   k 1                  ;Axis number
                   SetDeceleration
                   R 101

            LD     R 1                  ;Reset register 1 to 0 this is the counter
                   0                    ;to limit the homing attemps to 2
            EST                         ;0
;-------------------------------
            ST     1                    ;Homing Initialisation
                   I 0                  ;Input 0 = H Input 1 = L
                   I 6                  ;Retry homing procedure
                   O 2                  ;Homing FB
;Homing Initailisation of H320 Module Axis 1
            LD     HomeVelMin_1         ;;Load reference leave velocity
                   1000
            LD     HomeVelMax_1         ;;Load reference search velocity
                   3000
            Acc    H                    ;;set acc H
            RES    fEndHome_1           ;;Reset flag
            RES    fHomeErr_1
            EST                         ;1
;--------------------------------
            ST      2                   ;Dummy step
                    I 2                 ;Homing FB
                    O 3                 ;Homing Ok
                    O 4                 ;Homing not ok

            EST                         ;2
;--------------------------------
            ST      3                   ;Count Homing attempts
                    I 4                 ;Homing not ok
                    O 5                 ;Homing not ok after 2 trys
                    O 6                 ;Retry homing procedure
            Inc     R 1                 ;increment register 1

            Cmp     R 1                 ; compare register 1 to
                    K 2                 ;nbr 2


            EST                         ;3
;--------------------------------
            ST      4                   ;Send out Homing Err.
                    I 5                 ;Homing not ok after 2 trys
                    O 7                 ;Activate I 4 when homing ok
            Acc     H                   ; set acc H
            Out     O 113               ; simulates outputting error code

            EST                         ;4
;--------------------------------
            ST      5                   ;Initial settings First movement
                    I 3                 ;Homing Ok
                    I 1                 ;Input 1 = H Input 0 = L
                    O 8                 ;wait for finish of movement
;Main Program


            LDL     R 100               ;Acceleration
                    K 35000             ;Unit/s^2

            CFB     Exec
                    k 1                 ;Axis number
                    SetAcceleration
                    R 100

            LDL     R 101               ;Deceleration
                    K 35000             ;Unit/s^2
            CFB     Exec
                    k 1                 ;Axis number
                    SetDeceleration
                    R 101
            LDL     R 102               ;Velocity
                    20000               ;Unit/s
            CFB     Exec
                    k 1                 ;Axis number
                    SetVelocity
                    R 102

            LD      R 103
                    20000               ;destination value
            CFB     Exec
                    K 1                 ;Axis number
                    SetPosition         ;Set destination
                    R 103
            LDL     R 104
                    K 1                 ;axis for movement
            CFB     Exec
                    k 1                 ;Axis number
                    StartMotion         ;start movement
                    R 104

            EST                         ;5
;--------------------------------
            ST      6                   ;Second Movement
                    I 8                 ;wait for finish of movement
                    O 9                 ;wait for finish of movement
            LD      R 106
                    0                   ;destination value
            CFB     Exec
                    K 1                 ;axis number
```

```
              SetPosition          ;set destination
              R 106

       LDL    R 104
              K 1                  ;axis for movement
       CFB    Exec
              k 1                  ;Axis number
              StartMotion          ;start movement
              R 104                ;

       EST                         ;6
;--------------------------------
       TR     0                    ;Input 0 = H Input 1 = L
              I 0                  ;FB Initialisation Parameters
              O 1                  ;Homing Initialisation
       STH    I 0                  ;Wait for I 0 to be high
       ANL    I 1                  ;and I 1 to be low

       ETR                         ;0
;--------------------------------
       TR     1                    ;Input 1 = H Input 0 = L
              I 0                  ;FB Initialisation Parameters
              O 5                  ;Initial settings First movement
       STH    I 1                  ;wait for I 1 to be High
       ANL    I 0                  ;and I 0 to be low

       ETR                         ;1
;--------------------------------
       TR     2                    ;Homing FB
              I 1                  ;Homing Initialisation
              O 2                  ;Dummy step

       CFB    Home                 ;;Homing H320 module
              K 1                  ;;Axis Number
              0                    ;;Reference search direction(0=left;1=right)
              1                    ;;Reference leave direction(0=left;1=right)
              HomeVelMin_1         ;;Velocity minimum : 1000
              HomeVelMax_1         ;;Velocity maximum : 3000
              15                   ;;Homing timeout
              1                    ;;Encoder index pulse(0=No;1=Yes)

       STH    fEndHome_1           ;; when H set Acc

       ETR                         ;2
;--------------------------------
       TR     3                    ;Homing Ok
              I 2                  ;Dummy step
              O 5                  ;Initial settings First movement
       STH    fEndHome_1           ;; Set Acc
       ANL    fHomeErr_1           ;;Check Flag for Homing error

       ETR                         ;3
;--------------------------------
       TR     4                    ;Homing not ok
              I 2                  ;Dummy step
              O 3                  ;Count Homing attempts
       STH    fEndHome_1           ;wait for flag to go high
       ANH    fHomeErr_1           ;wait for flag to go high

       ETR                         ;4
;--------------------------------
       TR     5                    ;Homing not ok after 2 trys
              I 3                  ;Count Homing attempts
              O 4                  ;Send out Homing Err.
       Acc    Z                    ;set acc to Zero flag
       Out    F 0                  ;set F 0 to acc
       STH    F 0                  ;start if F 0 is set

       ETR                         ;5
;--------------------------------
       TR     6                    ;Retry homing procedure
              I 3                  ;Count Homing attempts
              O 1                  ;Homing Initialisation
       Acc    N                    ; set acc to negative flag
       Out    F 1                  ;set F 1 to acc
       STH    F 1                  ;continue if F 1 is high

       ETR                         ;6
;--------------------------------
       TR     7                    ;Activate I 4 when homing ok
```

```
            I 4                 ;Send out Homing Err.
            O 0                 ;FB Initialisation Parameters
     LD     R 1                 ;Reset register 1 to 0
            0

     STH    I 4                 ;(fix the homing error) and activate I 4 to
                                ;continue
     ANL    I 0                 ;homing channel not active
     ANL    I 1                 ;main program not active
     RES    O 113               ;reset homing error output

     ETR                        ;7
;-------------------------------
     TR     8                   ;wait for finish of movement
            I 5                 ;Initial settings First movement
            O 6                 ;Second Movement
     CFB    Exec
            K 1                 ;Axis number
            GetEventStatus      ;read event status register
            R 105               ;results of event status register

     STH    fOnDest_1           ;set H when destination is reached

     ETR                        ;8
;-------------------------------
     TR     9                   ;wait for finish of movement
            I 6                 ;Second Movement
            O 0                 ;FB Initialisation Parameters
     CFB    Exec
            K 1                 ;Axis number
            GetEventStatus      ;read event status register
            R 105               ;results of event status register

     STH    fOnDest_1           ;set H when destination is reached


$endgroup

     ETR                        ;9

     ESB                        ;0
```

7.6 Elements

The following elements can be queried by the user:
(the lists below are an example for module 1)

Inputs	**Description**
REF_1s2	REFerence switch
LS1_1s2	Limit Switch 1
LS2_1s2	Limit Switch 2
AxisSelect_1_2 (output)	RES = Axis 1, SET = Axis 2
AxisIn_1s2	Status of axis synchronisation Input
AxisOut_1s2	Status of axis synchronisation Output
AxisEvent_1_2	Axis event interrupt
PowerError_1_2	Power interne error
PowerEncError_1_2	Power encoder error
CableBreak_1s2	Cable break
SSI_timeout_1s2	SSI timeout
OK_LED_1_2	Status of OK led
HostIOError_1_2	Host IO error

(_1s2 indicates the axis is selected via the “Axis Select” output)
(_1_2 indicates the whole module)

Flags	**Description**
fHomeErr_x	Error during home procedure.
fEndHome_x	Home procedure executed.
fBrkPt1_x	First break point reached.
fBrkPt2_x	Second break point reached.
fOnDest_x	Destination point reached.
fPosErr_x	Excessive error position.
fWrapAround_x	Position overFlow
fCaptureTrig_x	Set to 1 when a position capture occurs.

(_x corresponds to the axis number)

Global (for all modules)	**Description**
rDiag (register)	Error diagnostic register
fPar_Err (flag)	Set high when an entered parameter in the FB Init or FB Homing files is out of range.

Always keep in mind that all the above symbols must always be in a $group H320 declaration and therefore, the complete symbol name is always *H320.symbol*. You can see this in: <View> <Data List> of the SAIA Project Manager.

This page is for information purposes only. These hardware specific mechanisms are accessed via the FBs and programming elements as described prior to this section.

7.7 Bus Interface

Since the PCD2.H32x is a double module, there are also 2*16 bus addresses mapped.

Address mapping:

Address	**Data read** (inputs)	**Data write** (outputs)
0	Data bit 0 (LSB)	Data bit 0 (LSB)
1	Data bit 1	Data bit 1
2	Data bit 2	Data bit 2
3	Data bit 3	Data bit 3
4	Data bit 4	Data bit 4
5	Data bit 5	Data bit 5
6	Data bit 6	Data bit 6
7	Data bit 7	Data bit 7
8	Data bit 8	Data bit 8
9	Data bit 9	Data bit 9
10	Data bit 10	Data bit 10
11	Data bit 11	Data bit 11
12	Data bit 12	Data bit 12
13	Data bit 13	Data bit 13
14	Data bit 14	Data bit 14
15	Data bit 15 (MSB)	Data bit 15 (MSB)
16	HostRdy ●	HostSlct ●
17	HostIOError	HostWrite ●
18	Axis Event Interrupt	HostRead ●
19	Power int. (±15V) Error	HostCmd ●
20	Power 5V Encoder Error	Axis select (0=axis 1; 1=axis 2)
21	SSI Busy 1 / 2 * ●	
22	SSI data available 1 / 2 * ●	
23	SSI_Timeout *	
24	OK-LED	SSI read low word ●
25		SSI read high word ●
26	Cablebreak Error 1 / 2 *	FPGA Write ●
27	LS1 1 / 2 *	Pointer Select ●
28	Ref 1 / 2 *	Pointer Address 0 ●
29	LS2 1 / 2 *	Pointer Address 1 ●
30	AxisIn 1 / 2 *	Pointer Address 2 ●
31	AxisOut 1 / 2 *	Pointer Address 3 ●

* These inputs are multiplexed with the Axis select output.
● Only used internally.

Notes

8. Error handling and diagnosis

8.1 Definition error checked by assembler

The following definition errors in file D2H320_B.MBA are checked during assembly:

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

"Remark: No H320 used (NbrModules = 0 in D2H320_B.MBA)"

- If the number of modules (NbrModules) is > 7, no code is assembled and the following error message is written in the 'Make' window:

"Error : more than 7 Modules H320 defined (NbrModules = 0...7)"

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

"Symbol not defined 'H320. SetVellocity "
(where the expression 'H320' is generated by $group h320)

- If the definition $group H320 is missing, the assembler reports:

"Symbol not defined"

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

8.2 Error handling in Run

8.2.1 Incorrect parameter

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

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

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

The error code is made up as follows:

```
rDiag    bit 31 . . . . . . . 24 23 . . . . . . . 16 15 . . . . . . .  8 7 . . . . . . . 0
              \  Reserve  /      \   FB no.  /     \  Par. no.  /    \ axis. no./
                                    (Init = FB 1)
                                    (Exec = FB 2)
                                    (Home = FB 3)
                                    (SetBrkPoint = FB 4)
```

Example: If the sample time (parameter 6) in FB 'Init' of module 2 is incorrectly defined (>32767), 'rDiag' is loaded with the hex value 00 01 06 02.

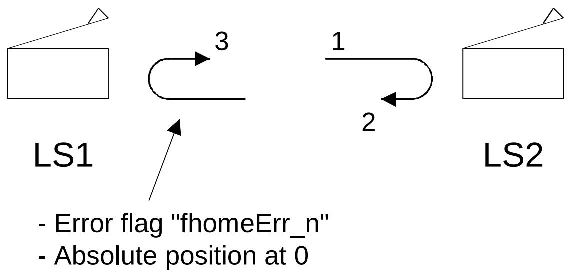
At each incorrect parameter, the diagnostic register is overwritten and always contains the last error. It should therefore be evaluated as soon as the 'fPar_Err' flag signals a range error. The absolute addresses of 'rDiag' and 'fPar_Err' can be seen in the 'project.MAP' file. This can be useful during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

- Run until flag 'h320.fPar_Err' = H
- Display register 'h320.rDiag' hex
- Delete flag 'h320.fPar_Err'

8.2.2 Error during homing

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

Reference switch absent or incorrectly wired:



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

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

Example:

```
CFB      Home          ; Homing axis 2
       k 2             ; axis number
         0             ;  search direction
       r 1010          ;  min. speed
       r 1011          ;  max. speed
         50            ;  timeout
       k 1             ;  homing with encoder index

STH      fHomeErr_2    ; Query home error flag
                       ;  of axis 2
CFB    h Errorhandl    ; Call (user specific)
                       ;  FBs, in case fHomeErr_2 = H
```

Notes

9. Installation and commissioning

9.1 Introduction

The following description shows the procedure for commissioning a servodrive with the PCD2.H32x motion control module. To guarantee fault-free operation of the H32x module, during commissioning the steps described below should be executed in the same sequence.

Selection criteria for a servo-drive with the PCD2.H32x module

A motion control unit always comprises the following parts:

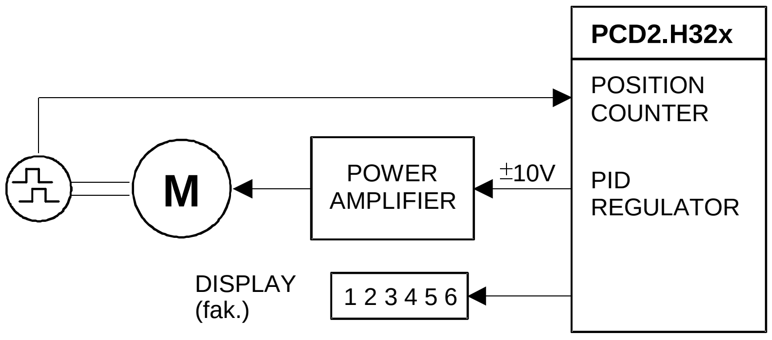
- A motion controller for setting the motion control parameters (position, velocity and acceleration) and for position control. This task is performed by the PCD1/2 with the H32x module.
- A servo-amplifier for triggering the servomotor.
- A servomotor for converting electrical into mechanical energy.
- A position transmitter, in the form of an incremental (or SSI) shaft encoder.
- A mechanical drive unit.

Selection of servo-amplifier and servomotor:

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

- Amplifier and motor must match each other (power, voltage and current).
- For precise position and speed control, a four-quadrant power output stage is required with an integral speed governor.
- The greatest possible governing range for amplifier and motor speed, so that the necessary torque can be applied even at low speeds.
- The H32x module supplies an analogue ±10V signal as the speed setpoint.
- For position capture, the H32x module needs an incremental shaft encoder that supplies at least two square-wave signals in phase quadrature.

Block diagram of a motion control drive with the H32x module



9.2. Installation and wiring

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

- Has the overall PCD1/2 system suffered any damage during transportation or assembly?
- Is the H32x module plugged into the space provided on the PCD1/2 bus and is the wiring complete on the relevant bus module?
- **System earth**
  For fault-free operation, a perfect earth to divert external noise voltage is indispensable. The PCD1/2 system should be connected to the "GND" terminal (24 VDC -) with the largest possible cross-section on the earth rail of the control box. It should further be ensured that all earth lines have been laid without loops.
- **Laying the cables**
  Regulations prescribe the laying of high-voltage cables and control cables in separate cable channels.
- **Motor controller output (± 10V analogue)**
  Check connections. The cable must be shielded.
- **Enable** for the driver is triggered through a "normal" digital output of the PCD1/2.
- **Encoder connections**
  For special attention with the 5V encoder:
  - Cable must be shielded and lines must be in twisted pairs.
  - Max. cable length 20m, min. conductor cross-section 0.25mm^2. In addition, check that encoder is correctly mounted (no slippage in coupling), check type and technical data (impulses per revolution).

9.3 Commissioning drive without motion control module

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

- Release connection of the H32x module’s controller output "Out" ($\underline{+}$ 10V) to the power stage at terminals 0 and "-" (minus) of the H32x module.
- The PCD1/2 and H32x module have been switched off. If this is not possible, execute point 9.4.1 first (switching on the supply voltages).
- The drive’s emergency stop limit switches should be set to prevent any damage arising from uncontrolled axis motion.

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

9.4 Drive with motion control module

9.4.1 Switching on the supply voltages

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

When the supply voltage of the PCD1 or PCD2 and H32x module is powered up for the first time (programming unit disconnected) the control LEDs should be noted.

On the PCD1:

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

On the PCD2:

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

On the motion control module PCD2.H320:

Meaning	**LED**	**Behaviour**
OK	Red	shows the correct starting up of the motion controller: must be **on**
PWR	Red	shows the presence of all necessary power supply: must be **on**
REF	Red	Reference switch: must be **off**
LS1	Red	Limit switch: must be **off**
LS2	Red	Limit switch: must be **off**

9.4.2 Preparing a basic user program

For commissioning a drive with the H32x module, a basic user program should be loaded into the PCD1/2 so that all checks and adjustments can be made. Refer to chapter ‘Programming’ for starting a project.

Evaluation of ’fPar_Err’ flag

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

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

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

9.4.3 Determining machine data

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

<u>Position tolerance</u> → see FB 'Init'

Recommended value 1 encoder revolution

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

<u>PID factors</u> → see FB 'Init'

Recommended start values:

Proportional factor	10
Integral factor	0
Differential factor	0
Integration limit	30000
Sample time D portion	15 (= 3 msec.)

<u>Machine factor</u> FB 'Init', PCD register (floatingpoint format)

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

Important:	The unit of measurement chosen here (m, cm, mm, 1/10mm, 1/100mm, µm) must also be used for the velocity , acceleration; jerk and all values relating to travel distance, breakpoint, velocity and acceleration throughout the user program for this axis.

Assume: $\text{mach.fact.} = \dfrac{4 \times In}{s}$ [impulses/unit of measurement]

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

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

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

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

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

<u>Velocity</u> FB 'Init'

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

Recommendation: 1/10 of maximum

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

Acceleration FB 'Init'

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

Recommendation: 1/10 of maximum

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

9.4.4 Direction, travel distance measurement (encoder)

To check direction of rotation and travel distance measurement, the following preconditions must be met:

- The connection between the H32x module's controller output (± 10V) and the power stage has been released at terminals 'OUT' and GND for the H32x module's axis.
- The PCD1/2 has been switched on (commissioning program loaded) and is in "Run".
- The programming unit is connected and the commissioning software has been started. All the necessary adjustments have been made. These settings must agree with definitions in the D2H320_B.MBA file.

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

Checking the direction:

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

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

If this is possible, the drive shaft can be turned by hand. Otherwise it must be moved by applying a set value (using a voltage source of ± 0.5 .. 1V) to the power stage.

Definition of positive and negative directions:

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

Checking path measurement:

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

A path value corresponding to the machine factor must be displayed as the actual position. If the display is incorrect, recheck the calculation and entry of machine factor .

Checking encoder signals A, B, IN:

By turning slowly on the free axis, you check the LEDs:

- LEDs A & B are turned on. Depending the speed you see them flash on and off.
- Once per turn you must see the LED IN flashing on.

Note, See the example timing chart on page 5 - 14 in chapter 5 of this manual.

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

- Faulty encoder
- Encoder signals A, B and IN are not in the order required at the H32x module's position decoder input.

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

9.4.5 PID controller

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

- With the controller powered off, the control circuit is closed by reconnecting the setpoint cable of the power stage to terminals 'OUT' and GND of the H32x module.
- Power up the controller:
  CAUTION: Activate emergency stop if the axis should proceed out of control. (There is –10V present on the analog output until the FB Init has been executed. The motor driver must not be enabled until the FB Init has been executed).
- The programming unit is connected and commissioning software has been started.

Checks:

1. Execute the FB Init and enable the driver.
   → The H32x module regulator is switched on and the axis is held in its position by the regulator.

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

Incorrect behaviour:

- The axis runs at maximum velocity.

Possible cause	• Position control circuit (H32x module) or speed control circuit (power stage) incorrectly poled. This means that, when the setpoint voltage is positive (= positive destination position) motion in the axis is in a negative direction. • The H32x module's analogue output is faulty and supplies a constant, maximum voltage of approx. +12V or –12V to the controller output.
Remedy	• Check direction, execute section 9.4.4 once again and make the necessary changes to poling. • Change the H32x module.

- The axis starts running briefly, and stops again.

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

- The axis runs jerkily to an incorrect destination position.

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

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

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

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

Setting the proportional factor (kp):

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

Setting the integral factor (ki):

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

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

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

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

Action:	- Reduce the above parameters.
	- Limit the integration sum

Setting the differential factor (kd):

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

Set-up rule for sample in the D portion:
For operation at low velocities, opt for a rather large sample time. Experience shows that sample time values generally lie between 2 ms and 9 ms.

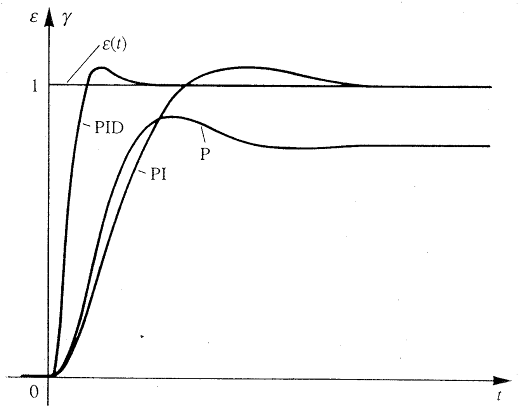
Optimization of settings:
If regulation does not proceed satisfactorily after values have been set as above, a further attempt must be made to vary individual parameters, so that a satisfactory result is achieved.

9.4.6 Effect of individual factors on controller behaviour

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

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

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



9.4.7 Simple commissioning program

Refer to chapters ‘Programming’ and ‘Commissioning’ for this purpose.

Notes

10. Commissioning Tool

This tool greatly simplifies the commissioning process of your new module. In this section we are going to explain the following:

- Installation of the commissioning tool.
- Commissioning tool components.
- Offline configurations.
- Using the Commissioning Tool with PG5
- Online configurations.

Note: If you have PG5 then you need to install the commissioning tool in the PG5 directory. If you have PG3 or PG4, then you should install the commissioning tool as a standalone program.

Before starting with the commissioning tool open the SAIA Project manager, open a project for the module program and create a .src file. For explanation purposes, give it a title of Home_FB Init. We will reference this later in the chapter when we explain the Homing and FB initialisation procedure.

10.1. Installation of the Commissioning Tool

Load the disk PCD8.H32.E and activate setup.exe. The installation steps are as follows:

1. Welcome Window. Click on Next.
2. License Agreement. Read, accept and click on Next.
3. Customer Information. Enter your name, company name and utilisation guidelines.
4. Customer Setup. Here you specify what system or systems you have. Click on the icon left of the feature to activate a pull down menu. There are four options to choose from. For this installation we are only concerned with the first one (this feature will be installed on the local hard drive) and the last one (this feature will not be available).
   - If you are only using a standard PCD with PG5, then select the first option for PCDComTool and the last option for xx7 and PCDSComm. The icon beside these two features should change to an X indicating that these features will not be installed.
   - If you have only xx7 PCDs then install only the xx7 ComTool.
   - If you have both standard and xx7 PCDs then install both features.
   - PCDSComm is only to be installed if you are using a standard PCD and PG4.
   - The lower section of the window shows the address location where the features are going to be installed. To change the suggested address, click on the buttons Change PCD and Change XX7. (The only restriction is if using PG5 then you have to load the tool into the PG5 directory).

Click on Next to continue the installation.

5. Ready to Install Program. Click on Install
6. Installing Commissioning Tool. Once the installation has started, another window will appear and ask "Are you sure you want to install Windows Script Version 5.5" Click on yes. Then click on yes again for the license agreement. Both installations will continue. the Windows Script Version installation will finish before the ComTool installation. Click on Ok for the Windows Script Version setup complete.
7. Install Wizard Completed. Click on Finish.

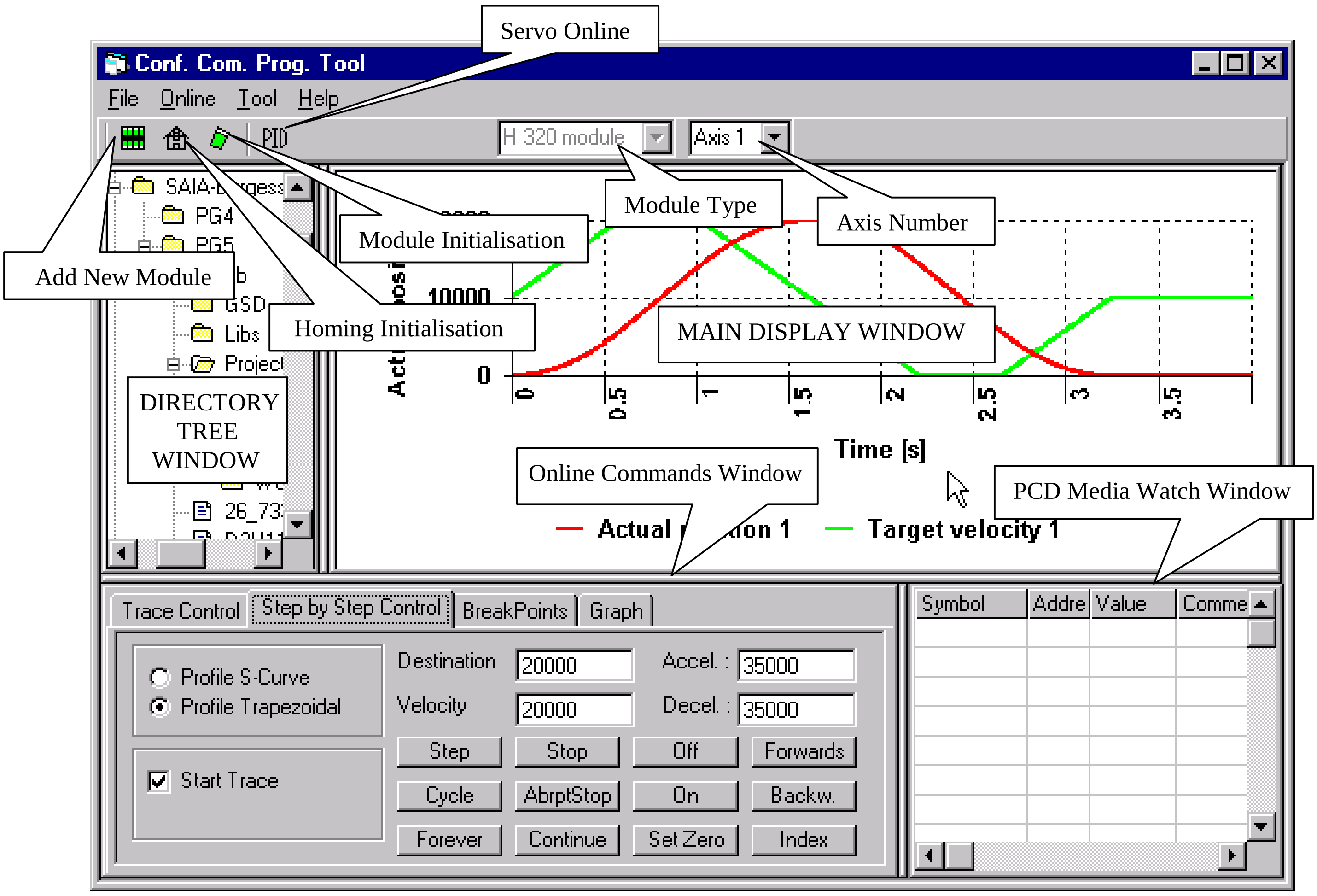
Note: if you are installing the program into PG5, the files are placed as follows:

1) D2H320CT.src is loaded into the FB file folder.
2) PCDComToolExample is a program that can be used with the commissioning tool to fine tune your servo. It is loaded in the Project file folder and is ready to be built and downloaded. (Make sure to change the FB intialisation parameters and also the Homing-paramters to match your application).
3) All other files are loaded into the main directory.

Follow these steps to open the tool:
<Start><Programs><Saia Commissioning Tool><PCD ComTool>

10.2. Tool Components

Main Window:



Profile S-Curve
Profile Trapezoidal [✓]
Start Trace [✓]

Destination	20000	Accel. :	35000
Velocity	20000	Decel. :	35000

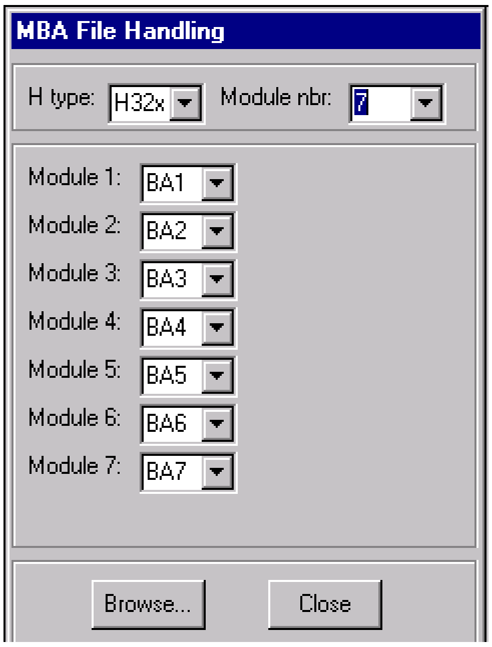
Step, Stop, Off, Forwards, Cycle, AbrptStop, On, Backw., Forever, Continue, Set Zero, Index

10.3. Offline Configurations

Add a New Module:

Clicking on this icon in the toolbar will display this dialog box. This function is for creating the .mba file.

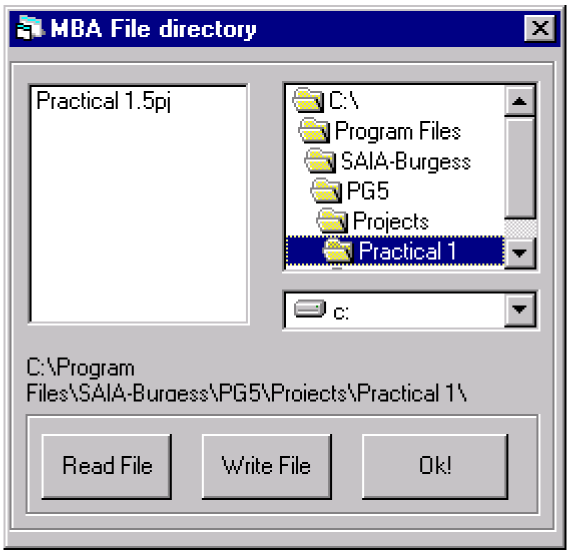
1. Enter the module type.
2. Enter the number of modules.
3. Enter the base addresses for the corresponding modules.
4. If you have different type modules, you have to create an .mba file for each type.
5. Click on the browse button to locate your project file.


MBA File Handling

H type: H32x Module nbr: 7

Module 1:	BA1
Module 2:	BA2
Module 3:	BA3
Module 4:	BA4
Module 5:	BA5
Module 6:	BA6
Module 7:	BA7

6. Clicking on the “Browse” button displays this dialog box.
7. Select the project file for your module Program
8. Click on “Write File”.
   This will add your .mba file to the project.

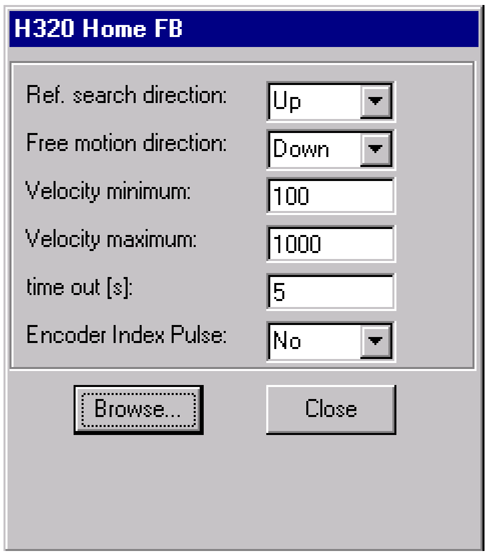


C:\Program Files\SAIA-Burgess\PG5\Projects\Practical 1\

Note: If you want to use the settings from an existing file, you can select the .mba file and click on “Read File” to display the settings. Click on “Write File” to add it to your project.

Homing Initialisation:

Clicking on this icon in the toolbar will display this dialog box. This function is for setting the parameters of the Homing segment of your program. This feature is for use with IL programming only. If programming in Graftec, the files must be created and modified manually.



Ref. search direction:	Up
Free motion direction:	Down
Velocity minimum:	100
Velocity maximum:	1000
time out [s]:	5
Encoder Index Pulse:	No

Reference search direction:
This is the direction the motor goes first when looking for the reference switch. If it reaches a limit switch before finding the reference switch it will reverse and go in the opposite direction.

Free motion direction:
Once the reference switch has been made, the signal goes high. The motor then goes in the Free motion direction until the reference switch signal goes low. At this point the Home position is set. (the home position value must be defined by the user)

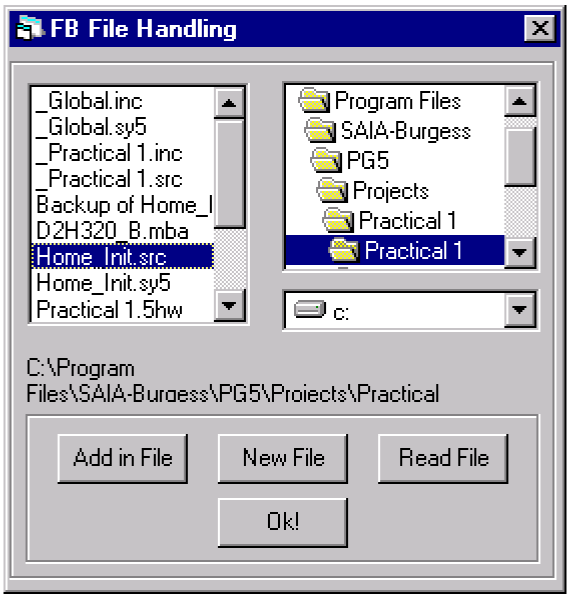
Velocity minimum:
This is the speed that the motor goes once the reference switch signal goes high. (Free motion direction mode.)

Velocity maximum:
This is the speed that the motor goes when it is looking for the reference switch.

Time out [s]:
This is the amount of time allowed for the reference to be activated. If it is not found in this amount of time the fHomeErr_x flag will be set and the program will be stopped.

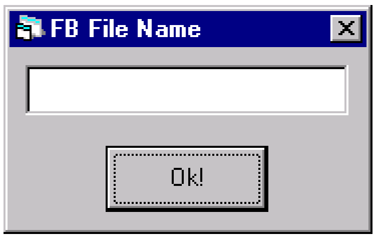
Encoder Index Pulse:
If you select yes, instead of the motor stopping when the reference switch goes from high to low, the motor continues to rotate until it finds the Encoder Index Pulse marker. When this signal goes from low to high, the Home position is set. (Home position is set but the value must be defined by the user)

Clicking on the “Browse” button of the Home FB dialog box will display this dialog box. Locate the Home_FB Init.src file for the Home and FB Initialisation program. (the one we had you create in the beginning of the chapter) Double click on it and then click on Add in File.



Add in File:
This function takes the parameters you have entered in the Home FB dialog box, creates the Homing Initialisation program, then adds it to the file you have selected.

New File:
If you have not created a file prior to this step, you can click on “New File” to create one. This will display the dialog box below.

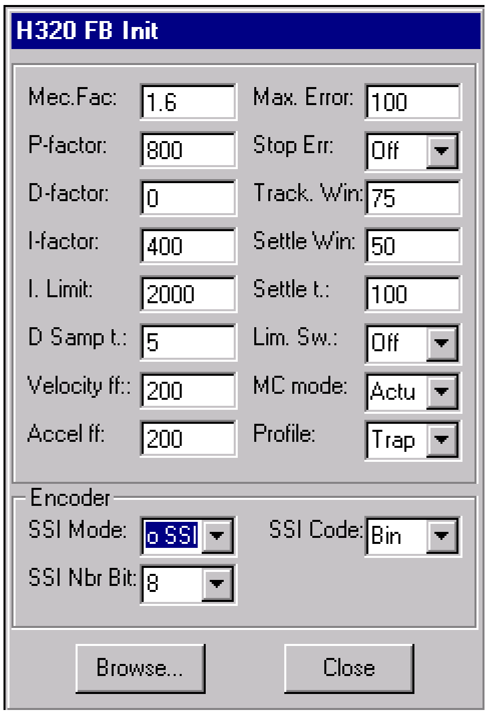


Clicking on Ok will create a program from the parameters you have entered in the Home FB dialog box and place it in your new file. To add the new file to the Program Files Folder(PG5), highlight the Program Files Folder and right click your mouse. From the popup menu select “Add Files”. Locate your new file via the browse window and then click on Add.

Read File:
If you would like to use an existing Homing initialisation program, you can locate the file and click on “Read File”. This will display the parameters of the selected file. Use “Add in File” or “New File” to add it to your project.

FB Initialisation:

Clicking on this icon in the toolbar will display this dialog box. This function is for setting the parameters for the FB initialisation segment of your program. Here we just give an explanation of what the boxes are. Their actual functionality is explained in Chapters 5 & 9.
This feature is for use with IL programming only.
If programming in Graftec, the files must be created and modified manually.


H320 FB Init

Mec.Fac:	1.6	Max. Error:	100
P-factor:	800	Stop Err:	Off
D-factor:	0	Track. Win:	75
I-factor:	400	Settle Win:	50
I. Limit:	2000	Settle t.:	100
D Samp t.:	5	Lim. Sw.:	Off
Velocity ff::	200	MC mode:	Actu
Accel ff:	200	Profile:	Trap

Encoder

SSI Mode:	o SSI	SSI Code:	Bin
SSI Nbr Bit:	8		

Mec Fac:
Mechanical factor (the number entered here determines the units of the parameters. (µm, mm, inch, etc.) Enter in floating point format.

P - Factor:
Proportional Factor. (0 – 32767) Enter in Integer format.

I - Factor:
Integral Factor. (0 – 32767) Enter in Integer format.

D - Factor:
Derivative Factor. (0 – 32767) Enter in Integer format.

I - Limit:
Integration Limit (0 – 32767) Enter in Integer format.

D Samp t:
Derivative Sampling Time. (0 – 32767) Enter in Integer format.

Velocity ff:
Velocity Feedforward Factor. (0 – 32767) Enter in Integer format.

Acel ff:
Acceleration Feedforward Factor. (0 – 32767) Enter in Integer format.

Max Error:
Position Error Limit (This limit is based more on the mechanical limitations of your system.)

Stop Err:
Automatically stop when a position error occurs.

Track Win:
Tracking Window Range (0 to Maximum which is 32767 divided by the Mechanical Factor)

Settle Win:
Settle Window Range (0 to Maximum which is 32767 divided by the Mechanical Factor)

Settle t:
Settle Time (0 to Maximum which is 32767 divided by the Mechanical Factor)

Lim Sw:
Limit switches active

MC mode:
Motion complete mode (Actual or Target)

Profile:
Profile selection. (S-curve or Trapezoidal)

Encoder:
- SSI Mode: if you have an SSI encoder select the frequency here. If not, select No SSI. (If you select No SSI, the other two boxes in this section are set automatically.
- SSI Code: select Binary or Gray.
- SSI Nbr Bit: 8 - 26.

When you have entered all the parameters, click on the browse button. This will display the FB File Handling dialog box. Follow the same procedure as explained in the Homing initialisation procedure to add it to the Home_FB Init.src file. Now both your Homing initialisation and FB initialisation parameters are in the same file.
You must repeat this process for each module that is installed in the PCD. Each module has 2 axis, so if you are using both axis, you have to enter a set of parameters for each one. To select the axis you want to enter parameters for, locate the “Axis number” combo box which is located above the Main Display Window and make your selection. The axis numbers correspond to the module numbers as follows:

Module 1	Axis 1	Axis 2
Module 2	Axis 3	Axis 4
Module 3	Axis 5	Axis 6
Module 4	Axis 7	Axis 8
Module 5	Axis 9	Axis 10
Module 6	Axis 11	Axis 12
Module 7	Axis 13	Axis 14

The module numbers are determined by the Base Address assigned to them in the .mba file

10.4. Using the Commissioning Tool with PG5

To use the commissioning tool with PG5, you need to add the file “D2H320CT.src” to the Program Files Folder of your project. This file is located in the FB File Folder in the PG5 directory. The easiest way to do this (assuming you have PG5 open) is:

1. From PG5, highlight the Program Files Folder in the Project Tree.
2. Right click the mouse.
3. Select “Add Files” from the pop up menu.
4. Navigate to the file via the browse window.
5. Select the option ‘store path to file, do not copy it.’
6. Click on Add.

Note: when the file is added to the Program Files Folder, the file will be linked to the project by default. Make sure to unlink it when finished commissioning your motor(s). The file is only for commissioning purposes, and if linked to your user program, it could possibly modify the behaviour of your program.

For tuning your motor we recommend that you create a .src file specificially for the task of tuning your motor. All you need in this file is the FB Initialisation parameters, and a COB with the Instructions:

```
COB     0
        0

extn    comtool
cfb     comtool

ECOB
```

The program "PCDComToolExample" was included in the installation package and it contains 2 sets of FB initialisation parameters and a COB with the instructions listed above. You can setup parameters for 1 or both axes of your H320 module. The commissioning tool will correspond with the axis displayed in the “Axis number” combo box above the Main Display Window. Before building and downloading this program, you need to change the parameters to match your setup, and update the base address in the .mba file to the location of your H320 module.
If for some reason you do not have this program, you can use the FB Init dialog box to create the file, and then just add $Include D2H320_B.Equ at the beginning of the file and the COB segment at the end of it. Don’t forget to move $EndGroup to the end of your COB. Make sure only this file and the D2H320CT.src files are linked in the Project Tree, then Build and Download them into the PCD.
If you want to use the commissioning tool with one of your existing programs, it is important to take into consideration when capturing a trace the program cycle time is greatly increased. If your program has some tight cycle time tolerances the behaviour of your program could be altered. Also you must make sure that the values entered in the step by step window for destination, velocity, acceleration, and deceleration match those that are in your program.

10.5. Online Configurations

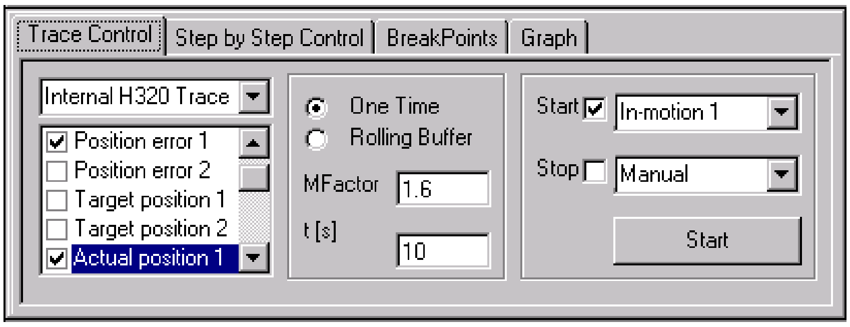
The Online Command window is the central base for performing online operations. There are 4 different windows available, selectable by the tabs at the top of the window.

Trace Control:

Trace Control is where you specify the data you want to view.There are 2 modes available in the Trace Control window.

- Internal H320 Trace
- Trace in PCD Media

The Internal H320 Trace mode captures and displays the performance curves of specific data related to the H320 characteristics. In the display below, we have chosen to trace the Position error and Actual position curves for Axis 1. (1 = axis 1 and 2 = axis 2). You can display up to 4 different curves at a time.


Trace Control | Step by Step Control | BreakPoints | Graph

Internal H320 Trace

- [✓] Position error 1
- [ ] Position error 2
- [ ] Target position 1
- [ ] Target position 2
- [✓] Actual position 1

(●) One Time
( ) Rolling Buffer

MFactor 1.6

t [s] 10

Start [✓] In-motion 1

Stop [ ] Manual

Start

The Trace in PCD Media mode enables you to trace the activity of up to four elements (Inputs, Outputs, Registers and Flags). This is very useful to view the timing of these elements in reference to the performance curve of the servomotor.

One Time:
Acquires data until the acquisition buffer is full. Once it is full, no more data is acquired.

Rolling Buffer:
Acquires data until a stop event or the ESC button is pressed. If the acquisition buffer is full and a stop event or ESC has not been activated, the data in the buffer is overwritten. Although this function can be used in Internal H320 Trace mode, its functionality is more useful for the Trace in PCD Media mode.

Mfactor:
Enter the Mechanical Factor for your encoder.

t [s]:
This is the duration of the acquisition. (For H320 Internal Trace mode only).

X Nbr. Points:
This is the number of points displayed on the X axis. Maximum 512. (For Trace in PCD Media mode only).

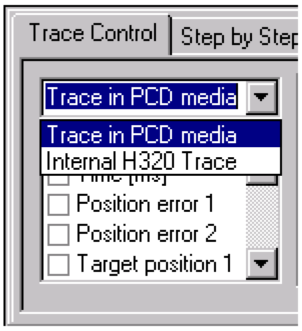
Start: (combo box)
- In Internal H320 Trace mode, this combo box lists several functions from which you can choose one that you want to activate the beginning of the trace acquisition. If you place a checkmark in the checkbox, then the trace will begin when that functions bit is high. If you do not place a checkmark in the checkbox, then the trace will begin when the bit is low. For example, if we select In-motion 1 and we have a checkmark in the checkbox, when the motor starts to move, and the In-motion 1 bit is high, the capturing of the trace begins.
- In Trace in PCD Media mode, there is only two selections available. Manual or PCD Media. PCD Media will open a dialog box where you can designate an Input, Output or Flag, that when it goes high, the trace will begin.

Stop: (combo box)
- In Internal H320 Trace mode, this function works identically to the Start function except that you are designating the function you want to stop the trace acquisition. Note, if the acquisition time expires or the acquisition buffer is full (512 points) before the specified element is activated the trace will still stop.
- In Trace in PCD mode, this also works identically to the Start function except that you are designating and Input, Output or Flag to determine when you want to stop the trace acquisition.

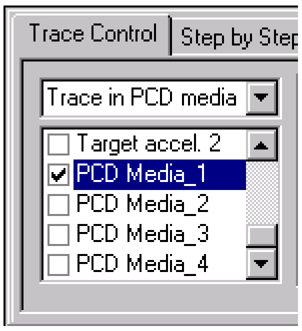
Start: (button)
Use this button when you are in the Trace in PCD Media mode. If you have selected manual in the Start combo box, clicking on this will start the acquiring of data and displaying the trace you have chosen to view. Assuming you have Rolling Buffer selected, this will continue until a stop command is activated whether it is an element you have designated or manually activating the stop button.
If you have selected PCD Media in the Start combo box, then clicking on this button will activate the data acquisition mode but the data is not read until the element that you have designated in the Start combo box goes high. Note, when you click on this button to start the acquisition mode, the button then becomes the stop button.

Designating Elements:

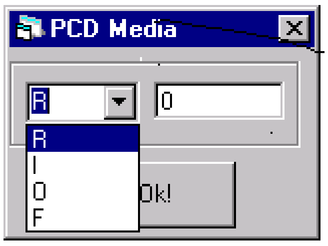
1. Select Trace in PCD media in the combo box.



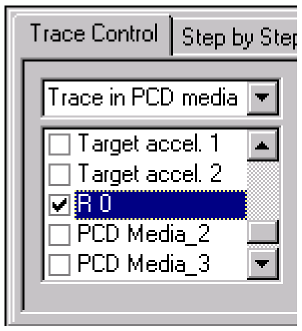
2. Scroll to the bottom of the selections list and select PCD Media_1 to PCD Media_4.

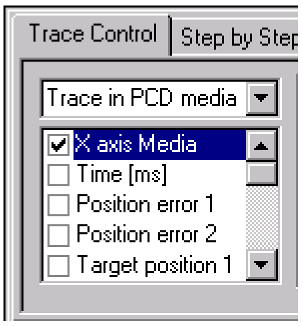


3. In the PCD Media dialog box, select the element type and address.



4. Now in the selections list, the element selected is displayed and when you have a checkmark in the checkbox, a trace will be displayed for the specified element.




Trace Control | Step by Step

Trace in PCD media

- [✓] X axis Media
- [ ] Time [ms]
- [ ] Position error 1
- [ ] Position error 2
- [ ] Target position 1

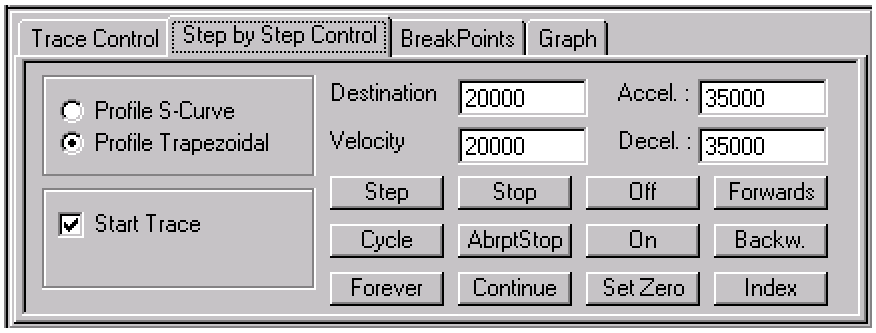
X axis Media is also an available selection. This is not to be confused with PCD Media_1 to PCD Media_4. X axis Media is for displaying a real time clock scale on the x axis of your graph. You will have to add a SysRd instruction to your program:

```
SYSRD   K 7000     ; Read system counter
        R 0 - 4095 ; Result of the read
```

The register you designate for the result, is the one you will enter for the X axis Media register. Reference the Instruction set guide for more details on this instruction.

Step by Step Control:

The Step by Step Control window is where you make very simple moves with the servo and then you can see the performance curve from that movement. The curves you view will be the ones that you have designated in the Trace Control Window.


Trace Control | Step by Step Control | BreakPoints | Graph

- Profile S-Curve
- [✓] Profile Trapezoidal
- [✓] Start Trace

Destination: 20000
Velocity: 20000
Accel. : 35000
Decel. : 35000

Profile S-Curve/Profile Trapezoidal:
Determines the generated trajectory. The characteristics of the two are explained in chapter 5.

Start Trace:
Place a checkmark in this box if you want to generate a trace when you activate a movement. The data acquisition is a little time consuming, so if you just want to test a movement and you are not interested in the performance curves, then you would make sure the checkbox is empty.
Start Trace is only available for Internal Trace mode.

Destination:
The destination of the movement.

Velocity:
This is the maximum speed that can be obtained during the cycle or step movement.

Accel:
The acceleration of the cycle or step movement.

Decel:
The deceleration of the cycle or step movement. (This parameter is only available for Profile Trapezoidal).

Jerk:
The maximum amount of change in the acceleration per cycle. (This parameter is only available for Profile S-Curve).

Note: The units of the values entered for Destination, Velocity, Accel, and Decel are based on the Mechanical Factor. The unit for the “Jerk” value is determined by the formula shown in Appendix A page A-37 of this manual under the SetJerk command.

Step:
Activates the first half of a cycle. (You have to “Set Zero” before each step is activated)

Cycle:
Activates one complete cycle. (You have to “Set Zero” before the initial cycle only. At the end of the cycle, the motor has returned to the starting point which is zero).

Forever:
Continually cycles until the Stop button is activated.

Stop:
Stops the movement using the deceleration parameter to decelerate to a complete stop.

Abrupt Stop:
Immediately stops the movement without a deceleration period.

Continue:
If a movement has been stopped before the destination has been reached, clicking on this button will continue the movement to the destination. Note, if your movement is a cycle, this does not complete the cycle, it will only complete the first step of the cycle until it reaches the destination.

Off:
Releases the servo loop so you can manually rotate the motor by hand.

On:
Engages the servo loop. Note, if you have released the servo loop and manually rotated the motor, then your zero point should be reset.

Set Zero:
Sets the zero point or starting point for the movements. (e.g. before executing a step, or if you have manually rotated the motor off the zero point).

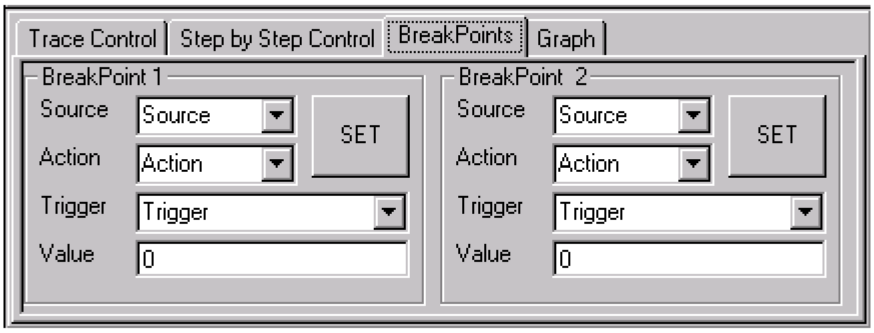
Forwards:
Activates a forward movement in velocity mode and continues running until the Stop button is activated.

Backwards:
Activates a backwards movement in velocity mode and continues running until the Stop button is activated.

Index:
Rotates until it finds the index pulse marker. Note, your velocity setting needs to be set to a low value, otherwise the index pulse marker will not been seen.

BreakPoints:

Breakpoints are used to program an event based on a specific condition.
Two separate breakpoints can be programmed, Breakpoint 1 and Breakpoint 2. These break-points can be programmed in your program, or be set as starting and stopping points via the Start and Stop combo boxes in the Trace Control Window.



Source:
Designates the axis that the trigger event is going to take place on.

Action:
Specifies what action is going to be taken when the trigger event occurs. This action is going to be taken on the axis that is displayed in the combo box above the Main Display window.

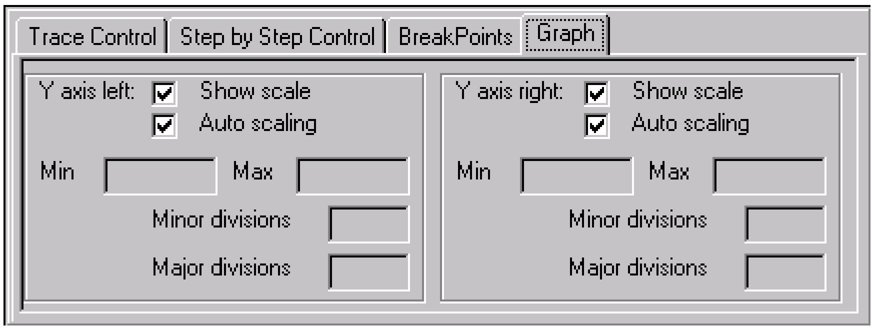
Trigger:
Specifies the event that is going to trigger the action taken on the breakpoint axis.

Value:
Specifies the value for the trigger point of the selected event.

SET:
Sets the breakpoint(s). Note, before tracing a performance curve, you must click on this button to activate the breakpoints. Even if you are doing the same trace over and over, you have to click on the “SET” button before each trace.

Graph:

The Graph window is for setting up the parameters of the graph for displaying the traces.
You can display up to four different traces. Selections 1 and 3 are referenced to the Y axis scale, and selections 2 and 4 are referenced to the Y axis 2 scale. Selection numbers are based on the order that they appear in the list.



Y axis left: ✓ Show scale ✓ Auto scaling
Min Max
Minor divisions
Major divisions

Y axis right: ✓ Show scale ✓ Auto scaling
Min Max
Minor divisions
Major divisions

Show scale:
Display or Hide the scale. (only the scale, the graph will always be displayed)

Auto scaling:
Always displays a scale adapted to the data acquired.

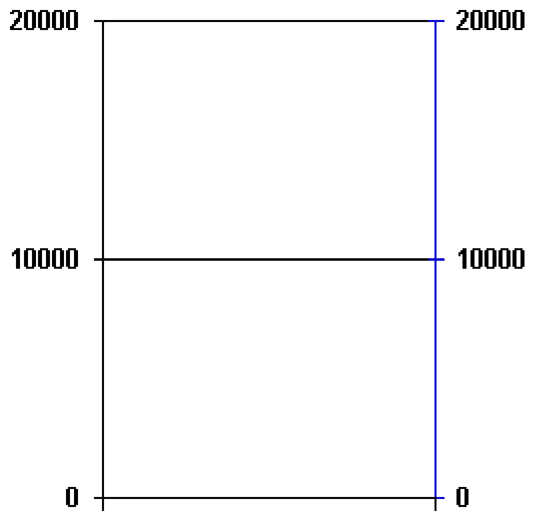
If you remove the checkmark from the checkbox, then the Min, Max, Minor divisions, and Major divisions become active. With these options, you can customize the scale for your graph.

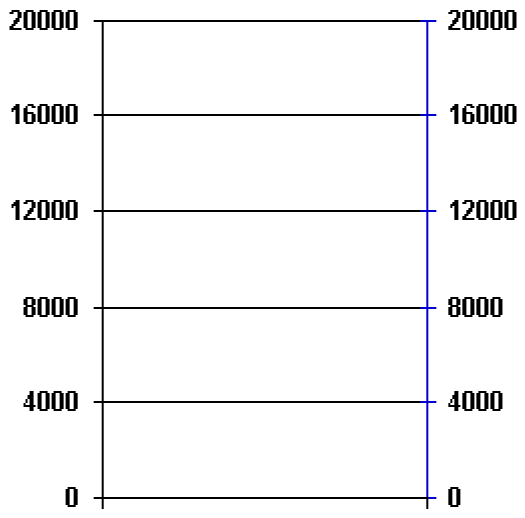
Min:
Beginning of the scale.

Max:
Top end of the scale.

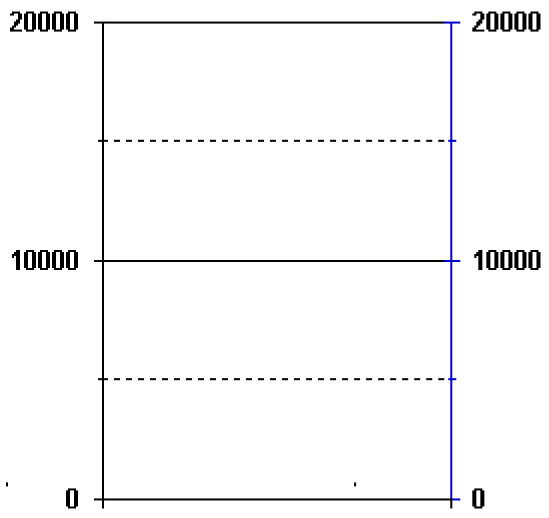
Minor divisions/Major divisions:

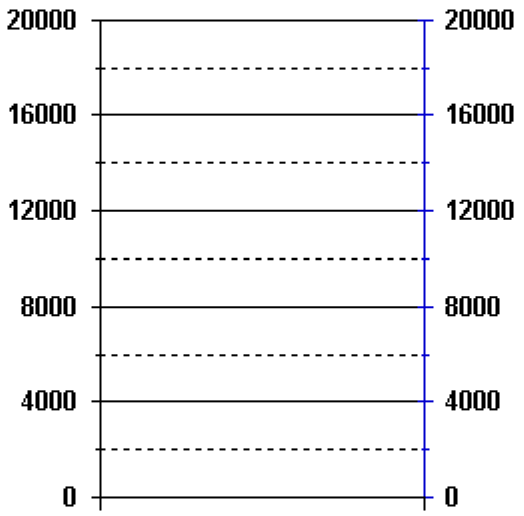
Specifies the divisions of your graph. For example, if your scale goes from 0 to 20000 and you enter a 2 in the Major division's box, then your graph would show the points 0, 10000, 20000. If you enter a 5 in the Major division's box, then your graph would show the points 0, 4000, 8000, 12000, 16000. See the displays below:





The value entered in the Minor divisions box is the number of divisions displayed between the Major divisions. A value of 2 was entered for the Minor division in the display below:

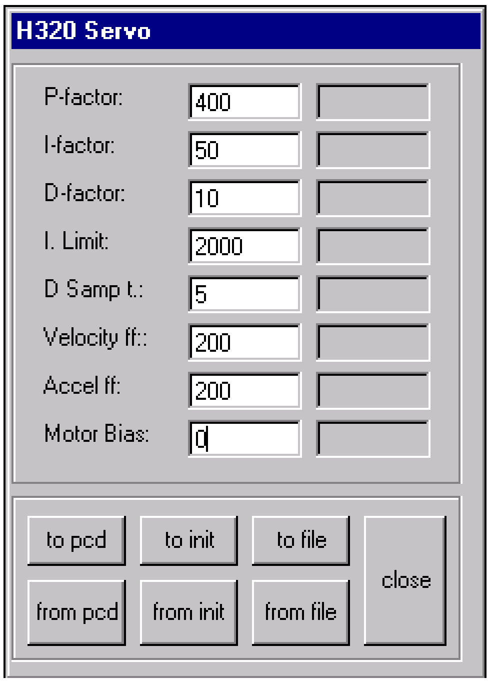




Tip: To store a captured trace to a Microsoft Word document:

1. Capture a trace.
2. If not opened, open the Servo Online Dialog Box. Make sure the parameters of the displayed trace are written in the appropriate boxes.
3. Press Alt + Print Screen.
4. Open a new Word document.
5. Under the Edit menu click on Paste.
6. The Commissioning Tool window is pasted to the document.
7. Add notes, eg. time, date, project, etc.
8. Save document.

Servo Online: PID



Clicking on this icon in the toolbar will display this dialog box. The purpose of this dialog box is to simplify the process for fine tuning the servo motors. The parameters are explained in chapter 5 and chapter 9. The function of the buttons is as follows:

to pcd:
Loads the parameters displayed into the PCD.

from pcd:
Gets and displays the current parameters loaded in the PCD.

to init:
Replaces the existing parameters in the selected FB initialisation file with the displayed parameters. Clicking on this button will open the FB Handling dialog box for you to locate the FB initialisation file. Once you locate the file double click on it and then click on “Add in File”.

from init:
Takes the parameters from a FB initialisation file and displays them. Clicking on this button will open the FB Handling dialog box for you to locate the FB initialisation file. Once you locate the file double click on it and then click on “Read File”.

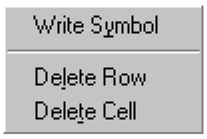
to file:
Sends the parameters listed to a .txt file. This is for making a backup file of your parameters. Clicking on this button will open the FB Handling dialog box for you to locate the .txt file. If one already exists, double click on it and then click on “Add in File” and this will add a parameter list to the file. Note, it does not replace any existing lists. If one does not exist, simply click on “New File” to create one.

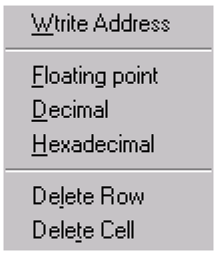
from file:
Takes the parameters listed in the .txt file and displays them. Note, if there is more than one list in the file, the last list entered is displayed. Clicking on this button will open the FB Handling dialog box for you to locate the FB initialisation file. Once you locate the file double click on it and then click on “Read File”.

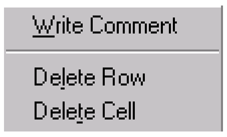
close:
Closes the Servo online dialog box.

PCD Media Watch Window:

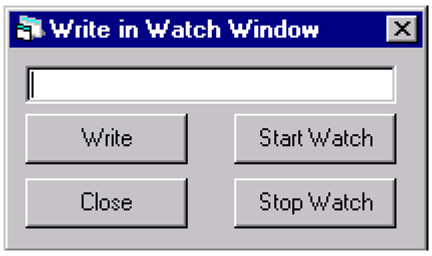
The PCD Watch Window is for monitoring the status or value of specified elements. (Registers, Inputs, Outputs, or Flags). Similar to the Watch Window in PG5.
Select the cell you wish to make an entry in. Once the cell is highlighted, right click your mouse and this will bring up a pop up menu for the highlighted cell. The following pop up menus are available:







Selecting “Write Symbol”, “Write Address”, or “Write Comment”, opens the “Write in Watch Window” dialog box.



Enter the data corresponding to the highlighted cell and click on Write. This will make the entry into the Watch Window.
When entering the address, if the value is just a binary bit, then you don’t have to make anymore selections. If the value is floating point, decimal, or hex, it is necessary to specify the format from the pop up menu.
Once your entries are made, click on “Start Watch” to activate the window. Click on “Stop Watch” to deactivate the window.
“Close” closes the “Write in Watch Window” dialog box. It does not deactivate the watch window.

Notes:

11. Security aspects

If a drive unit is triggered with a PCD2.H32x module, particular attention should be paid to the following points:

Drive power-up phase

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

During this power-up phase and until the FB Init is executed, the analog controller output of the H32x module can have value anywhere between -10V and +l0V. **For this reason it is absolutely necessary that the power section of the drive is either switched on or released by the user program through a digital output <u>after the FB Init has been executed.</u>** This guarantees that the drive does not proceed out of control during this power-up phase.

Monitoring the position error
(See flag 'fPosErr_n')

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

The following causes can result in exceeding the position error:

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

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

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

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

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

Limit Switches

These can be used in different ways:

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

Security limit switches

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

Watchdog

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

X0Bs for PCD hardware errors

Program XOBs 0 to 13 and, if necessary, disconnect the drive directly via a digital output.

Notes

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

Function block 'Init'

Init

FB: Initialization of an H32x module

		Init Function Block
Axis number →	= 1	
Machine factor →	= 2	
Proportional factor →	= 3	
Integral factor →	= 4	
Derivative factor →	= 5	
Integration limit →	= 6	
Derivative sampling time →	= 7	
Velocity feedforward factor →	= 8	
Acceleration feedforward factor →	= 9	
Position error limit →	= 10	
Auto stop on position error →	= 11	
Tracking window →	= 12	
Settle window →	= 13	
Axis-settled time →	= 14	
Limit switch mode →	= 15	
Motion complete mode →	= 16	
Profile mode →	= 17	
Encoder mode →	= 18	
SSI: nbr of bits →	= 19	
SSI: code →	= 20	
	FB Levels: 1	
	Index Modified: No	
	Processing Time: 16 ms	

Function description:

This FB defines the settings of a PCD2.H32x module and reads the module base address from the D2H320_B.MBA file.

Parameter 1 need to be specified as a constant 'K', parameter 11 as PCD register addresses (absolute or symbolic) and all other Parameters as integer.
Parameter 2 is the machine factor. This contains the encoder parameters and the mechanical parameters of the system. (Must be entered in floating-point format, e.g. for '4' as 4.0 or 4E1) and all other parameters should be specified as integer values.
The FB Init has to be executed first in the user program since the addressing for the FB Exec and Home is calculated.

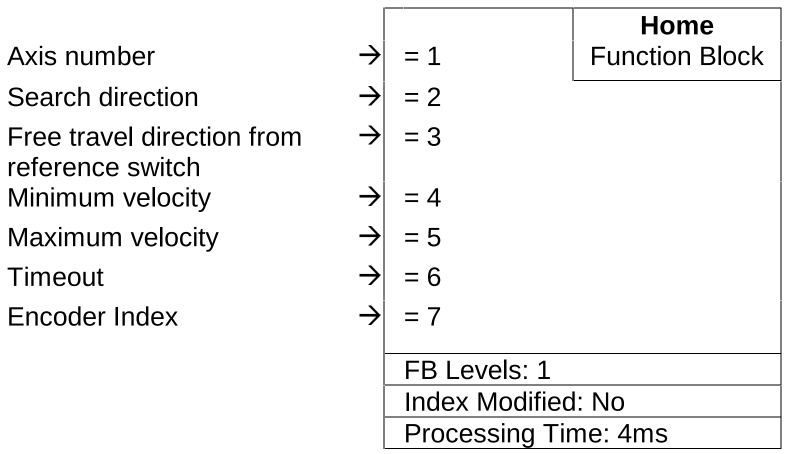
Par.	**Description**	**Type**	**Format**	**Value**	**Comment**
= 1	Axis number	K	K n	1 – 14	Parameter as constant or as an integer in a register
= 2	Machine factor	R	Floating	0 - 9.223371 E^{18}	Pulse/Unit Factor containing parameters for the encoder and the system.
= 3	Proportional factor		Integer	0 – 32767	P factor of PID controller.
= 4	Integral factor		Integer	0 – 32767	I factor of PID controller.
= 5	Differential factor		Integer	0 – 32767	D factor of PID controller.
= 6	Integration limit		Integer	0 – 65535	Limit on influence of I portion in system.
= 7	Derivative sampling time		Integer	0 – 32767	Sample interval for D factor
= 8	Velocity feedforward factor		Integer	0 – 32767	
= 9	Acceleration feed-forward factor		Integer	0 – 32767	
= 10	Position error limit	R	Integer		(User Unit) 2)
= 11	Auto stop on position error				0: Off 1: On
= 12	Tracking window 1)		Integer	0 – 32767	(User Unit) 2)
= 13	Settle-window 1)		Integer	0 – 32767	(User Unit) 2)
= 14	Axis-setteled time		Integer	0 – 6717	milliseconds
= 15	Limit switch mode		Integer		0: Off 1: On
= 16	Motion complete mode		Integer		0: commanded 1: actual
= 17	Profile mode		Integer		0: trapziodal 1: velocity contour 2: S-Curve 3: electronic gearing
= 18	Encoder Mode 3)		Integer		0: incremental encoder 1: 250kHz SSI 2: 500kHz SSI 3: 1MHz SSI
= 19	SSI: Nbr of bits 3)		Integer	8 – 26	
= 20	SSI: Code 3)		Integer		0: binary coded 1: gray coded

1) the window is in fact +/- the range value
2) User Units depending the Machine factor (um, mm, inch, etc…).
Range: 32767 devided by Machine factor
3) Each axis can be initialized independent in incremental or SSI mode. But SSI frequency, number of bits and code are common for both axis.

Function block 'Home'

Home

FB: Initialization of home position



Function description:

This FB defines the homing settings for the PCD2.H32x module. Parameter 1 needs to be entered as a constant 'K' and all other parameters must be entered as integers.

- The **'fEndHome_n' flag should be reset before any homing,** otherwise homing cannot start. This flag is automatically set high when homing is complete.
- The **'fHomeErr_n'** is set if the reference could not be found within the specified timeout.
- The acceleration has to be defined before executing the FB Home.

(n indicates the axis number).

Par.	Description	Type	Format	Range	Comment
= 1	Axis number	K	K n	1 – 14	
= 2	Search direction		Integer	0 – 1	Defines the direction in which to find the reference switch: 0 = backward 1 = forward
= 3	Free travel direction from reference switch		Integer	0 – 1	Defines the free travel direction, away from the reference switch: 0 = backward 1 = forward
= 4	Minimum velocity	R	Integer	--	Velocity for leaving the reference switch.
= 5	Maximum velocity	R	Integer	--	Velocity for seeking the reference switch.

Par.	Description	Type	Format	Range	Comment
= 6	Timeout		Integer	0 – 65535 [s]	Time until homing will be halted. 0: no timeout
= 7	Encoder Index			0, 1	0: No 1: Yes

Encoder Index No	homing reaches reference switch and stops
Encoder Index Yes	homing reaches reference switch and goes back until encoder index reached.

Note: after homing reaches the reference switch or reference switch + encoder index signal, the user must define this position. (see SetAct ualPostion page A-32)

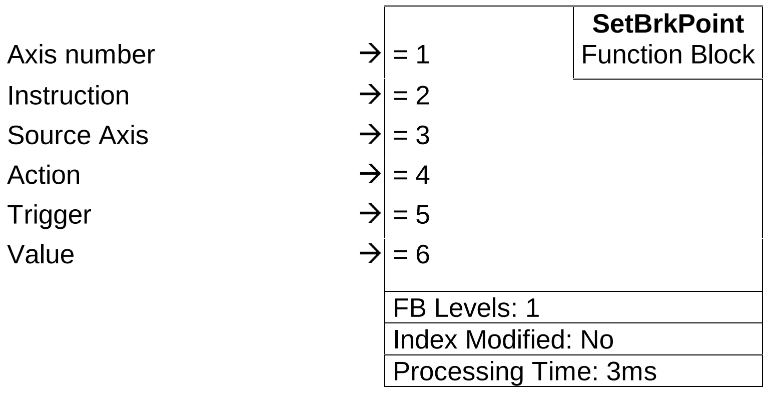
Warning:
The Homing procedure works in velocity mode, which means the velocity "Vmin" or "Vmax" (parameters 2 and 3 of the FB Home) are positive in the case the user chooses "UP" and negative in the other case. At the end of this FB the velocity configured remains "Vmin".

In closed-loop the velocity cannot be negative!

So before working in closed-loop, the velocity must be reinitialised in positive number.

Function block ’SetBrkpoint’

SetBrkPoint **FB:** Special FB for breakpoint instructions execution



Function description:

SetBreakpoint establishes a breakpoint for the specified *axis* to be triggered by a condition or event on *sourceAxis*, which may be the same as or different from *axis*.
Up to two concurrent breakpoints can be set for each axis.

Par.	Description	Type	Format	Range	Comment
= 1	Axis number	K	K n	1 – 14	
= 2	Instruction		Integer		1)
= 3	SourceAxis	R	Integer		1)
= 4	Action	R	Integer		1)
= 5	Trigger	R	Integer		1)
= 6	Value	R	Integer		[Units] 1)

The following constants are declared within file ‘D2H320_B.EQU’ to simplify the usage.

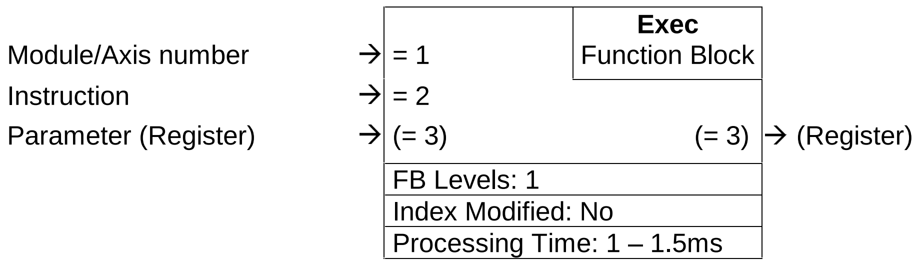
Parameter	***Symbol***	***Code***	***Description***
Instruction			
	SetBreakpoint_1	01D4h	Set breakpoint 1 type
	SetBreakpoint_2	11D4h	Set breakpoint 2 type
SourceAxis			
	SrceAxis_1	0FFF0h	Axis 1 (used directly with an AND to rebuild the data word)
	SrceAxis_2	0FFF1h	Axis 2
Action			
	ActnNone	0FF0Fh	None
	ActnUpdate	0FF1Fh	Update
	ActnAbruptStop	0FF2Fh	Abrupt Stop
	ActnSmoothStop	0FF3Fh	Smooth Stop
	ActnMotorOff	0FF4Fh	Motor Off
Trigger			
	TrgNone	000FFh	None
	TrgPositiveComPos	001FFh	Trigger: Positive commanded posi-tion
	TrgNegativeComPos	002FFh	Trigger: Negative commanded posi-tion
	TrgPositiveActPos	003FFh	Trigger: Positive Actual position
	TrgNegativeActPos	004FFh	Trigger: Negative Actual position
	TrgComPosCrossed	005FFh	Trigger: Commanded position crossed
	TrgActPosCrossed	006FFh	Trigger: Actual position crossed
	TrgTime	007FFh	Trigger: Time
	TrgEventStatus	008FFh	Trigger: Event Status *
	TrgActivityStatus	009FFh	Trigger: Activity Status *
	TrgSignal	00AFFh	Trigger: Signal *
Value			
	none	value	

Note: for these (*) trigger instructions the value is a mask corresponding to EventStatusRegister, ActivityStatusRegister, SignalStatusRegister. The mask required is a “double word mask.” This means there is a high word and a low word. The high word is used as a sense mask which determines the active state (high or low) of the bits. The low word determines the bit for triggering the breakpoint. If you want the trigger to occur when the designated bit goes high, then in the high word you would set this bit to a 1. Likewise if you want the trigger to occur when the bit goes low, you would set this bit to a 0. For example, if you want to use Axis In of the Signal status register (bit 6) the value parameter loaded would be 400040H for active high, and 000040H for active low.
See **Chap. 5 Function Description** for StatusRegister description.

Function block ’Exec’

Exec

FB: Execution of an instruction for the H32x module



Function description:

This FB is used to send execution instructions to the PCD2.H32x module.

The axis number (parameter 1) must be a constant (k 1…k 14). The base address is defined in the 'D2H320_B.MBA' file. The FBs support max. 7 PCD2.H32x modules per PCD system.

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

The parameters of an instruction (e.g. the acceleration value in the instruction SetAcceleration) are transferred in a register (parameter 3). If an instruction requires no parameters (e.g. Start) any register can be transferred, or 'rNotUsed', which is predefined in the D2H320_B.EQU file.

As the diagramm indicates, in parameter 3 can also be the result of an executed instruction (e.g. GetActualPosition).

The following list shows an overview over the instructions used by an user program, with an indication of parameter 3 usage.

Individual instructions for the PCD2.H320 (FB parameters)

Parameter 2		Parameter 3				Description	Page
Symbol	Code	none	in	out	buff		
Reset *	1A39h	X				Reset chipset	A-10
NoOperation *	1A00h	X				Perform no operation, used to verify communica-tions	A-11
ClearInterrupt *	1AACh	X				Reset Axis Event Interrupt	A-11
GetTime *	063Eh			X		Get the number of cycles occured since start	A-12
GetHostIOError *	02A5h			X		Get the most recent IO error code	A-12
MultiUpdate**	0C5Bh		X			Multiple axis immediate parameter update for buffered instructions	A-13
StartMotion**	0F00h		X			Starts Motion of specified axis and clears event-status	A-13
SetMotorBias	050Fh		X			Set motor output offset	A-14
SetDerivativeTime	059Ch		X			Set derivative sampling time	A-14
SetKaff	0593h		X		X	Set acceleration feedforward gain	A-15
SetKd	0527h		X		X	Set derivative gain	A-15
SetKi	0526h		X		X	Set integral gain	A-16
SetKp	0525h		X		X	Set proportional gain	A-16
SetKvff	052Bh		X		X	Set velocity feedforward gain	A-17
SetIntegrationLimit	0995h		X		X	Set integration limit	A-17
SetKout	059Eh		X			Set the output factor for the digital servo filter	A-18
SetAxisMode	0D87h		X			Set axis operation mode (enabled or disabled)	A-18
SetLimitSwitchMode	0D80h		X			Set limit switches (on or off)	A-19
SetMotionCompleteMode	0DEBh		X			Set the motion complete mode (target or actual)	A-19
SetMotorMode	0DDCh		X			Set motor mode (open/closed loop)	A-20
SetAutoStopMode	0DD2h		X			Set auto stop on position error (on or off)	A-20
SetInterruptMask	052Fh		X			Set interrupt mask	A-21
GetInterruptAxis	03E1h			X		Get the axis with pending interrupts	A-22
GetActivityStatus	03A6h			X		Get activity status	A-23
GetEventStatus	0331h			X		Get event status word	A-24
GetSignalStatus	03A4h			X		Get the current axis Signal Status Register	A-25
GetSignalSense	03A3h			X		Get the interpretation of the Signal Status Bits	A-26
SetSignalSense	05A2h		X			Set the interpretation of the Signal Status Bits	A-27
ResetEventStatus	0534h		X			Reset bits in event status word	A-28
SetMotorCommand	0577h		X		X	Set direct value to motor output register	A-29
GetMotorCommand	0369h			X		Read buffered motor output command	A-29
SetAcceleration	0990h		X		X	Set acceleration	A-30
GetAcceleration	074Ch			X		Get acceleration	A-30
SetDeceleration	0991h		X		X	Set deceleration	A-31
GetDeceleration	0792h			X		Get deceleration	A-31
SetPosition	0910h		X		X	Set position destination	A-32
SetActualPosition	094Dh		X			Set the actual encoder position	A-32
GetActualPosition	0737h			X		Get the actual encoder position	A-33
GetCommandedVelocity	071Eh			X		Get the commanded velocity	A-33
GetCommandedPosition	071Dh			X		Get the commanded position	A-34
GetPositionError	0799h			X		Get actual position error	A-34
SetPositionErrorLimit	0997h		X			Set maximum position error limit	A-35
GetActualVelocity	07ADh			X		Get the actual encoder velocity	A-35
SetVelocity	0911h		X		X	Set velocity limit	A-36
SetJerk	0913h		X		X	Set jerk (for S-Curve)	A-37
SetGearRatio	0914h		X		X	Set command electronic gear ratio	A-38
GetCaptureValue	0736h			X		Get current axis position capture value and reset the capture	A-39
SetProfileMode	0DA0h		X		X	Set profile mode (S-curve, trapezoïdal, velocity-contouring or electronic gear)	A-39
SetCaptureSource	0DD8h		X			Set capture source (Ref.switch, Index, or AxisIn)	A-40
SetStopMode	0DD0h		X		X	Set stop command (abrupt stop, smooth stop, or	A-41

Parameter 2		**Parameter 3**				**Description**	**Page**
Symbol	Code	none	in	out	buff		
						none)	
SetGearMaster	11AEh		X			Set master axis and source (actual or target-based)	A-42
SetAxisOutSource	15EDh		X			Set axis out monitor signal source	A-43
ClearPositionError	1B47h	X			X	Set position error to 0	A-44
RdIdent	0D00h			X		Read module signature	A-45

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

* marked instructions are affecting the whole module (both axis). The first parameter can specify axis one or two of the module.

** marked instructions need parameter 3 to secify the axis concerned (axis 1, axis 2 or both). The first parameter can specify axis one or two of the module.

All other instructions affect the individual axis.

Reset **Instruction:** Reset the chipset

Function Bloc **Exec**

Function Description:
Reset restores the module to its initial condition, setting all variables to their default values.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 - 14	
= 2	Instruction			Reset	
= 3	Empty PCD register or ʾrNotUsed’	R			no data coming back

The default values are shown in the following table:

Acceleration	0	Kp	0
ActualPosition	0	Kvff	0
AutoStopMode	0	LimitMode	1
AxisMode	1	MotionCompleteMode	0
AxisOutSource	0	MotorBias	0
Breakpoint 1	0	MotorCommand	0
Breakpoint 2	0	MotorLimit	32767
BreakpointValue 1	0	MotorMode	1
BreakpointValue 2	0	OutputMode	1
BufferLength	0	Position	0
BufferReadIndex	0	PositionErrorLimit	32767
BufferStart	200h	ProfileMode	0
BufferWriteIndex	0	SampleTime	204
CaptureSource	0	SettleTime	0
Deceleration	0	SettleWindow	0
DerivativeTime	1	SignalSense	0
EncoderModulus	0	Stop	0
EncoderSource	0	TraceMode	0
GearMaster	0	TracePeriod	1
GearRatio	0	TraceStart	0
IntegrationLimit	0	TraceStop	0
InterruptMask	0	TraceVariable 1	0
Jerk	0	TraceVariable 2	0
Kaff	0	TraceVariable 3	0
Kd	0	TraceVariable 4	0
Ki	0	TrackingWindow	0
Kout	65535	Velocity	0

All axes are enabled at reset.
Profile, servo filter, and other axis-specific parameters are reset on all axes.
External-memory buffer parameters are reset for all buffers. BufferStart is reset to (200h), the lowest user-accessible address.
Axis-specific conditions are reset on all axes. External-memory buffer conditions are reset on all 16 memory buffers.

NoOperation **Instruction:** Performs no operation

Function Bloc **Exec**

Function description:
The NoOperation command has no affect on the chipset. It is useful as a “null” operation to verify communications with the Motion Processor.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			NoOperation	
= 3	Empty PCD register or 'rNotUsed'	R			no data coming back

ClearInterrupt **Instruction:** Reset the interrupt line

Function Bloc **Exec**

Function description:
ClearInterrupt resets the AxisEvent_x_y 1) signal to its inactive state. If interrupts are still pending, the signal will return to its active state within one cycle. It is used after an interrupt has been recognized and processed within the PCD program. This command does not affect the Event Status Register. To clear the acitve event a ResetEventStatus instruction has to be executed.
If this command is executed when no interrupts are pending, it has no effect.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			ClearInterrupt	
= 3	Empty PCD register or 'rNotUsed'	R			no data coming back

see GetInterruptAxis, SetInterruptMask

1) AxisEvent_x_y is a pre-defined variable for input 18 of the module.
The variable is defined for max. 7 modules
module 1: AxisEvent_1_2
module 2: AxisEvent_3_4 a.s.o.

GetTime **Instruction:**

Function Bloc **Exec**

Function description:
Returns the number of cycles that have occurred since the processor was last initialized or reset.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetTime	
= 3	Time	R	Integer	0 – 2^{32}-1	

GetHostIOError **Instruction:** Gets Host I/O error code

Function Bloc **Exec**

Function description:
GetHostIOError returns the code for the last Host I/O error, then resets to 0 both the *error* and the Host I/O bit in the Status-Read word. Generally this command is issued only after the Host I/O error bit in the Status-read word indicates there was an I/O error.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 - 14	
= 2	Instruction			GetHostIOError	
= 3	ErrorCode	R	Integer		*Error code*

Error code	*Description*
0	No error
1	Processor Reset
2	Invalid instruction
3	Invalid axis
4	Invalid parameter
5	Trace running
6	*reserved*
7	Block out of bounds
8	Trace buffer zero
9	Bad serial checksum
10	Not primary port
11	Invalid negative value
12	Invalid parameter change
13	Invalid move after limit condition
14	Invalid move into limit

MultiUpdate

Instruction: both axis immediate parameter update

Function Bloc **Exec**

Function description:
MultiUpdate causes an Update to occur on all axes whose corresponding bit is set to 1 in the mask argument. After this command is executed, and for those axes which are selected using the mask, all buffered data parameters are copied into the corresponding run-time registers.
The following instruction is buffered: ClearPositionError.
The following trajectory parameters are buffered: Acceleration, Deceleration, GearRatio, Jerk, Position, ProfileMode, StartVelocity, StopMode, and Velocity.
The following PID filter parameters are buffered: DerivativeTime, IntegrationLimit, Kaff, Kd, Ki, Kp, and Kvff.
The following Motor Command parameter is buffered: SetMotorCommand

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			MultiUpdate	
= 3	Mask	R		1 - 3	1: Axis 1 2: Axis 2 3: Axis 1+2

If the command SetPosition is executed before a MultiUpdate command, the movement will be started when the MultiUpdate command is executed but the EventStatusRegister will not be reset.
Note: always use the StartMotion command to start a motion. The EventStatusRegister is reset with this command.

StartMotion

Instruction: Starts motion

Function Bloc **Exec**

Function description:
Starts motion on specified axis and clears the EventStatus register.
If this instruction is executed before a SetPosition instruction, only the **fOnDest** flag will be set to zero.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			StartMotion	
= 3	Mask	R		1 – 3	1: Axis 1 2: Axis 2 3: Axis 1+2

SetMotorBias

Instruction: Set Motor offset

Function Bloc **Exec**

Function description:
SetMotorBias sets the bias voltage (offset) of the digital servo filter for the specified axis in mV.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 - 14	
= 2	Instruction			SetMotorBias	
= 3	Offset	R		-10000… .. +10000	Millivolt

see SetMotorCommand

SetDerivativeTime

Instruction: Set derivative sampling time

Function Bloc **Exec**

Function description:
SetDerivativeTime sets the sampling time, in number of servo cycles, for the servo filter to use in calculating the derivative term for the specified *axis*.
Restrictions:
This command is NOT buffered. The new sampling time value will take effect immediately after the command is sent.
This command does not affect the overall cycle time which is 200us sampling time.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 - 14	
= 2	Instruction			SetDerivativeTime	
= 3	PCD register for value	R	integer	0..2^{15}-1	1 = 200us 10 = 2ms 100 = 20ms 1000 =200ms …

see GetDerivative, GetIntegral, MultiUpdate, Update

SetKaff

Instruction: Set acceleration feedforward gain

Function Bloc **Exec**

Function description:
SetKaff sets the acceleration feedforward gain of the digital servo filter for the specified *axis*.
Restrictions:
SetKaff is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

Par.	**Designation/function**	**Type**	**Format**	**Value**	**Comment**
= 1	Axis number	K		1 – 16	
= 2	Instruction			SetKaff	
= 3	Kaff	R		0..2^{15}-1	

see SetKd, SetKi, SetKp, SetKvff, MultiUpdate

SetKd

Instruction: Set derivative gain

Function Bloc **Exec**

Function description:
SetKd sets the derivative gain of the digital servo filter for the specified axis.
Restrictions:
SetKd is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

Par.	**Designation/function**	**Type**	**Format**	**Value**	**Comment**
= 1	Axis number	K		1 - 14	
= 2	Instruction:			SetKd	
= 3	Kd	R	integer	0..2^{15}-1	

see SetKaff, SetKi, SetKp, SetKvff, MultiUpdate

SetKi

Instruction: Set integral gain

Function Bloc **Exec**

Function description:
SetKi sets the integral gain of the digital servo filter for the specified *axis*.
Restrictions:
SetKi is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetKi	
= 3	Ki	R		0..2^{15}-1	

see SetKaff, SetKd, SetKp, SetKvff, MultiUpdate

SetKp

Instruction: Set proportional gain

Function Bloc **Exec**

Function description:
SetKp sets the proportional gain of the digital servo filter for the specified *axis*.
Restrictions:
SetKp is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetKp	
= 3	Kp	R		0..2^{15}-1	

see SetKaff, SetKi, SetKd, SetKvff, MultiUpdate

SetKvff

Instruction: Set velocity feedforward gain

Function Bloc **Exec**

Function description:
SetKvff sets the velocity feedforward gain of the digital servo filter for the specified *axis*.
Restrictions:
SetKvff is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetKvff	
= 3	Kvff	R	integer	0..2^{15}-1	

see SetKaff, SetKi, SetKd, SetKp, MultiUpdate

SetIntegrationLimit

Instruction: Set integration limit

Function Bloc **Exec**

Function description:
SetIntegrationLimit loads the integration-limit register of the digital servo filter for the specified *axis*.
IntegrationLimit value of 100 will limit the total accumulated integration error to 25,600 count*cycles.
Restrictions:
This is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetIntegrationLimit	
= 3	IntegrationLimit	R		0..2^{15}-1	

see GetDerivative, SetDerivativeTime, MultiUpdate

SetKout

Instruction: Set output scale factor of digital servo filter

Function Bloc **Exec**

Funtion Description:
Sets the output scale factor of the digital servo filter for the specified axis. The default value of Kout is 65535.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetKout	
= 3	Kout	R		0..2^{16}-1	

see Set/GetKaff, Set/GetKi, Set/GetKp, Set/GetKvff

SetAxisMode

Instruction: Set axis operation mode

Function Bloc **Exec**

Function description:
SetAxisMode enables (On) or disables (Off) the specified *axis*. A disabled axis will not respond to profile or other motion commands and the servo-loop is not active.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetAxisMode	
= 3	Value	R	integer	0, 1	0: OFF 1: ON

SetLimitSwitchMode

Instruction: Set limit switching

Function Bloc **Exec**

Function description:
SetLimitSwitchMode enables (On) or disables (Off) limit-switch sensing for the specified *axis*. When the mode is enabled, the axis will cause the corresponding limit-switch bits in the Event Status register and Activity Status register to be set when it enters either the positive or negative limit switches and the axis will be immediately stopped. When it is disabled these bits are not set, regardless of whether the axis is in a limit switch or not.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetLimitSwitch-Mode	
= 3	Value	R	integer	0, 1	0: OFF 1: ON

see GetActivityStatus, GetEventStatus

SetMotionCompleteMode

Instruction: Set motion complete mode

Function Bloc **Exec**

Function description:
SetMotionCompleteMode establishes the source for the comparison which determines the motion-complete status for the specified *axis*.
When set to commanded mode the motion is considered complete when the profile velocity reaches zero and no further motion will occur without an additional host command.
This mode is unaffected by the actual encoder location.
When set to actual mode the motion complete bit will be set when the above condition is true AND the actual encoder position has been within the Settle Window for the number of servo loops specified by SettleTime (SettleWindow and SettleTime set in FB Init). The settle "timer" is started at zero at the end of the trajectory profile motion so at a minimum a delay of SettleTime cycles will occur after the trajectory profile motion is complete.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetMotionCom-pleteMode	
= 3	Value	R	integer	0, 1	0: commanded 1: actual

SetMotorMode **Instruction:** Set motor mode

Function Bloc **Exec**

Function description:
SetMotorMode determines the mode of motor operation. When set to On, the axis is in *closed-loop* mode, and is controlled by the output of the servo filter.
When set to Off, the axis is in *open-loop* mode, and is controlled by commands placed directly into the motor output register by the host.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetMotorMode	
= 3	Value	R	integer	0, 1	0: open loop 1: closed loop

see GetActivityStatus, Set/GetMotorCommand

SetAutoStopMode **Instruction:** Set auto stop on position error

Function Bloc **Exec**

Function description:
SetAutoStopMode determines the behavior of the specified *axis* when a motion error occurs. When auto stop is enabled (SetAutoStopMode 1), the axis goes into open-loop mode when a motion error occurs. When Auto-Stop is disabled (SetAutoStopMode 0), the axis is not affected by a motion error.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 16	
= 2	Instruction			SetAutoStopMode	
= 3	Value	R	integer	0, 1	0: no stop 1: stop on error

see GetEventStatus

SetInterruptMask

Instruction: Set interrupt mask

Function Bloc **Exec**

Function description:
SetInterruptMask determines which bits in the Event Status register of the specified axis will cause a host interrupt. For each interrupt mask bit that is set to 1, the corresponding Event Status register bit will cause an interrupt when that status register bit goes active (is set to 1). Interrupt mask bits set to 0 will not generate interrupts.
GetInterruptMask returns the current mask for the specified axis.
Example: The interrupt mask value 28h will generate an interrupt when either the "in positive limit" bit or the "capture received" bit of the event status register goes active (set to 1).

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetInterruptMask	
= 3	Mask	R			*interupt mask**

see ClearInterrupt, GetInterruptAxis

* *interruptMask*	
Motion complete	0001h
Wrap-around	0002h
Breakpoint 1	0004h
Capture received	0008h
Motion error	0010h
In positive limit	0020h
In negative limit	0040h
Instruction error	0080h
Commutation error	0800h
Breakpoint 2	4000h

GetInterruptAxis

Instruction: Get axis with pending interrupt

Function Bloc **Exec**

Function description:
GetInterruptAxis returns a field which identifies all axes with pending interrupts. Axis numbers are assigned to the low-order four bits of the returned word; bits corresponding to interrupting axes are set to 1. If the host interrupt signal has not been set, the returned word is 0.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetInterruptAxis	
= 3	Value	R	integer	0,1,2	1: Axis 1 2: Axis 2

see ClearInterrupt, Set/GetInterruptMask

GetActivityStatus

Instruction: Get activity status

Function Bloc **Exec**

Function description:
GetActivityStatus reads the 16 bit activity status register for the specified *axis*. Each of the bits in this register continuously indicate the state of the chipset without any action on the part of the host. There is no direct way to set or clear the state of these bits, since they are controlled by the chip set.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetActivityStatus	
= 3	ActivityStatus	R	31bit		Information is bit-coded*

* Name	Bit Number	Description
Phasing initialized	0	always 0
At maximum velocity	1	Set to 1 when the trajectory is at maximum velocity. This bit is determined by the trajectory generator, not the actual encoder position.
Tracking	2	Set to 1 when the axis is within the tracking window
Current profile mode	3-5	Contains trajectory mode encoded as follows: bit 5 bit 4 bit 3 Profile Mode 0 0 0 trapezoidal 0 0 1 velocity contouring 0 1 0 s-curve 0 1 1 electronic gear
reserved	6	not used, may be 0 or 1
Axis settled	7	Set to 1 when the axis is settled
Motor on/off	8	Set to 1 when motor mode is on, 0 when off.
Position capture	9	Set to 1 when a value has been captured by the high speed position capture hardware but has not yet been read. The GetCaptureValue command must be executed before another capture can occur.
In-motion	10	Set to 1 when the trajectory generator is executing a profile on the axis.
In positive limit	11	Set to 1 when the positive limit switch is active
In negative limit	12	Set to 1 when the negative limit switch is active
Profile segment	13 -15	Only used during S-curve profile mode. Contains value of 0 when the profile is at rest. Contains phase number 1-7 when profile is in motion.
not used	16 - 31	

GetEventStatus

Instruction: Get event status word

Function Bloc **Exec**

Function description:
GetEventStatus reads the event register for the specified *axis*.
The following table shows the encoding of the data returned by this command.

Restrictions:
All of the bits in this status word are set by the chipset and cleared by the host. To clear these bits use the ResetEventStatus command.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetEventStatus	
= 3	EventStatus	R	31 bit		Information is bit-coded*

see GetActivityStatus, GetSignalStatus

* Name	Bit Number	Description
Motion complete	0	Set to 1 when motion is completed. SetMotionCompleteMode determines if this bit is based on the trajectory generator position or the encoder position.
Wrap-around	1	Set to 1 when the actual (encoder) position wraps from maximum allowed position to minimum or vice versa
Breakpoint 1	2	Set to 1 when breakpoint 1 is triggered
Capture received	3	Set to 1 when a position capture occurs
Motion error	4	Set to 1 when a motion error occurs
In positive limit	5	Set to 1 when the axis enters a positive limit switch condition
In negative limit	6	Set to 1 when the axis enters a negative limit switch condition
Instruction error	7	Set to 1 when instruction error occurs
reserved	8 - 10	Not used, may be 0 or 1.
Commutation error	11	Set to 1 when a commutation error occurs
reserved	12 - 13	Not used, may be 0 or 1.
Breakpoint 2	14	Set to 1 when breakpoint 2 is triggered
not used	15 - 31	

The following flags are refreshed executing this instruction:

fOnDest_n
fWrapAround_n
fBrkPt1_n
fBrkPt2_n
fCaptureTrig_n

fPosErr_n

(n corresponding to axis number)

GetSignalStatus

Instruction: Get current axis signal status register

Function Bloc **Exec**

Function description:
GetSignalStatus returns the contents of the signal status register for the specified axis. The signal status register contains the current value of the various hardware signals connected to each axis of the chipset. The value read is combined with the signal sense register (SetSignalSense command) and then returned to the user. For each bit in the signal sense register that is set to 1 the corresponding bit in the GetSignalStatus command will be inverted, so that a low signal will be read as 1 and a high signal will be read as a 0. Conversely for each bit in the signal sense register that is set to 0 the corresponding bit in the GetSignalStatus command is not inverted, so that a low signal will be read as 0 and a high signal will be read as a 1.
All the bits in the GetSignalStatus command are inputs except for AxisOut. The value read for this bit is equal to the current value output by the axis out mechanism.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetSignalSense	
= 3	SignalStatus	R			Information is bit-coded *

see GetActivityStatus, GetEventStatus

*Returned data Description (status of)	Bit Number
Encoder A input	0
Encoder B input	1
Encoder Index input	2
Ref. Switch	3
Positive limit (LS1)	4
Negative limit (LS2)	5
AxisIn	6
not used	7
not used	8
not used	9
AxisOut	10
not used	11..31

GetSignalSense **Instruction:** Get interpretation of signal status bits

Function Bloc **Exec**

Function description:
GetSignalSense returns the current signal sense mask.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetSignalSense	
= 3	SignalSense	R	31 bit		Information is bit-coded *

see SetSignalStatus

Indicator	mask	Bit Number
Encoder A	0001h	0
Encoder B	0002h	1
Encoder Index	0004h	2
Encoder Home	0008h	3
Positive limit	0010h	4
Negative limit	0020h	5
AxisIn	0040h	6
not used	0080h	7
not used	0100h	8
not used	0200h	9
AxisOut	0400h	10
StepOutput	0800h	11
MotorOutput	1000h	12
not used		13 – 31

SetSignalSense

Instruction: Set interpretation of signal sense bits

Function Bloc **Exec**

Function description:
SetSignalSense establishes the sense of the signals connected to the Signal Sense register by using a bitwise mask that corresponds to the bits of the Signal Status register, for the specified *axis*.
For each sense bit that is 0, the input is active low, or not inverted.
For each sense bit that is 1, the input is active high, or inverted.
Inverting the MotorOutput has the effect of reversing the direction of motion when a positive or negative motor command is given.
The default setting for this register is all bits are 0. (active low)

Restrictions:
Inverting ther encoder A,B, or index may prevent the index capture mechanism from operating correctly.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetSignalSense	
= 3	SignalSense	R	31 bit		Information is bit-coded *

*** see** GetSignalStatus

Indicator	mask	Bit Number
Encoder A	0001h	0
Encoder B	0002h	1
Encoder Index	0004h	2
Encoder Home	0008h	3
Positive limit	0010h	4
Negative limit	0020h	5
AxisIn	0040h	6
not used	0080h	7
not used	0100h	8
not used	0200h	9
AxisOut	0400h	10
StepOutput	0800h	11
MotorOutput	1000h	12
not used		13 – 31

ResetEventStatus **Instruction:** Reset event status register

Function Bloc **Exec**

Function description:
ResetEventStatus clears (sets to 0) , for the specified *axis*, each bit in the EventStatus Register that has a value of 1 in the *mask* sent with this command. All other Event Status register bits (bits which have a mask value of 0) are unaffected.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			ResetEventStatus	
= 3	EventStatus	R	integer	31 bit	

see GetEventStatus

Name	mask	Bit Number
Motion complete	0001h	0
Wrap-around	0002h	1
Breakpoint 1	0004h	2
Capture received	0008h	3
Motion error	0010h	4
In positive limit	0020h	5
In negative limit	0040h	6
Instruction error	0080h	7
Commutation error	0800h	11
Breakpoint 2	4000h	14
no used		15 – 31

SetMotorCommand

Instruction: Set direct value to motor output

Function Bloc **Exec**

Function description:
SetMotorCommand loads the motor-command buffer register of the specified *axis*.
For the MC2400 series, this command is used to control the magnitude of the output waveform.
Scaling example:
If it is desired that a motor command value of 13.7 % of full scale be output to the motor than this register should be loaded with a value of 13.7 *32,768/100 = 4,489 (decimal).
Restrictions:
SetMotorCommand is valid only when the motor is "off".
SetMotorCommand is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetMotorCommand	
= 3		R	integer	$-2^{15} – 2^{15}-1$	

see SetMotorBias, SetMotorLimit, SetMotorMode, MultiUpdate

GetMotorCommand

Instruction: Get buffered motor output command

Function Bloc **Exec**

Function description:
GetMotorCommand reads the contents of the motor-command buffer register.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetMotorCommand	
= 3		R	integer	16 bit	(+/-) 10,000 mV

see SetMotorBias, SetMotorLimit, SetMotorMode, MultiUpdate

SetAcceleration

Instruction: Set acceleration limit

Function Bloc **Exec**

Function description:
SetAcceleration loads the maximum acceleration buffer register for the specified *axis*. This command is used with the Trapezoidal, Velocity Contouring, and S-curve profiling modes.

Restrictions:
SetAcceleration may not be issued while an axis is in motion with the S-curve profile.
SetAcceleration is not valid in Electronic Gearing profile mode.
SetAcceleration is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetAcceleration	
= 3		R	integer	31 bit	Unit/s^2

see Set/GetDeceleration, Set/GetJerk, Set/GetPosition, Set/GetVelocity, MultiUpdate

GetAcceleration

Instruction: Get acceleration limit

Function Bloc **Exec**

Function description:
GetAcceleration reads the maximum acceleration buffer register set by the previous SetAcceleration command.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetAcceleration	
= 3		R	integer	31 bit	Unit/s^2

SetDeceleration **Instruction:** Set deceleration limit

Function Bloc **Exec**

Function description:
SetDeceleration loads the maximum deceleration buffer register for the specified *axis*. This command sets the magnitude of the deceleration register, which always has a negative sign.

Restrictions:
This is a buffered command. The new value set will not take effect until the next MultiUpdate instruction is entered.
This command is used with the Trapezoidal, S-curve, and Velocity contouring profile modes. It is not used with the electronic gearing profile mode.

Note: If deceleration is set to zero, then the value specified for acceleration (SetAcceleration) will automatically be used to set the magnitude of deceleration.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetDeceleration	
= 3		R	integer	31 bit	Unit/s^2

see Set/GetAcceleration, SetJerk, SetPosition, SetVelocity, MultiUpdate

GetDeceleration **Instruction:** Get deceleration limit

Function Bloc **Exec**

Function description:
GetDeceleration reads the Maximum Deceleration buffer.
Restrictions:
This command is used with the Trapezoidal, S-curve, and Velocity contouring profile modes. It is not used with the electronic gearing profile mode.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetDeceleration	
= 3	PCD register for value	R	integer	31 bit	Unit/s^2

see Set/GetAcceleration, SetJerk, SetPosition, SetVelocity, MultiUpdate

SetPosition

Instruction: Set position destiniation

Function Bloc **Exec**

Function description:
SetPosition specifies the trajectory destination of the specified *axis*. It is used in the Trapezoidal and S-curve profile modes.
Restrictions:
SetPosition is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetPosition	
= 3		R	integer	+/- 31 bit	units

see Set/GetAcceleration, Set/GetDeceleration, SetJerk, SetVelocity, GetPositionError, SetPositionErrorLimit, MultiUpdate

SetActualPosition

Instruction: Set actual encoder position

Function Bloc **Exec**

Function description:
SetActualPosition loads the actual position register (encoder position) for the specified *axis*. At the same time, the current commanded position is replaced by the loaded value minus the current actual position error. This prevents a servo "bump" when the new axis position is established. The destination position (see SetPosition) is also modified by this amount so that no trajectory motion will occur when the update instruction is issued. In effect, this instruction establishes a new reference position from which subsequent positions can be calculated. It is commonly used to set a known reference position after a homing procedure.

Note: The position error is zeroed.
SetActualPosition takes effect immediately, it is not buffered.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetActualPosition	
= 3		R	integer	31 bit	units

see GetPositionError; GetActualVelocity, Set/GetActualPositionUnits, AdjustActualPosition

GetActualPosition **Instruction:** Get actual encoder position

Function Bloc **Exec**

Function description:

GetActualPosition reads the contents of the encoder's actual position register. This value will be the result of the last encoder input, which will be accurate to within one cycle.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetActualPosition	
= 3	PCD register for value	R	integer	31 bit	Units

GetCommandedVelocity **Instruction:** Get commanded velocity

Function Bloc **Exec**

Function description:
GetCommandedVelocity returns the current commanded velocity value for the specified *axis*. Commanded velocity is the instantaneous velocity value output by the trajectory generator.
This command functions in all profile modes.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetCommanded-Velocity	
= 3	PCD register for value	R	integer	31 bit	Units

see GetCommandedAcceleration, GetCommandedPosition

GetCommandedPosition

Instruction: Get commanded position

Function Bloc **Exec**

Function description:
GetCommandedPosition returns the current commanded position for the specified *axis*. Commanded position is the instantaneous position value output by the trajectory generator.
This command functions in all profile modes.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetCommandedPo-sition	
= 3	PCD register for value	R	integer	31 bit	Units

see GetCommandedAcceleration, GetCommandedVelocity

GetPositionError

Instruction: Get position error

Function Bloc **Exec**

Function description:
GetPositionError returns the current position error of the specified *axis*. The error is the difference between the actual position (encoder position) and the commanded position (instantaneous output of the trajectory generator).

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetPositionError	
= 3	PCD register for value	R	integer	31 bit	Units

see Set/GetPosition, Set/GetPositionErrorLimit

SetPositionErrorLimit **Instruction:** Set maximum position error limit

Function Bloc **Exec**

Function description:
SetPositionErrorLimit sets the absolute value of the maximum position error allowable by the chipset for the specified *axis*. If the position error exceeds this limit, a motion error occurs. Such a motion error may or may not cause the axis to stop moving depending on the value set using the SetAutoStopMode command.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetPositionError-Limit	
= 3		R	integer	31 bit	Units

see GetPositionError, GetActualPosition, SetPosition

GetActualVelocity **Instruction:** Get actual encoder velocity

Function Bloc **Exec**

Function description:
GetActualVelocity reads the current actual velocity for the specified *axis*. This value is the result of the last encoder input, so it will be accurate to within one cycle.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetActualVelocity	
= 3	PCD register for value	R	integer	31 bit	Units/s *

see Set/GetActualP

* Resolution is 16 bit only !!

SetVelocity **Instruction:** Set velocity limit

Function Bloc **Exec**

Function description:
SetVelocity loads the Maximum Velocity buffer register for the specified *axis*.

Restrictions:
SetVelocity may not be issued while an axis is in motion with the S-curve profile.
SetVelocity is not valid in Electronic Gearing profile mode.
The velocity must not be < 0 except in the Velocity-Contouring profile mode.
SetVelocity is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	**Designation/function**	**Type**	**Format**	**Value**	**Comment**
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetVelocity	
= 3		R	integer	+/- 31 bit	Units/s

see Set/GetAcceleration, Set/GetDeceleration, SetJerk, SetPosition, MultiUpdate

SetJerk **Instruction:** Set jerk

Function Bloc **Exec**

Function description:
SetJerk loads the jerk register in the parameter buffer for the specified *axis*.
In S-Curve mode every cycletime acceleration will get biger by the jerk until the maximum acceleration reached.
Note:
if jerk is set to 0, no motion will start, because acceleratin stays at 0!

Restrictions:
SetJerk is a buffered command. The value set using this command will not take effect until the next Update or MultiUpdate instruction.
This command is used only with the S-curve profile mode. It is not used with the trapezoidal, velocity contouring, or electronic gear profile modes.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetJerk	
= 3	Jerk	R	integer	31 bit	unit/s^3

see Set/GetAcceleration, Set/GetDeceleration, SetPosition, SetVelocity, MultiUpdate

$$\text{Jerk [unit/s}^3\text{]} = \frac{\text{Acc [unit/s}^2\text{]}}{t_{\text{S-curve}}\text{ [s]}}$$

acc	maximum acceleration, deceleration to reach (in S-Curve the same)
$t_{\text{S-curve}}$	time to reach maximum acceleration

SetGearRatio **Instruction:** Set electronic gear ratio

Function Bloc **Exec**

Function description:
SetGearRatio sets the ratio between the master and slave axes for the electronic gearing profile for the current *axis*. Positive ratios cause the slave to move in the same direction as the master, negative ratios in the opposite direction. The specified ratio has a unity scaling of 65,536.
Restrictions:
This is a buffered command. The new value set will not take effect until the next MultiUpdate instruction is entered.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetGearRatio	
= 3	GearRatio	R	integer	+/- 31 bit	

see SetGearMaster, MultiUpdate

$$\text{GearRatio} = \frac{\text{Ratio (Slave/Master) x 65536 x Machine Factor}_{\text{Slave}}}{\text{Machine Factor}_{\text{Master}}}$$

$$\text{Ratio (Slave/Master)} = \frac{\text{GearRatio x Machine Factor}_{\text{Master}}}{\text{65536 x Machine Factor}_{\text{Slave}}}$$

If the Machine Factor is equal on both axis, a ratio of 65536 means 1:1

GetCaptureValue **Instruction:** Get current axis position capture value

Function Bloc **Exec**

Function description:
GetCaptureValue returns the contents of the Position Capture Register for the specified *axis*. This command also resets the capture hardware to allow another capture to occur.
Restrictions:
The PositionCapture Bit 3 in the EventStatus register does not reset automatically. Use the instruction ResetEventStatus with the corresponding mask.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			GetCaptureValue	
= 3	PCD register for value	R	integer	31 bit	

see SetCaptureSource

SetProfileMode **Instruction:** Set curve profile

Function Bloc **Exec**

Function description:
SetProfileMode sets the profile mode, selecting Trapezoidal, Velocity Contouring, S-curve, Electronic gear or External for the specified *axis*.
Restrictions:
SetProfileMode is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetProfileMode	
= 3	ProfileMode	R		0 - 4	*

see SetGearMaster, SetGearRatio, MultiUpdate

* Profile	Value
Trapezoidal	0
Velocity contouring	1
S-curve	2
Electronic gear	3
External *	4

* for future use

SetCaptureSource **Instruction:** Set capture source

Function Bloc **Exec**

Function description:
SetCaptureSource determines which of the signals, Encoderindex, Ref.switch or AxisIn, is used to trigger the high-speed capture of the actual axis position for the specified *axis*.
If Encoderindex selected as capture source, a position capture will trigger if encoder signals A, B and Index achieve a particular state. This behavior is defined by the SignalSense register using SetSignalSense instruction. Default value in the SignalSense register is 0, resulting in a position capture if all signals achieve low state.
If AxisIn selected only Index is replaced by AxisIn. The capture trigger is still depending of encoder signals A and B.

Restrictions:
In SSI mode, the Position Capture is not available at all.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetCaptureSource	
= 3	CaptureSource	R	integer	31 bit	0: Index 1: Ref.switch 2. AxisIn

See page 5 – 15 of Chapter 5 of this manual for more information on the high speed capture register.

SetStopMode

Instruction: Set stop mode

Function Bloc **Exec**

Function description:
SetStopMode stops the specified axis. The available stop modes are AbruptStop, which instantly (without any deceleration phase) stops the axis, SmoothStop which uses the programmed deceleration value and profile shape for the current profile mode to stop the axis, or NoStop which is generally used to turn off a previously set stop command.

Note: After an Update a buffered stop command (SetStopMode command) will reset to the NoStop condition. In other words if the command SetStopMode is followed by an Update command and then by a GetStopMode command, the retrieved stop mode will be NoStop.
GetStopMode returns the stop mode set using SetStopMode.

Restrictions:
SmoothStop mode is not available in the electronic-gearing profile.
SetStopMode is a buffered command. The value set using this command will not take effect until the next MultiUpdate instruction.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetStopMode	
= 3	StopMode	R		0, 1, 2	0: NoStop 1: AbruptStop 2: SmoothStop

see MultiUpdate

SetGearMaster **Instruction:** Set master axis and source

Function Bloc **Exec**

Function description:
SetGearMaster establishes the slave (*axis*) and master (*masterAxis*) axes for the electronic-gearing profile, and sets the source, Actual or Commanded, of the master axis position data to be used.
The MasterAxis determines what axis will drive the slave axis. Both the slave and the master axes must be enabled (SetAxisMode command).

The SlaveSource determines whether the master axis' commanded position (as determined by the trajectory generator) will be used to drive the slave axis, or whether the master axis' actual (encoder) position will be used to drive the slave.

Restrictions:
For electronic gear mode to operate properly the master axis must be enabled.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetGearMaster	
= 3	GearMaster	R	integer	31 bit	*

see Set/GetGearRatio

* Value	SlaveSource Bit 8	MasterAxis Bit 0
	0: actual position	0: Axis 1
	1: commanded position	1: Axis 2

all other bits to zero!

Example:

0H:	Master is axis 1, Slave is axis 2 with commanded position driven by actual position of master axis
101H:	Master is axis 2, Slave is axis 1 with commanded position driven by the commanded position on master axis

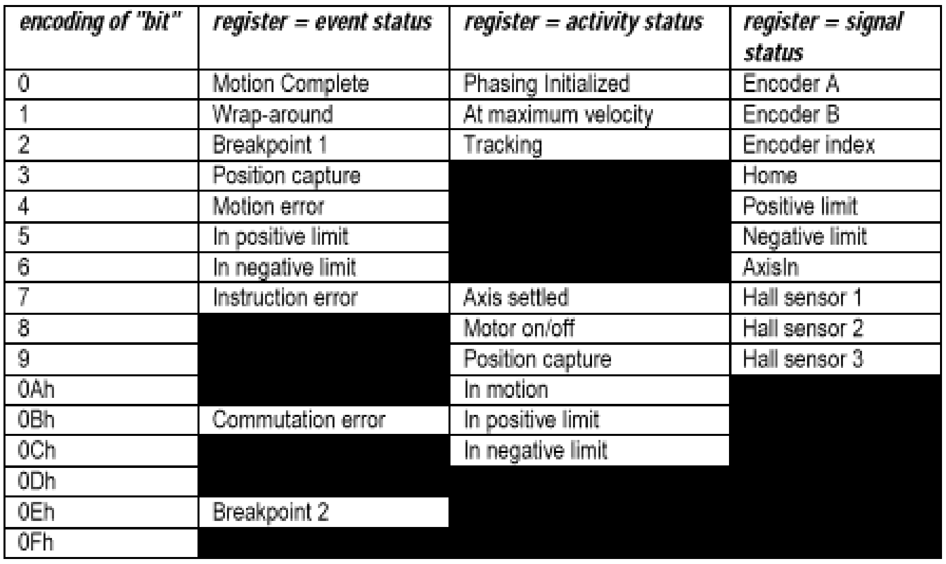
SetAxisOutSource **Instruction:** Set axis out monitor signal source

Function Bloc **Exec**

Function description:
SetAxisOutSource maps the specified *bit* of the specified status *register* of *axisn* to the AxisOut pin for the specified *axis*. The state of the AxisOut pin will thereafter track the state of *bit*. If *register* is absent (encoding of 0), *bit* is ignored, and the specified AxisOut pin is, in effect, turned off (inactive).

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			SetAxisOutSource	
= 3	AxisOutSrc	R			*

*** Register Bits**	*register*	*bit*	*sourceAxis*
31 12	11 8	7 4	3 0
not used	0: (none) 1: EventStatus 2: ActivityStatus 3: SignalStatus	see table below	0: Axis 1 1: Axis 2



encoding of "bit"	*register = event status*	*register = activity status*	*register = signal status*
0	Motion Complete	Phasing Initialized	Encoder A
1	Wrap-around	At maximum velocity	Encoder B
2	Breakpoint 1	Tracking	Encoder index
3	Position capture		Home
4	Motion error		Positive limit
5	In positive limit		Negative limit
6	In negative limit		AxisIn
7	Instruction error	Axis settled	Hall sensor 1
8		Motor on/off	Hall sensor 2
9		Position capture	Hall sensor 3
0Ah		In motion	
0Bh	Commutation error	In positive limit	
0Ch		In negative limit	
0Dh			
0Eh	Breakpoint 2		
0Fh			

ClearPositionError **Instruction:** Set position error to zero

Function Bloc **Exec**

Function description:
ClearPositionError sets the current profile's commanded position equal to the actual position (encoder input), thereby clearing the position error for the specified *axis*. This command can be used when the axis is at rest, or when it is moving. If it is used when the axis is moving the host should be aware that the trajectory destination position (used in trapezoidal and s-curve modes) is not changed by this command.

Restrictions:
ClearPositionError is a buffered command. The new value set will not take effect until the next Update or MultiUpdate instruction is entered. This command cannot be executed while the chip is performing an s-curve profile.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Axis number	K		1 – 14	
= 2	Instruction			ClearPositionError	
= 3	rNotUsed	R			

see GetPositionError, MultiUpdate, Set/GetPositionErrorLimit

RdIdent **Instruction:** Read module identifier

Function Bloc **Exec**

Function description:
RdIdent can be used to check correct interconnection between PCD CPU and the module and to verify the FPGA version.
If the interconnection works correctly, the value will be read (see table below). If the module is faulty (or the addressing is wrong), the value 0 is read.

Par.	Designation/function	Type	Format	Value	Comment
= 1	Module number	K		1 – 14	
= 2	Instruction			RdIdent	
= 3	PCD Register	R	integer		Value *

Table of identification codes for H320:

* Value (Hex)	FPGA-Version	Module type
140x	x: 0…F	H320
142x	x: 0…F	H322
145x	x: 0…F	H325
147x	x: 0…F	H327

Notes